Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature

Michael Osborne

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary Term 2010

Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature

---

Summary

We develop a family of Bayesian algorithms built around Gaussian processes for various problems posed by sensor networks. We firstly introduce an iterative Gaussian process for multi-sensor inference problems, and show how our algorithm is able to cope with data that may be noisy, missing, delayed and/or correlated. Our algorithm can also effectively manage data that features changepoints, such as sensor faults. Extensions to our algorithm allow us to tackle some of the decision problems faced in sensor networks, including observation scheduling. Along these lines, we also propose a general method of global optimisation, *Gaussian process global optimisation* (GPGO), and demonstrate how it may be used for sensor placement.

Our algorithms operate within a complete Bayesian probabilistic framework. As such, we show how the hyperparameters of our system can be marginalised by use of Bayesian quadrature, a principled method of approximate integration. Similar techniques also allow us to produce full posterior distributions for any hyperparameters of interest, such as the location of changepoints. We frame the selection of the positions of the hyperparameter samples required by Bayesian quadrature as a decision problem, with the aim of minimising the uncertainty we possess about the values of the integrals we are approximating. Taking this approach, we have developed *sampling for Bayesian quadrature* (SBQ), a principled competitor to Monte Carlo methods.

We conclude by testing our proposals on real weather sensor networks. We further benchmark GPGO on a wide range of canonical test problems, over which it achieves a significant improvement on its competitors. Finally, the efficacy of SBQ is demonstrated in the context of both prediction and optimisation.

Contents

Acknowledgements

This work was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership. Thanks are owed to the UK Meteorological office for the use of the MIDAS dataset. We would also like to thank B. Blaydes of the Bramblemet/Chimet Support Group, and W. Heaps of Associated British Ports (ABP) for allowing us access to the weather sensor network, hosting our RDF data on the sensor web sites, and for providing raw sensor data as required.

Personally, I would like to thank:

E. T. Jaynes, for while I think I would probably have found him a fairly crotchety gent if we had ever met, his writings changed my life;

Alex Rogers, for being an excellent collaborator and source of the sensor network motivation that drove the entire thesis;

Richard Mann, for almost proof-reading and having me in his home over Christmas, even if I did have to do Morris Dancing;

Roman Garnett, without whom this thesis would never have happened, for loaning me vast sums of money without complaint, sorting out my computer stuff without making me feel like a complete loser and making sure the excitement of co-authored papers lasted until seconds before the hard submission deadlines;

Min Lee and HJ Lee, for being excellent Korean dudes who supplied the Kalman Filter and VBHMM methods tested against;

Steve Roberts, for allowing me complete academic freedom and not being too upset by how long this thesis process took.

My thesis is based on a number of joint-authored peer-reviewed publications. These are listed below, followed by a description of my contribution to them.

A. Rogers, M. Osborne, S. Ramchurn, S. Roberts, and N. R. Jennings. Information Agents for Pervasive Sensor Networks. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 294–299. IEEE Computer Society Washington, DC, USA, 2008.

M. A. Osborne, A. Rogers, S. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 109–120, April 2008.

M. A. Osborne, A. Rogers, S. J. Roberts, S. D. Ramchurn, and N. R. Jennings. Bayesian Gaussian process models for multi-sensor time-series prediction. In *Inference and Learning in Dynamic Models*. Cambridge University Press, 2010.

I contributed all theory and Gaussian process experimental results. Rogers provided the problem motivation, Rogers and Ramchurn provided the experimental data and real-world implementations. Roberts and Jennings provided advice on general theory.

M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimisation. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009.

Core GPGO theory was joint work between myself and Garnett. I supplied extensions, including the optimisation of noisy functions. Garnett generated the extensive experimental results. Roberts provided some initial problem motivation.

R. Garnett, M. A. Osborne, and S. Roberts. Sequential Bayesian prediction in the presence of changepoints. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009

S. Reece, R. Garnett, M. A. Osborne, and S. J. Roberts. Anomaly detection and removal using non-stationary Gaussian processes. Technical report, University of Oxford, Oxford, UK, 2009.

R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 2010. (To appear).

I supplied all theory concerning Bayesian quadrature for hyperparameter posteriors and marginalisation, as well as for changes in observation likelihood. Garnett supplied all changepoint covariances (the bulk of the novelty of the papers), with the exception of the smooth drastic change in covariance, due to Reece. Garnett was also responsible for all experimental tests. Roberts supplied problem motivation and some data.

M. A. Osborne, R. Garnett, and S. Roberts. Active data selection for sensor networks with faults and changepoints. In *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia*. ACM New York, NY, USA, 2010. (To appear).

All novel theory (for active data selection) was my responsibility. Garnett generated all experimental results. Roberts supplied some data and general theory.

R. Garnett, M. A. Osborne, and S. Roberts. Bayesian optimization for sensor set selection. In *International Conference on Information Processing in Sensor Networks (IPSN 2010)*, 2010. (To appear).

All theory was joint work between myself and Garnett. Garnett was again responsible for all experimental tests. Roberts provided advice on problem context.

Chapter 1

Introduction

1.1 Sensor Networks

Sensor networks are an increasingly integral component of modern science, enabling the monitoring of environments both hostile and remote. Early sensor networks comprised little more than automated data loggers, passively collecting data for later off-line scientific analysis. More recent sensor networks typically make current data available through the internet, and are finding use in the real-time monitoring of environmental events such as floods or storm events (see Hart and Martinez [2006] for a review of such environmental sensor networks).

Using real-time sensor data in this manner presents many novel challenges. Many of the information processing tasks that would previously have been performed off-line by the owner of an environmental sensor network must now be performed in real-time, and are being entrusted to algorithms rather than human judgement. Furthermore, it may also be desirable to use the trends and correlations observed in previous data to predict the value of environmental parameters into the future, or to predict the reading at a sensor that is temporarily unavailable (e.g. due to network outages). Finally, we note that modern networks are often open, in the sense that additional sensors may be deployed, and existing sensors may be removed or repositioned at any time. This means that our tasks may have to be performed with only limited knowledge of the precise location, reliability, and accuracy of each sensor.

Even given perfect knowledge about our sensors, the data they provide is often

confounded by any number of problematic features. Firstly, as previously noted, data will frequently be missing. Worse, such sensor failure is often caused by abnormal events that are of particular interest to us. Where we have multiple sensors, their readings will almost invariably be correlated; the information provided by each is dependent on what readings we have already received. Similarly, the measurements of some sensors may be delayed relative to others.

Our data may also experience abrupt changes in its fundamental characteristics, known as *changepoints*. For example, a data stream might undergo a sudden shift in its mean or variance; a periodic signal might have a change in period, amplitude, or phase; or a signal might undergo a change so drastic that its behaviour after a particular point in time is completely independent of what happened before. Our sensors are particularly prone to misleading failures of various types: a sensor reading may become stuck at an incorrect value; may suddenly obtain a bias from the real values; or may gradually drift away from the real values. A robust prediction algorithm must be able to make accurate predictions even under such unfavorable conditions.

Note that taking an observation from a sensor at all is often associated with a non-negligible expenditure of energy, time or both. Further, if interested in covertness, an observation will often increase a sensor's risk of detection. Hence we may require our network to actively make decisions about which observation to take – when, where and of what type – with the goal of making only a small number of highly informative observations. Similar problems are found in the question of how many sensors are required to monitor a particular phenomenon, and how those sensors should be initially deployed.

1.2 Solution Concepts

We abstract our required tasks as being executed by an *agent*. As we'll discuss in Chapter 2, an agent possesses two faculties: the ability to reason and the power to act. That is, the agent is defined firstly by what knowledge it has, on the basis

of which it can employ Bayesian probability theory in order to perform inference about any other unknown quantities. It is defined secondly by its goals, according to which Bayesian decision theory will uniquely determine which actions it takes. We introduce algorithms along these lines to address the problems of prediction and decision required by sensor network applications.

In particular, we consider the range of Bayesian tools afforded us by Gaussian processes (see Chapter 3). These will allow us to address all the challenges posed by sensor networks outlined above. Implementing our algorithms, however, poses many numerical issues. Gaussian processes are traditionally used for regression on fixed data sets, to produce (at significant computational cost) a single fixed set of predictions. However, we require dynamic and on-line algorithms for our sensor network applications. To meet this requirement, in Chapter 5 we propose an iterative formulation that will allow us to efficiently update our predictions through time. As the Gaussian process gives us complete probability distributions, we can produce both mean predictions and associated measures of uncertainty. We will show how such a measure can be used to retain only the most informative data, leading to further gains in the efficiency of our algorithm. The uncertainty of our Gaussian process can also be used as a measure of the value of a prospective observation. Taking the minimisation of this uncertainty as our goal, we can employ decision theory to take only measurements that are expected to be sufficiently informative.

However, the problems of optimisation required by decision theory are only infrequently tractable. As such, we are usually forced to adopt numerical methods of optimisation. In Chapter 6, we frame the problem of optimisation as itself a (simpler) decision problem. Essentially, we sequentially select our observations of the objective function in order to optimise the final expected value found by our algorithm. This leads us to a novel algorithm, *Gaussian process global optimisation* (GPGO). A myopic version of our algorithm is similar to a previous proposal [Jones et al., 1998] employing Gaussian processes (under the name "kriging"). However, we can improve further upon this criterion by considering multiple function evaluations into the future. Our

clear Bayesian formalism also permits us to benefit from the specification of the required confidence in our returned value. Further, our use of Gaussian processes allows us to incorporate useful prior information about the objective function (such as periodicity), as well as learning from observations of the derivative. This final fact also leads to an innovative resolution of conditioning issues. We also present an extension of our method to the optimization of noisy functions.

We now turn to how best to implement in software the other tool of an agent, probability theory. The key practical challenge issued here is integration. In the context of sensor networks, we need to perform non-analytic integrals in order to sum over the essentially infinite number of possible models that might potentially describe the environment. Further non-analytic integrals give us the full posterior probability distribution of our models given the data. This allows us, for example, to determine the posterior distribution for the location of any changepoints in our data.

In Chapter 7, we address this problem of integration as one of Bayesian inference in its own right. In particular, we employ the methods of Bayesian quadrature [O'Hagan, 1991]. Essentially, we take a Gaussian process for the integrand, using what samples of it are available in order to perform inference about the value of the integrand elsewhere. We then turn again to decision theory and introduce the *sampling for Bayesian quadrature* (SBQ) method in Chapter 8. SBQ selects our set of integrand samples with the goal of ensuring our estimate of our integral is as good as possible, in the sense of minimising our uncertainty about its value. This represents us a principled and effective competitor to Monte Carlo methods.

In summary, our adoption of Bayesian techniques has given us a theoretical approach to many of the challenges of sensor networks. It has also given us algorithms designed with our computational constraints in mind, providing principled approximative techniques for both integration and optimisation. Importantly, these form the chief difficulties in implementing, respectively, the probability and decision theory required by agents. As such, we are hopeful our work will form the basis for building more sophisticated agents.

Chapter 2

Bayesian Theory

2.1 Probability Theory

This thesis is fundamentally concerned with how agents should reason under uncertainty. The foundation for research into any such problem is probability theory, as excellently presented by Jaynes [2003]. We explicitly state that we are concerned with a Bayesian approach to probability, in which the probability $P(A|I)$ is interpreted as the degree of belief in proposition A given that proposition I is known to be true. Notationally, I is often used as a 'catch-all' for background information; it represents the sum of all information we possess relevant to the inference at hand. In such a case, we will call I the *background*, or *context* [Buntine, 1994]. Note we interpret all probabilities as being conditional on the information available in this way.

We also consider any probability to be unique given the conditioning information. That is, $P(A|I)$ unambiguously determines a single probability value, without the need for any subscripts or other definition. Any agent provided with information I, no more and no less, should arrive at the same probability for proposition A. Bayesian probability is subjective only in the sense that results will depend on the information available to the relevant agent.

To further illustrate this point, consider $P(A|I)$ as a measure of the degree to which I logically implies A [Jeffreys, 1998]. That is, this probability is the truth-value associated with the statement $I \Rightarrow A$. If A is logically deducible from I, e.g. if I is the statement 'the ball is red and made of rubber' while A is the statement 'the

ball is red', the probability is one. Similarly, if the two propositions are inconsistent, e.g. if now A is the statement 'the ball is white', the resulting probability is zero. Probability values between these two extremes concern the grey area in which it is possible to neither categorically deny nor confirm a proposition with only the information at hand. Bayesian probability theory, while being entirely consistent with traditional deductive logic, allows the consideration of a much wider realm of possible propositions. In this sense, probability theory should be viewed as *extended logic*. This extension is far more relevant to the kind of everyday reasoning at which humans are so adept – 'Now, where did I leave my keys?'. For this reason, Bayesian theory is often described as being just common sense, expressed mathematically.

To briefly summarise the mathematical laws of probability[1], for probabilities taken to be real numbers between zero and one, we have the two relationships

$$P(\neg A|I) + P(A|I) = 1 \tag{2.1.1a}$$

$$P(A,\, B|I) = P(A|I)\, P(B|A, I) = P(B|I)\, P(A|B, I)\,, \tag{2.1.1b}$$

where $P(A, B|I)$ will be understood to mean the probability of the logical conjunction $P(A \wedge B|I)$. $\neg A$ implies the negation of the proposition A, that is, $\neg A$ is the proposition that A is false. Interestingly, these two operations, conjunction and negation, form a sufficient set of operations to generate all functions of propositions. These laws are important enough to warrant their own names – (2.1.1a) is known as the *Sum Rule* and (2.1.1b) the *Product Rule*.

Bayesian inference is built upon the rearrangement of the Product Rule (2.1.1b),

$$P(B|A, I) = \frac{P(A|B, I)\, P(B|I)}{P(A|I)}\,,$$

which is known as Bayes' Rule. $P(B|I)$ is known as the *prior* for B, representing our state of belief about B before learning A. $P(A|B, I)$ is the *likelihood*[2] for B

[1] These are derivable from various reasonable sets of postulates about how degrees of belief should behave, notably those taken by Cox [1946]. Further developments of Cox's ideas have been made by Knuth [2005].

[2] The term likelihood is most often used to describe a function of B for fixed A: $L(B) = P(A|B, I)$.

given A, reflecting the impact of the new information A. $P(A|I)$ is the *evidence* for A; a normalisation constant that can be written as $P(A|I) = P(A|B,I)\,P(B|I) + P(A|\neg B,I)\,P(\neg B|I)$ and thus expressed purely in terms of prior and likelihood terms. The combination of these factors thus gives us the *posterior* for B given A, $P(B|A,I)$. This term represents our state of belief about B after having learned A – Bayes' rule allows us to update our probabilities in the light of new information! Bayes' Rule hence provides a canonical rule for how any agent should reason on the basis of what information it has.

Our analysis will commonly be concerned with the value of some variable. In our notation, the propositional variable (presumably unknown), will be denoted using a ring superscript, as per $\mathring{x}$, and the un-superscripted form for one of its possible values, as x. As an example, $\mathring{x}$ might be 'the colour of the ball' and x might be 'red'. The set of propositions $\{\mathring{x}{=}x;\, \forall x\}$, then, are exhaustive and mutually exclusive – that is, one and only one of them is true. We can then consider the probability distribution $P(\,\mathring{x} = \cdot \mid I\,)$ over x. Where clear, $\mathring{x}$ may be dropped to give $P(x|I)$, or $P(\cdot|I)$ if we wish to refer to the distribution itself. We can use the laws of probability (2.1.1) to write the normalisation condition,

$$\sum_x P(\,\mathring{x}{=}x \mid I\,) = 1\,, \tag{2.1.2}$$

and also what is termed the *marginalisation* of a variable $\mathring{y}$,

$$P(\,\mathring{x}{=}x \mid I\,) = \sum_y P(\,\mathring{x}{=}x,\, \mathring{y}{=}y \mid I\,)\ . \tag{2.1.3}$$

Similarly, the lower-case $p(\,\mathring{x}{=}x \mid I\,)$ will be used to denote the *probability density function* (pdf) for a variable $\mathring{x}$ that may take a continuum of values[3]. Again, $\mathring{x}$ may be dropped for notational convenience. This quantity is defined by

$$p(\,\mathring{x}{=}x \mid I\,) \triangleq \lim_{\delta x \to 0} \frac{P(\,x \leq \mathring{x} < x + \delta x \mid I\,)}{\delta x}\,. \tag{2.1.4}$$

The typical problem of Bayesian inference is to infer some parameter B given experimental data A, hence consideration of a function of the parameter $L(B)$ has significant intuitive appeal. Note, however, that $L(B)$ is not a probability for B; for example, $\int L(B)\,\mathrm{d}B$ will not necessarily be one.

[3]Note that it is the author's belief that the universe is itself fundamentally discrete, real numbers a fiction introduced to efficiently model it. Units are, then, a consequence of this continuous approximation; units are not ontologic.

As noted by Jaynes [2003], this limit is in general a non-trivial operation. Ignoring it and proceeding as though the laws of probability must necessarily apply to continuous variables can lead to error. However, in practice, so long as we restrict ourselves to finite, normalisable pdfs, we can be justified in using pdfs almost exactly as if they were probabilities [Bretthorst, 1999]. The replacement of the sums in (2.1.2) and (2.1.3) with appropriate integrals give the equivalent relationships for pdfs,

$$1 = \int p(\mathring{x}{=}x \mid I)\, \mathrm{d}x \tag{2.1.5}$$

$$p(\mathring{x}{=}x \mid I) = \int p(\mathring{x}{=}x, \mathring{y}{=}y \mid I)\, \mathrm{d}y\,. \tag{2.1.6}$$

However, there are still a few minor hazards to be aware of. Importantly, (2.1.4) clearly implies that $p(\mathring{x}{=}x \mid I)$ is *not* an invariant quantity. As the left hand side of (2.1.4) is a dimensionless probability, and $\mathrm{d}x$ has identical units to x, $p(\mathring{x}{=}x \mid I)$ must have units equal to the inverse of those of x. Hence it is incorrect to apply functions of dimensionless quantities to a pdf of a quantity with units – only if x is dimensionless are expressions such as $\log p(\mathring{x}{=}x \mid I)$ and $\exp p(\mathring{x}{=}x \mid I)$ valid.

We are often interested in making an isomorphic transformation of variables $x \to y = f(x)$. In this case, our requirement that there be the same probability mass around x and y leads to the expression

$$\begin{aligned} P(x \leq \mathring{x} < x + \mathrm{d}x \mid I) &= P(y \leq \mathring{y} < y + \mathrm{d}y \mid I) \\ p(\mathring{x}{=}x \mid I)\, \mathrm{d}x &= p(\mathring{y}{=}y \mid I)\, \mathrm{d}y \\ p(\mathring{x}{=}x \mid I) &= p(\mathring{y}{=}y \mid I) \left| \frac{\partial y}{\partial x} \right| . \end{aligned} \tag{2.1.7}$$

Hence in changing variables, the pdf is scaled by the Jacobian of the transformation. An alternative way of seeing the same thing is to use the change of variables $y = f(x')$

in (2.1.6), giving

$$\begin{aligned} p(\mathring{x}{=}x \mid I) &= \int p(\mathring{x}{=}x \mid \mathring{y}{=}y, I)\; p(\mathring{y}{=}y \mid I)\; \mathrm{d}y \\ &= \int \delta\big(x - f^{-1}(y)\big)\; p(\mathring{y}{=}y \mid I)\; \mathrm{d}y \\ &= \int \delta(x - x_*)\; p(\mathring{y}{=}f(x_*) \mid I)\; \left|\frac{\partial y}{\partial x_*}\right| \mathrm{d}x_* \\ &= p(\mathring{y}{=}f(x) \mid I)\; \left|\frac{\partial y}{\partial x}\right| . \end{aligned} \tag{2.1.8}$$

where $\delta(x-a)$ is a Dirac delta density[4] in x centred at a. As an example, if we transform variables as $y = \log x$, a uniform $p(\mathring{y}{=}f(x) \mid I)$ corresponds to a $p(\mathring{x}{=}x \mid I)$ with form $\frac{1}{x}$ for $x \geq 0$.

For many applications, it is desirable to provide a low-dimensional characterisation of a pdf. Taking the maximum of pdfs for such a purpose, while common, is prone to danger. It's quite possible that the maximum x_m of a pdf $p(x \mid I)$ occurs at a very narrow peak, such that the probability $P(x_m \leq \mathring{x} < x_m + \epsilon \mid I)$ of x lying within some small distance ϵ of x_m is negligible. To avoid such problems, we define the useful summary statistics of a pdf, the *mean* (or *expected value*)

$$m(\mathring{x} \,|\, I) \triangleq \int x\, p(x|I)\, \mathrm{d}x\,,$$

and *covariance*

$$C(\mathring{x} \,|\, I) \triangleq \int \big(x - m(\mathring{x} \,|\, I)\big)^{\mathsf{T}} (x - m(\mathring{x} \,|\, I)\big)\, p(x|I)\, \mathrm{d}x\,.$$

The covariance of a single variable is known as its *variance*, and its square root, its *standard deviation* (SD). Note that, however, these statistics are sensitive to representation. If we make the isomorphic transformation of variables $x \to y = f(x)$, it is generally true that

$$m(\mathring{y} \,|\, I) \neq f(m(\mathring{x} \,|\, I))\,.$$

This is equally true of the mean of a (discrete) probability distribution. We must hence be careful to request the mean of the variable we are actually interested in.

[4]Note that the identical notation $\delta(x-a)$ will also be used to describe a Kronecker delta function in the discrete variable x. Which delta is intended can be simply determined by considering whether the relevant variable is discrete or continuous.

2.2 Decision Theory

Thus far, we have shown how an agent should employ probability theory in order to perform inference about its environment. Decision theory allows us to extend the capacity of an agent to include a prescriptive theory of how best to act, to influence its environment. In order to do so, we first need to define for our agent a *loss function*, which specifies its goals. The loss function assigns to each proposition a real number representing the cost should it be true (for example, the monetary cost of the proposition's realisation). Decision theory can then be simply summarised: Given the choice of an action $\mathring{a}$, we select the action a that minimises our expected loss $\int \lambda(x)\, p(\,\mathring{x}{=}x \mid \mathring{a} = a,\, I\,)\,\mathrm{d}x$, where $\mathring{x}$ is the relevant state of the environment, and the loss associated with it taking the value x is defined as $\lambda(x)$. Alternatively, a *utility function* could be assigned, expressing the desirability of each proposition. In this case, we would choose the action that maximises our expected utility. Clearly the negative of any utility function is an equivalent loss function; we can trivially switch between the two approaches according to our preference.

In selecting loss functions for our agents, it is important never to lose sight of our own goals as system architects. As put by Russell and Norvig [2003]:

> As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

To illustrate with an example from Russell and Norvig, consider designing a loss function for an agent in control of a vacuum cleaner. One approach might be to reward our agent for the quantity of dust it sucks up. However, such a loss function might result in our agent vacuuming up dust, dumping it all on the floor and then vacuuming it up once more – doubling its reward! A more clear-headed and robust approach would be to reward our agent for the cleanliness of the floor, the quantity we are actually interested in.

In addition to this prescriptive formulation, the agent model can also be used in a descriptive sense. Almost anything can be modelled within the agent framework. Take, for example, the case of a ball placed on a hill-side, 'deciding' upon which direction to roll. Here we might model the ball as having a loss function equal to its height above sea level, so that its goal is to get as low as possible. Its information we might restrict to a measurement of the gradient at its current position, along with a linear model of the hillside. If in a minimum, even a local one, it will not roll at all. The true power of the agent framework, however, is realised when applied to biological entities, which have been granted by evolution very clear goals. This applies equally to software created by such entities to achieve their ends; we'll talk more about modelling software as agents later.

2.3 Bayesian Networks

Bayesian networks (BNs) are a useful graphical expression of the conditional relationships between variables in a model. Jensen [1996] and Heckerman [1999] provide introductions to the field, certainly sufficient for the use we will be making of them in this work. More thorough treatment is available in Pearl [1988] and coverage of the more general subject of graphical models can be found in Lauritzen [1996].

To illustrate the concept, consider a vignette in which Ian hears the sounds of a crowd's cheers and applause emerging from the radio broadcast of the cricket in the next room. Given that Ian is an avid cricket fan, he is very keen to discover what has happened in the game, but by the time he makes it to the radio, the commentators have moved onto discussing the quality of the catering at the ground and do not immediately reveal the current score. Define the propositions C : 'The crowd have applauded', H : 'One of the batsmen has made a hundred' and W : 'A wicket has

fallen'. Ian's prior information I leads him to believe that

$$\begin{aligned} P(W, H|I) &= P(W|I)\, P(H|I) \\ \Rightarrow P(W|H, I) &= P(W|I) \\ \Rightarrow P(H|W, I) &= P(H|I)\,. \end{aligned} \tag{2.3.1}$$

For any propositions that obey the relationship (2.3.1), we say that W and H are *independent* given I. This is equivalent to the statement that, given I, learning H does not change your belief about W, and vice versa. This might be the case if I contained the fact that, the last time Ian checked, one of the two batsmen was clearly struggling on a low score (and hence was likely only to get out), while the other was in excellent form and had raced to ninety-nine runs (and hence was likely only to make a century). Either or both of W and H might now be true, the probability of one not influencing the probability of the other. I also causes Ian to believe, quite understandably, that W and H both influence the probability of C occurring – the event of either W or H is likely to induce the crowd to cheer. We can express the probability of these three propositions by using firstly (2.1.1b) and then (2.3.1) to give

$$\begin{aligned} P(W, H, C|I) &= P(W, H|I)\, P(C|W, H, I) \\ &= P(W|I)\, P(H|I)\, P(C|W, H, I)\,. \end{aligned} \tag{2.3.2}$$

We can then use a BN to represent this *joint* probability, meaning the probability of all propositions in our system given what knowledge we have.

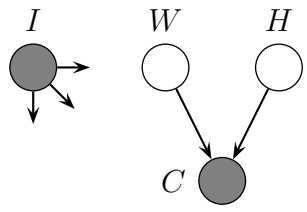


Figure 2.1: Bayesian Network for cricket vignette.

In our notation, nodes correspond exactly to the propositions considered by probability theory – the two terms will be used interchangeably. When we consider a

variable $\mathring{x}$, we will let its node correspond to the exhaustive and mutually exclusive set of propositions $\{\mathring{x}=x;\, \forall x\}$. Shaded nodes correspond to propositions whose truth is actually known – hence, when a variable node is shaded, its true value is known. Such nodes, which commonly correspond to observations, are said to be *instantiated*. Any prior knowledge nodes such as I are then instantiated by definition.

Lines between nodes indicate that there is a probabilistic connection between them. Before we can be more precise, we need to introduce some terminology. If there is an arrow from a node A to another node B, A is said be the *parent* of B and similarly B is said to be the *child* of A. We will use $Pa(B)$ and $Ch(B)$ to describe the set of parents and children respectively of a node B. Similarly, to say that a node is a *descendent* or *ancestor* of another is to assert exactly the relationships that might be expected from the use of these terms in genealogy.

We can now state the probabilistic constraint expressed by a BN: Any set of nodes are independent given all of (and only) their ancestors.

A context node, as with I in Figure 2.1, is depicted with unterminated arrows protruding from it. Such nodes are defined as being a parent of every other node in the BN, as their graphical representation is intended to suggest. This notation implies that the probabilities of *all* other nodes in the BN are considered to be dependent on such nodes. Indeed, in general, we would expect any node to be connected to all others in this way – the probability of any node could depend on the value of any other. That they are not is a consequence of the presence of independence. In fact, BNs are special in that independence given I is taken as the default, assumed if no lines are drawn to indicate otherwise. This is the reason I is depicted as a universal parent node.

BNs are useful precisely because such independence is common, or at least a good approximation. Our justification for (2.3.1) within the context of our vignette is an example of such an approximation.

On this topic, note that the joint of three nodes A, B and C, in which A and

C are believed independent given B and I can be expressed in multiple forms,

$$\begin{aligned} P(A, B, C \mid I) &= P(B \mid A, I)\, P(C \mid B, I)\, P(A \mid I) \\ &= P(A \mid B, I)\, P(C \mid B, I)\, P(B \mid I) \\ &= P(B \mid A, I)\, P(B \mid C, I)\, P(C \mid I)\ . \end{aligned}$$

These are illustrated in Figures 2.2a, 2.2b and 2.2c. They are, however, distinct from the distribution in which A and C are held to be independent given I alone,

$$P(A, B, C \mid I') = P(B \mid A, C, I')\, P(C \mid I')\, P(A \mid I')$$

as in Figure 2.2d. What Figure 2.2 is intended to demonstrate is that the directionality of the lines is somewhat arbitrary. Indeed, any BN can be rewritten as a 'moralised graph' [Lauritzen, 1996], in which parents are 'married' (that is, joined by a line) and directional arrows removed. However, in such a graph (Figure 2.2f), the visual distinction between the probability distributions expressed by Figure 2.2d and Figure 2.2e (in which there is no independence whatsoever) is removed[5]. The use of directional arrows allows us to clearly express the form Figure 2.2d, capturing a common form of independence.

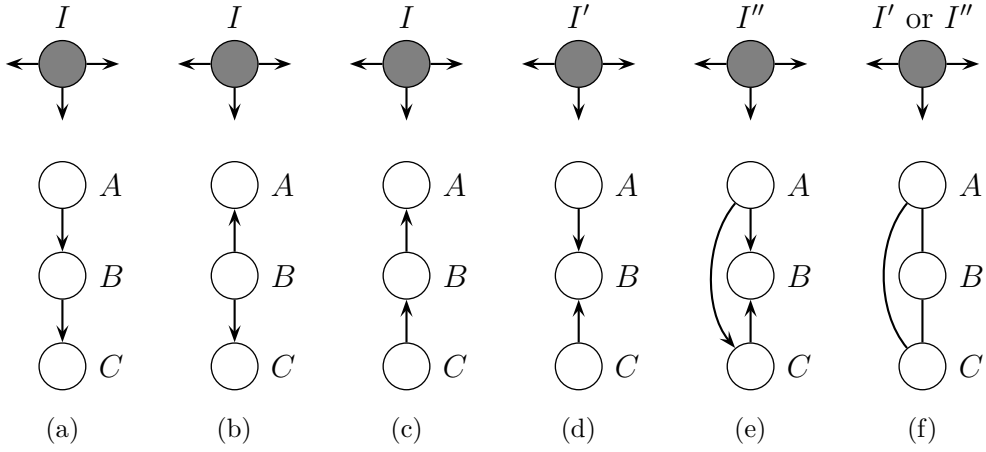


Figure 2.2: BNs (a), (b) and (c) are identical probabilistically, but are distinct from BN (d). The moral graph (f) is identical for both (d) and (e).

[5]The relevant independence properties could, of course, still be specified by appropriate choices of probability distribution.

We can now generalise the connection between joint distributions and BNs suggested by the simple example (2.3.2). A joint probability is expressible by any expansion according to the Product Rule (2.1.1b) of the form

$$P(\, S \mid I \,) = \prod_{n=1}^{N} P\left(\, S_n \, \middle| \, \bigcup_{m=1}^{n-1} S_m,\, I \, \right) ,$$

where $\{S_n;\, n = 1, \ldots, N\}$ represents a convenient partitioning of the set of propositions $S = \bigcup_{n=1}^{N} S_n$. As an example, to reproduce (2.3.2), write $S_1 = \{W, H\}$ and $S_2 = \{C\}$. As that example showed, the most convenient partitioning to choose is clearly one that allows us to exploit the independencies given I that exist in our model. A BN guides us in the selection of this partitioning.

The role of the context I here is crucial. A BN is a depiction of a joint distribution conditioned on I and I alone; the arrows drawn reflect only the independence structure given I. Within a different context, the nodes will, in general, be related in a completely different way. As such, there can only ever be a single context for any given BN; the context defines a BN. This distinguishes the context from other instantiated nodes. The convention of a BN is that the illustrated conditional relationships would be identical even if these other nodes were unknown. The importance of context to conditional dependence is illustrated by the famous Yule-Simpson Paradox, as discussed by, for example, Pearl [2000].

Now, consider the construction of a joint over nodes S from a BN given a context I. First, find the set of nodes S_1 that have no parents other than I. Constructing $P(\, S_1 \mid I \,)$ is then easy – conditioned on their parents I, all nodes in S_1 are independent. Hence $P(\, S_1 \mid I \,)$ is a product of independent terms, as per (2.3.1). Then we consider the set of nodes S_2 that have no parents other than S_1 and I. $P(\, S_2 \mid S_1,\, I \,)$ can now be readily constructed by realising that the nodes in S_2 are independent given S_1 and I. Continuing in this way, we arrive at a neatly factored form for the joint $P(\, S \mid I \,)$, for any arbitrary partitioning,

$$P(\, S \mid I \,) = \prod_{n=1}^{N} P(\, S_n \mid Pa(S_n) \,) \ .$$

This economical form for the joint permits many analytical niceties, a strong motivation for any assumptions of independence.

This joint then allows us to determine any other distribution of interest. When working with multiple variables, it is almost always advisable to write down the joint 'probability of everything' first, and then compute any other required probabilities from that. If we are interested in the distribution of only a subset of S given I, then we are required to marginalise, as in (2.1.6). Note that if no other term in the expansion of the joint is dependent on a node $\mathring{x}$, then the marginalisation over x will reduce to a normalisation integral such as $\int P(\, x \mid y,\, I\,)\,\mathrm{d}x = 1$. Hence, nodes without children may be unproblematically marginalised by simply removing the relevant term from the joint – removing the node from the BN. To marginalise a node with children is not much more complicated – we simply connect all the node's parents to all its children in the BN.

Similarly, if we have more information than simply I, we can use the joint and the Product Rule in the form

$$P(\, x \mid y,\, I\,) = \frac{P(\, x,\, y \mid I\,)}{\int P(\, x,\, y \mid I\,)\,\mathrm{d}x}\,,$$

in order to find the distribution conditional on whichever nodes $\mathring{y}$ have been instantiated.

Associated with each node in the BN is not just a probability, but also a loss[6]. This could in general be dependent on the realisation of any other node. To illustrate using our cricket example, the loss associated with C might be different for different values of H, as in the case that Ian gets annoyed by a crowd's refusal to cheer an excellent hundred. That is, we could imagine a *loss network* specifying the interdependent desirabilities of every proposition considered, a loss function over the entire joint set of propositions. The factorisation of this loss function may, in general, be different from the factorisation of the probability function. This noted, however, we

[6]Networks that additionally include loss functions and actions are traditionally referred to as *influence diagrams* [Pearl, 1988]. We, however, abuse the term 'Bayesian network' to include such networks, as we view Bayesian theory as comprising both probability and decision theory.

will henceforth consider only simple loss functions that do not display this kind of dependence.

This loss function, together with the set of instantiated nodes, are of only epistemic relevance to a particular, specified agent. Indeed, a loss function and a set of known propositions *define* an agent. Recall that the probabilities that influence an agent's actions are uniquely determined by the knowledge the agent possesses. With the additional specification of a loss function, the actions that an agent takes are also uniquely determined.

Now, some notation. A node $\mathring{y}$ is said to be *deterministic* given $\mathring{x}$ and I if $p(\,\mathring{y}{=}y \mid \mathring{x}{=}x,\, I\,) = \delta(y - f(x))$ for some function $f(\cdot)$ determined by I. We further require that the set $\mathring{x}$ be minimal, in the sense that $p(\,\mathring{y}{=}y \mid \mathring{x}_s{=}x_s,\, I\,)$ is not a delta function for any strict subset $\mathring{x}_s$ of $\mathring{x}$. Note that, of course, $\mathring{x}$ will not be reciprocally deterministic given $\mathring{y}$ unless the map f has an inverse.

If a node is deterministic given all its parents in a BN, we draw this node with a double circle. An example is given by Figure 2.3a, in which we consider that the crowd will applaud only if a batsman makes a hundred or gets out.

A special example of a deterministic node, then, is an action node, that is, one representing the action chosen by an agent. Clearly, the agent will take the action specified by decision theory, and so, an action node is deterministic given all the information available to the agent as well as its loss function. We represent such nodes using a square, as shown in Figure 2.3b. In that example, we consider the crowd as an agent that knows both H and W, and incorporate knowledge of its loss function into I. The loss function might associate a lower cost with taking the action C (than $\neg C$) if the crowd knows H and $\neg W$, and a lower cost with $\neg C$ (than C) if the crowd knows W and $\neg H$. That is, the crowd is lending blatantly one-eyed support to the batting side, cheering if and only if a batsman makes a hundred. In such a case, observing C supplies perfect information about H. Here it has been implicitly assumed that the crowd agent has completely accurate observations of W and H. In general, however, we may need to distinguish between actual variables and an agent's

potentially noisy observations of them, for which we would need to introduce new variables, W' and H'. In our example, learning C only really informs us about the crowd agent's knowledge of W and H, which might in general not consist of delta distributions at their real values. A crowd might see a run scored and cheer what it thinks is consequently a batsmen's century, not realising that the umpire has signalled leg byes. Where such an effect is significant, we need to introduce further variables to represent the noisy information actually available to agents.

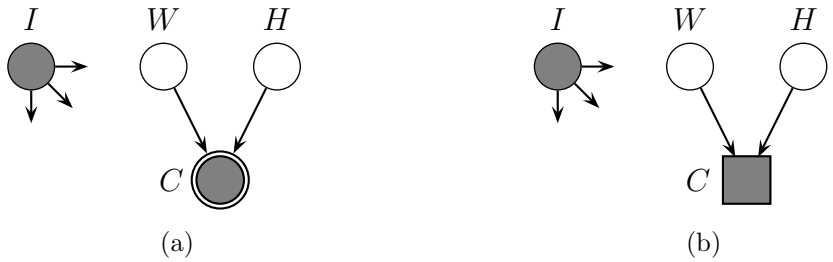


Figure 2.3: BNs illustrating (a) determinism and (b) actions for the cricket vignette.

As noted above, a BN represents the inference and decision making process for only one particular agent. All other agents must be reasoned about by performing inference concerning their loss function and sets of information. In particular, the actions of an agent's future self can only be regarded as those of another agent, and inference performed accordingly. Of course, an agent will normally have highly informative probability distributions for the loss function and information sets of its future self.

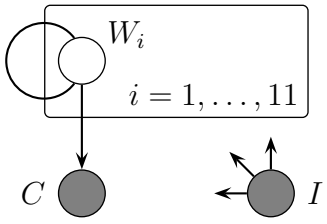


Figure 2.4: A BN illustrating plate notation for the cricket vignette.

Another convenient notational convention is that of *plates* [Buntine, 1994], in which multiple similar nodes are grouped within a rectangle. For example, if our

cricket fan Ian had just switched the radio on for the first time and was uncertain about which batsmen were at the crease, he might wish to entertain the sets of propositions W_i : 'Batsman number i has got out'. The resulting BN is depicted in Figure 2.4, expressing the knowledge that there are eleven W_i nodes, each of which might cause C. The probabilities $p(\, C \mid W_i, \, I \,)$ may not in general be the same for all i, although for most plates such edges will indeed by identical. The arc protruding through the plate indicates that W_i nodes are all dependent on each other – knowing that the sixth batsman has been dismissed clearly influences our state of belief about an opening batsmen having gotten out.

2.4 Priors

One issue we have avoided thus far is where our priors actually come from. While the laws of probability (2.1.1) tell us how to manipulate probabilities, they do not prescribe what values a probability should actually take in light of the available information[7]. We have asserted that $P(\, A \mid B \,)$ is uniquely determined by A and B, but have not described what exactly the relationship is. Our interpretation of $P(\, A \mid B \,)$ as the degree to which B implies A provides some meaning; we want $P(\, A \mid B \,)$ to monotonically increase from zero to one as A varies from impossible to certain in the light of B. Beyond this, then, our task is to describe rules to determine the function $P(\, \cdot \mid \cdot \,)$.

It behooves us as a matter of honesty to admit that this problem is as yet an unresolved one. Nor is it likely to be ever fully solved; what we are asking for is a mapping to the real numbers from the enormously complicated space of all potential sets of knowledge. As Appleby [2005] puts it, the necessity of having to 'guess' a probability – the lack of the complete rules we require – is the source of most of the abhorrence of orthodox frequentist statisticians to Bayesian methods. A plethora of methods have been proposed, as reviewed in Kass and Wasserman [1996]. We outline

[7] Of course, the laws do impose normalisation constraints in the case of mutually exclusive and exhaustive propositions, (2.1.2) and (2.1.5)

here only those principles in agreement with our view of probability theory.

While the methods we will discuss will be explicitly concerned with determining priors, note that exactly the same arguments can be used in the assignment of any probability. In particular, we will often want to use the same techniques to determine the allowable forms of likelihoods. In general, our 'inputs' to a problem of Bayesian inference will be the form of the prior and likelihood, and after cranking the handle of Probability Theory, we obtain 'outputs' that are the posterior and the evidence.

The most fundamental intuition we appeal to in assigning priors is termed the *principle of insufficient reason* [Jaynes, 2003]. Define H as the proposition that 'heads' is the result of a coin toss. There is something deeply appealing about the assignment of a probability of one half to both H and $\neg H$[8]. At its heart, the reasons for this assignment are because our knowledge I is simply insufficient to discriminate between the two, to lead us to believe that one face should be preferred to the other. $I \Rightarrow H$ to exactly the same extent that $I \Rightarrow \neg H$. Given that our knowledge is entirely symmetric concerning the faces of the coin, we are compelled to assign them equal probability. By the same reasoning, we would normally assign a probability of one sixth to each of the possible outcomes of a die roll. This principle can be applied to any other discrete variable about which we are similarly ignorant.

Now consider priors over continuous variables. An intuitive extension of the principle of indifference for a variable $\mathring{x}$ whose outcomes we are unable to distinguish between is the uniform distribution $p(\,\mathring{x} = x \mid I\,) = 1$ – that is, we do not believe any one value of $\mathring{x}$ is any more probable than any other. Due to (2.1.7), however, such a prior corresponds to a non-uniform distribution for the new variable $\mathring{y} = \log \mathring{x}$. The statement of ignorance about $\mathring{x}$ implies a statement of some knowledge about $\mathring{y}$!

Jeffreys [1946, 1998] proposed that priors should be assigned that remain the same even if such a change of variables is made. In particular, he proposes a functional

[8]Of course, we stress again that this probability is entirely epistemic. There is no 'physical' probability, or propensity, of one half intrinsically associated with the coin. Indeed, a coin toss is entirely determined by Newtonian mechanics: given sufficient knowledge of the configuration of the toss, there is no reason why it could not be accurately predicted.

of distributions that is invariant under any change in variables of $p(\cdot \mid I)$. This quantity can then be used to constrain the allowable forms for $p(\cdot \mid I)$ itself.

Such arguments were further developed by Jaynes [2003] in his consideration of *transformation* or *invariance* groups. Jaynes [2003] emphasises that it is unreasonable, however, to expect a prior to be invariant under all changes in variables – we need to specify exactly what kinds of invariance we require of it. Rather than throwing up our hands and claiming 'complete ignorance', we need to ask ourselves 'what is the shape of our ignorance'? For example, consider some variable $\mathring{m}$ for which our state of knowledge could be stated as being indifferent to the possibilities m and $m + a$, for a any constant. That is, we believe m and $m + a$ to be equally probable. Such a variable is known as a *location parameter*. An example might be the exact time at which we think we might have lost a watch – it might have been this morning, or it might equally have been this afternoon. Put another way, our belief distribution would look identical whether we use Greenwich Mean Time or the time in Perth, Australia – it is invariant to such a change in representation. From (2.1.7) we have

$$\begin{aligned} p(\,\mathring{m}{=}m \mid I\,) &= p(\,\mathring{m} + a = m + a \mid I\,) \left|\frac{\partial(m+a)}{\partial m}\right| \\ p(\,\mathring{m}{=}m \mid I\,) &= p(\,\mathring{m} + a = m + a \mid I\,)\ . \end{aligned} \tag{2.4.1}$$

Now, the definition of our ignorance implies that our belief distribution over a variable $(\mathring{m} + A)$ should be exactly the same function as that over $\mathring{m}$,

$$f(x) \triangleq p(\,\mathring{m}{=}x \mid I\,) = p(\,\mathring{m} + a = x \mid I\,)\ .$$

Hence, substituting into (2.4.1), taking the derivative with respect to a and then setting $a = 0$,

$$\begin{aligned} f(m) &= f(m+a) \\ 0 &= f'(m) \\ \alpha &= f(m)\,, \end{aligned}$$

where α is some constant. That is, the location parameter has been assigned exactly the uniform distribution that intuition expects. Now consider a variable $\mathring{s}$ about which we are ignorant in the sense that, for any constant b, we consider s and $b\,s$ equally probable. We term such a variable a *scale parameter*. In terms of our earlier example, this would require that we would be no more sure about the hour in which we lost our watch than the relevant year! Clearly the time of watch loss is not a scale parameter. A better example might be my guess as to the time until we receive a new watch as a gift from a generous friend. For such a variable,

$$g(x) \triangleq p(\,\mathring{s}=x \mid I\,) = p(\,b\,\mathring{s} = x \mid I\,)\ .$$

Hence, from (2.1.7),

$$\begin{aligned} g(s) &= g(b\,s)\,\left|\frac{\partial(b\,s)}{\partial s}\right| \\ g(s) &= g(b\,s)\,b\,, \end{aligned}$$

so, taking the derivative with respect to b and then setting $b = 1$,

$$\begin{aligned} 0 &= g'(s)\,s + g(s) \\ \frac{g'(s)}{g(s)} &= -\frac{1}{s} \\ \ln g(s) &= -\ln s + \ln\beta \\ g(s) &= \frac{\beta}{s}\,, \end{aligned} \tag{2.4.2}$$

for β some constant. (2.4.2) is known as the Jeffreys Prior for a scale parameter.

However, note that the priors returned by these invariance group arguments are improper – that is, they do not satisfy $\int p(\,\mathring{x} = x \mid I\,)\,\mathrm{d}x = 1$. This is due to the fact that such arguments only ever represent an approximation of a true state of knowledge. For the location parameter describing the time at which we think we lost our watch, clearly we are not equally ignorant between having lost it yesterday and having lost it at a time prior to the formation of the solar system. All true priors should encode knowledge of these sorts of limits. However, a proper prior can

always be chosen for which our invariance group prior is a good approximation over the region of interest. Importantly, simple truncation of an improper prior, as in

$$p(\, x \mid I \,) = \begin{cases} \frac{1}{2L} & -L \leq x \leq L \\ 0 & \text{otherwise} \end{cases} ,$$

should be avoided if at all possible, as inferences derived from it can be highly sensitive to the exact selection of the limit L [Berger, 2006]. A better prior might be a Gaussian with a very large variance, which is very nearly uniform around the location of its mean μ. An improper prior should always represent the end-product of some limit applied to the proper prior, as in $m \to \mu$. As emphasised by Dawid et al. [1996], the correct approach is to always use proper priors and then, if possible, pass to such a limit at the very end of a calculation. The priors we use in the remainder of this work will be exclusively proper. In particular, we use Gaussian priors for real parameters, log-Gaussian priors for strictly positive parameters and uniform priors for bounded parameters.

2.5 Second-order Probability

We now consider what meaning, if any, we can assign to the probability of a probability, a *second-order* probability. This will prove of critical importance to our efforts in Chapter 8.

We approached invariance groups as a way of characterising ignorance. We note that this reasoning applies equally well to situations in which we have very precise, positive knowledge of symmetries. For example, consider the archetypical Bernoulli urn. Define $\mathring{x}$ to be the colour of the next ball drawn from the urn. Now imagine two different states of knowledge; I_a, representing the knowledge that 'the urn contains two million balls, which may be either red or white' and I_b 'the urn contains exactly one million white balls and one million red balls'. In both cases, we are equally ignorant about both the possible results of the draw. Neither state of knowledge would lead us to discriminate between $\mathring{x} = \text{Red}$ and $\mathring{x} = \text{White}$. The principle of

insufficient reason applies equally to both, giving

$$P(\mathring{x}=\text{Red} \mid I_a) = \frac{1}{2} \tag{2.5.1a}$$

$$P(\mathring{x}=\text{Red} \mid I_b) = \frac{1}{2}\,. \tag{2.5.1b}$$

However, I_b is well removed from what we would normally consider ignorance – indeed, it actually represents a great deal of highly pertinent information. It seems reasonable [Jaynes, 2003, Chapter 18] that there is something that distinguishes I_b from I_a. In particular, we feel much more 'confident' in the statement (2.5.1b) than in (2.5.1a). To characterise this 'confidence', consider learning the new piece of information $J_d =$ '10 red balls have just been drawn'. If we had previously known only I_a, it seems reasonable that J_d would lead to suspicions that the urn might contain *only* red balls, or at least a much higher proportion of them. However, given I_b, our probability will be almost entirely unchanged, other than of course being revised slightly downwards to account for the drawn balls. Hence,

$$P(\mathring{x}=\text{Red} \mid I_a,\, J_d) \gg \frac{1}{2}$$

$$P(\mathring{x}=\text{Red} \mid I_b,\, J_d) \simeq \frac{1}{2}\,.$$

Hence the assignment (2.5.1b) given I_b seems much more resilient to new information that (2.5.1a) given I_a. So our 'confidence' in a probability seems really dependent on how it would change if given some ancillary information J. We investigate this possibility by defining, for a now arbitrary variable $\mathring{x}$ and arbitrary pieces of information I and J,

$$\mathring{q}(x) \triangleq P(\mathring{x}=x \mid I,\, J)\ .$$

Note that we can consider $\mathring{q}$ a set of exclusive and mutually exhaustive propositions, using exactly the same notational convention we use for any other variable. That is, $\mathring{q} = q$ is the proposition that $P(\mathring{x}=x \mid I,\, J) = q(x)$, that $P(\mathring{x}=\cdot \mid I,\, J)$, uniquely determined by I and J, assumes the particular form $q(\cdot)$. The joint distribution is now represented by Figure 2.5a. As $\mathring{q}$ is a unique probability distribution given I and

J, this node is depicted as deterministic upon I and J,

$$p(\,\mathring{q}{=}q \mid I,\, J\,) = \delta(q - P(\,\mathring{x}{=}\cdot \mid I,\, J\,))\ .$$

Hence the only uncertainty that can exist about $\mathring{q}$ is due to uncertainty about J (or I). Given I, $\mathring{q}$ is just a repackaging of any uncertainty in J. The clear links between this kind of probability $\mathring{q}$ and information J are stressed by de Finetti [1977].

Note also that $\mathring{q}$ screens off $\mathring{x}$ from the influence of I and J. Imagine programming a probability-robot with information I and J; $\mathring{q}$ then represents the (two) numbers in the robot's positronic brain associated with its beliefs about $\mathring{x}$. Any part of I or J that is germane to $\mathring{x}$ has been completely captured by $\mathring{q}$. This relationship is expressed by

$$P(\,x \mid q\,) = q(x)\,.$$

Of course, I and J may still directly influence our beliefs about any variable that is not $\mathring{x}$. Note that clearly $\mathring{q}$ is not exerting any causal influence on $\mathring{x}$ – recall that a Bayesian network represents only our epistemic beliefs.

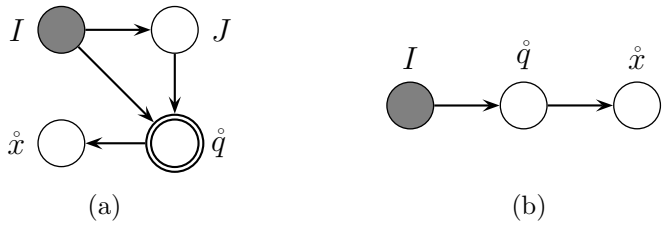


Figure 2.5: Bayesian networks including $\mathring{q}$ as defined in (2.5).

However, now consider that we do not, in fact, know J. Clearly this is the most relevant case to inference – we rarely know exactly what information we might come by in the future. Hence given our lack of certainty about J, the correct thing to do is to marginalise it out,

$$\begin{aligned} p(\,x,\, q \mid I\,) &= \int P(\,x \mid q\,)\, p(\,q \mid I,\, J\,)\, p(\,J \mid I\,)\,\mathrm{d}J \\ &= q(x) \int p(\,q \mid I,\, J\,)\, p(\,J \mid I\,)\,\mathrm{d}J \\ &= q(x)\, p(\,q \mid I\,)\ . \end{aligned}$$

This new joint is depicted in Figure 2.5b. What have we gained by using this representation? One benefit of this new variable is that we are able to use $p(\mathring{q}{=}q \mid I)$ as a convenient substitute for having to specify a distribution over J. The variable J has been considered a representative of any information we could possibly learn that is pertinent to our beliefs about $\mathring{x}$. Equivalently, it could also represent any hypothesis or contingency relevant to $\mathring{x}$. Either way, it is clearly of very high dimension. Our prior over J would likely contain many discontinuities and piecewise components in order to express our beliefs over the enormous space of possible data we could hypothetically receive. It would be very difficult indeed to specify such a distribution. Instead, we have changed variables from J to $\mathring{q}$, for which we must specify a distribution over a probability over x – for Boolean $\mathring{x}$, this requires only a distribution over a single real number between zero and one. The uncertainty in J has been 'folded into' the more convenient $\mathring{q}$.

We now illustrate the significance of the prior $p(q \mid I)$ in the case of the Bernoulli urn example considered earlier. Firstly, note that it is only the mean of this distribution that can in any way affect inference about $\mathring{x}$,

$$P(\mathring{x}{=}\text{Red} \mid I) = \int_0^1 q\, p(q \mid I)\, \mathrm{d}q\,.$$

Hence only this first moment of $p(q \mid I)$ is pertinent to any decisions or inference that in turn depend on $\mathring{x}$, but not otherwise on any of the factors J. For the urn example, this is the case if we are interested in the colour of a drawn ball, but do not otherwise have any direct interest in factors such as the numbers of each type of ball in the urn, how long and in what manner it was shaken, the local acceleration due to gravity etc. that we can imagine affecting which ball we draw. The higher moments of $p(q \mid I)$ become relevant only when we consider the impact of learning some of those factors.

Figure 2.6 depicts possible prior distributions for the different states of knowledge we have considered. To extend the example, we have defined an additional possible state of knowledge I_c = 'the urn contains either exclusively white balls or

exclusively red balls.' Note firstly that all three distributions have the same mean of one half, as required by (2.5.1). Inasmuch as we are interested only in $\mathring{x}$, the three possible states of knowledge are indistinguishable. However, as before, consider learning new information. For the diffuse $p(\, q \mid I_a \,)$, we can easily imagine q, our epistemic probability for x, changing to almost any value between zero and one as we receive new data. If we are better informed, as for I_b, we would expect that q would be more resilient to the receipt of new information. Indeed, the peaked shape of $p(\, q \mid I_b \,)$ suggests that it is very improbable that we would learn something that would markedly change our belief q from one half. In the final case of I_c, what we learn will almost certainly push q to close to either one or zero – seeing even a single ball will reveal the entire composition of the urn.

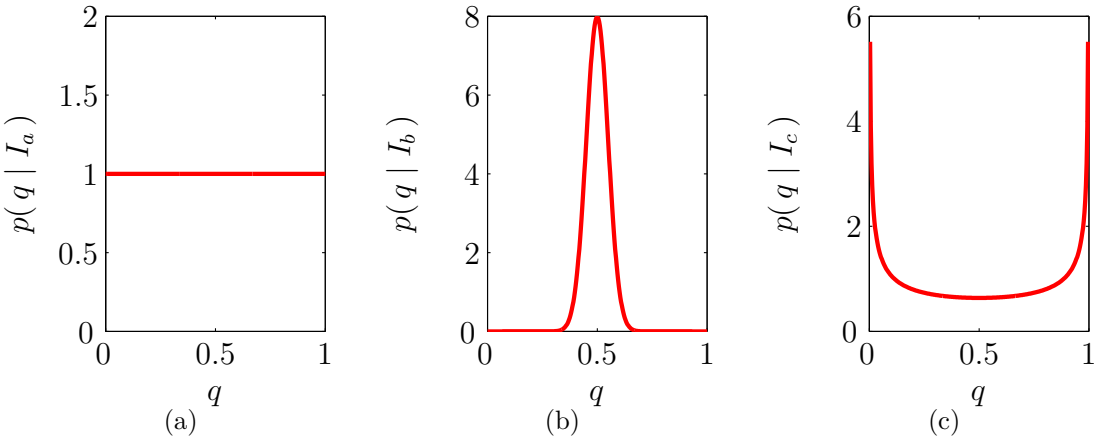


Figure 2.6: Second-order probability distributions for the Bernoulli urn problem, given the different states of knowledge:
(a): I_a = 'the urn contains two million balls, which may be either red or white.'
(b): I_b = 'the urn contains roughly the same number of red and white balls.'
(c): I_c = 'the urn contains either mostly white balls or mostly red balls.'

We can be a little more precise about the impact of new data. In particular, consider learning some piece of information K, whilst still being alert to the possibility of learning further information J in the future. Hence, for exactly the reasons

presented above, we may wish to proceed by introducing the $\mathring{q}$ of (2.5) to give

$$\begin{aligned} P(\, x \mid I,\, K\,) &= \int q(x)\, p(\, q \mid I,\, K\,)\, \mathrm{d}q \\ &= \frac{\int q(x)\, p(\, q \mid I\,)\; p(\, K \mid q,\, I\,)\, \mathrm{d}q}{\int p(\, q \mid I\,)\; p(\, K \mid q,\, I\,)\, \mathrm{d}q}\,, \end{aligned} \tag{2.5.2}$$

where by use of Bayes' rule we can update our prior distribution $p(\, q \mid I\,)$ to give the required posterior $p(\, q \mid I,\, K\,)$. This can then continue to be updated in this way whenever new information J is obtained.

We now return to the urn and consider repeatedly drawing from it with replacement, as depicted in Figure 2.7. K here represents the observations we make of N drawn balls. In considering repeated trials, we normally wish to assume that our knowledge I is identically pertinent to each trial; it does not discriminate amongst them. Importantly, we also take J to be similarly symmetric in the label of each trial; to represent knowledge that would affect our beliefs about the result of each trial equally. Finally, given both I and J, we also believe the trials to be independent. These conditions are satisfied in our example if J is taken to be the numbers of each type of ball in the urn. We have chosen a set of variables such that we can write

$$P(\, \{x_i;\, i = 1, \ldots, N\} \mid I,\, J\,) = \prod_{i=1}^{N} P(\, x_i \mid I,\, J\,)\,, \tag{2.5.3}$$

where $\mathring{x}_i$ is the colour of the ith draw and $p(\, \mathring{x}_i = \mathrm{Red} \mid I,\, J\,) = q$ is constant in i. Define the proposition that we obtain a particular sequence of exactly n red balls in N draws by

$$\mathring{\boldsymbol{x}}_l = \begin{cases} \mathring{x}_i = \mathrm{Red} & i = l_1, \ldots, l_n \\ \mathring{x}_i = \mathrm{White} & i = l_{n+1}, \ldots, l_N \end{cases}\,,$$

allowing us to express its probability as

$$P(\, \boldsymbol{x}_l \mid I,\, J\,) = q^n\, (1 - q)^{N-n}\,. \tag{2.5.4}$$

Note that the trials are not independent given I, our initial state of knowledge, alone. $P(\, x_i \mid I\,)$ is certainly constant in i; we are equally ignorant about each trial. However, $P(\, x_2 \mid I\,) \neq P(\, x_2 \mid I,\, x_1\,)$ – as we draw balls, we learn about the composition of the urn, giving us information relevant to the next draw. Including J,

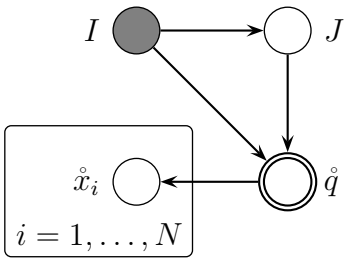


Figure 2.7: Bayesian network for repeated trials $\mathring{x}_i$.

assumed constant across trials, has allowed us to express the impact of such observations. This choice of representation has also given us the use of the convenient $\mathring{q}$, with the fortuitous independence structure of (2.5.4).

However, the trials are *exchangeable* given I alone. This means that our joint over trials $P(\,\mathring{\boldsymbol{x}}_l \mid I\,)$ is invariant with respect to permutations of the indices. Effectively, the order of the trials is unimportant; all that is relevant are the number of trials that returned red balls and the number that returned white. That this is true can be seen by writing, using (2.5.4),

$$\begin{aligned} P(\,\boldsymbol{x}_l \mid I\,) &= \iint P(\,\boldsymbol{x}_l \mid q\,)\, p(\,q \mid I,\, J\,)\, p(\,J \mid I\,)\,\mathrm{d}J\,\mathrm{d}q \\ &= \int q^n\,(1-q)^{N-n}\, p(\,q \mid I\,)\,\mathrm{d}q\,. \end{aligned} \tag{2.5.5}$$

Interestingly, a theorem of de Finetti [1937] (a proof of which can be found in Heath and Sudderth [1976]) states that for any infinite set of trials for which we possess an exchangeable belief distribution, there exists a variable $\mathring{q}$ (and associated prior $p(\,q \mid I\,)$) to fill exactly the same role as in (2.5.5). But recall that the existence of $\mathring{q}$ is nothing more than a rewriting of the presence of J, at least as far as $\mathring{x}$ is concerned. Hence de Finetti's theorem suggests that if our beliefs remain exchangeable even as we imagine taking an infinite number of trials, then there must always exist a set of variables J, which, if learnt, would render all trials independent with identical probability. This theorem underlines the applicability of our methods.

Of course, there are far more factors than simply J, the composition of the urn, that affect the colour of the ball that we draw. Our draw is uniquely determined by

a complete specification of how we reach into the urn and the exact position of each ball inside it. However, the experiment has been artificially designed such that these variables are not something we can ever imagine learning about. We are limited to observations of the colours of drawn balls, insufficient information to determine the myriad degrees of freedom of the interior of the urn. When marginalised out, these variables represent no bias towards either red or white – our knowledge of them is insufficient to favour one outcome over the other[9]. Given our unassailable ignorance of all such factors, the best we can hope for is to eliminate in this way the uncertainty due to J, leaving us with a probability of $\mathring{q}$.

If we were to somehow learn of some of these variables, our beliefs about the trials would no longer be exchangeable. Knowing the positions of all the balls at a specific point in time represents very pertinent information about the next draw, but much less information about the draw subsequent to that. Hence while we would lose the analytic niceties afforded by exchangeability and $\mathring{q}$, our resulting beliefs would be far more precise – giving better predictions as a result.

Now reconsider (2.5.2) in light of Figure 2.7. Define $K = \mathring{\boldsymbol{x}}_l$ and write

$$\begin{aligned} P(\, x \mid I,\, \boldsymbol{x}_l \,) &= \frac{1}{p(\, \boldsymbol{x}_l \mid I \,)} \iint p(\, J \mid I \,)\; P(\, x \mid q \,)\; P(\, \boldsymbol{x}_l \mid I,\, J \,)\; p(\, q \mid I,\, J \,) \,\mathrm{d}q \,\mathrm{d}J \\ &= \frac{\int q^{n+1} \,(1-q)^{N-n}\, p(\, q \mid I \,) \,\mathrm{d}q}{\int q^{n} \,(1-q)^{N-n}\, p(\, q \mid I \,) \,\mathrm{d}q} \,. \end{aligned} \tag{2.5.6}$$

Note that (2.5.6) would be identical if K had instead been simply an observation of the numbers of red and white balls – binomial coefficients would have cancelled from both numerator and denominator. As an example, we can investigate the diffuse $p(\, q \mid I \,) = 1$ depicted in Figure 2.6a, expressing total ignorance about the proportions of red and white balls. For this case, we employ the complete Beta function to obtain

$$P(\, x \mid I,\, \boldsymbol{x}_l \,) = \frac{n+1}{N+2} \,, \tag{2.5.7}$$

which is Laplace's rule of succession, discussed in depth in Jaynes [2003, Chapter 18].

[9] Indeed, even given perfect knowledge, the vast *computational* requirements of employing our knowledge would likely preclude accurate prediction.

The concept of 'physical' probabilities or 'propensities' is often met in dealings with repeated trials. The discussion above makes it clear why some may be tempted to adopt such notions – for exchangeable trials, we can indeed find a probability $\mathring{q}$ that applies equally to each trial. However, we stress again that this probability is purely and simply an expression of our beliefs assuming knowledge of J, the variables that act identically on each trial. Inference performed on $\mathring{q}$ itself is useful, so long as we are clear that it is no more than a placeholder for J. Our previous depiction of $\mathring{q}$ as being a physical object in a suitably informed robot's mind is often helpful as an *intuition pump* [Dennett, 1991] for how $\mathring{q}$ interacts with other variables. Indeed, an important use of $\mathring{q}$ emerges when we consider multi-agent systems, when our agents will need to possess beliefs about what is going on inside other agents' heads. But these probabilities $\mathring{q}$ need not be 'physical' for us to possess a belief about them. A probabilistic agent is entitled to a belief about *any* proposition. The probability of probability is a concept that has met with much opposition, as discussed by Goldsmith and Sahlin [1983]. We propose, however, that the concept is widely useful and intuitive – so long as we don't let our intuition confuse as to what we are really doing.

Chapter 3

Gaussian Processes

3.1 Introduction

Gaussian processes [Rasmussen and Williams, 2006] represent a powerful way to perform Bayesian inference about functions. We begin by considering functions as vectors, a long list of all possible function outputs that is associated with a similarly long list of all possible function inputs. For most useful functions, however, the number of possible inputs is infinite – we typically want to consider functions over the real numbers. To manage this potential difficulty, we define a Gaussian process (GP) as being a probability distribution over a (possibly infinite) number of variables, such that the distribution over any finite subset of them is a multi-variate Gaussian. We are now free to use a GP as a prior distribution for a function.

This choice provides a truly remarkable degree of flexibility, as we'll see later. However, our choice is undeniably informed by the remarkably expedient properties of the Gaussian distribution (Appendix A.1). In particular, the marginals of a multivariate Gaussian distribution are themselves Gaussian. Similarly, the distribution of any subset of variables conditioned upon any other subset is also Gaussian. These properties, illustrated in Figures 3.1 and 3.2, allow us to readily consider subsets of the potentially infinite lists of function outputs and inputs. The first property allows us to trivially marginalise over any unobserved values of our function, even in the common case in which there are an infinite number of such values. The fact that such marginals are Gaussian, along with the second property, allows inference that is

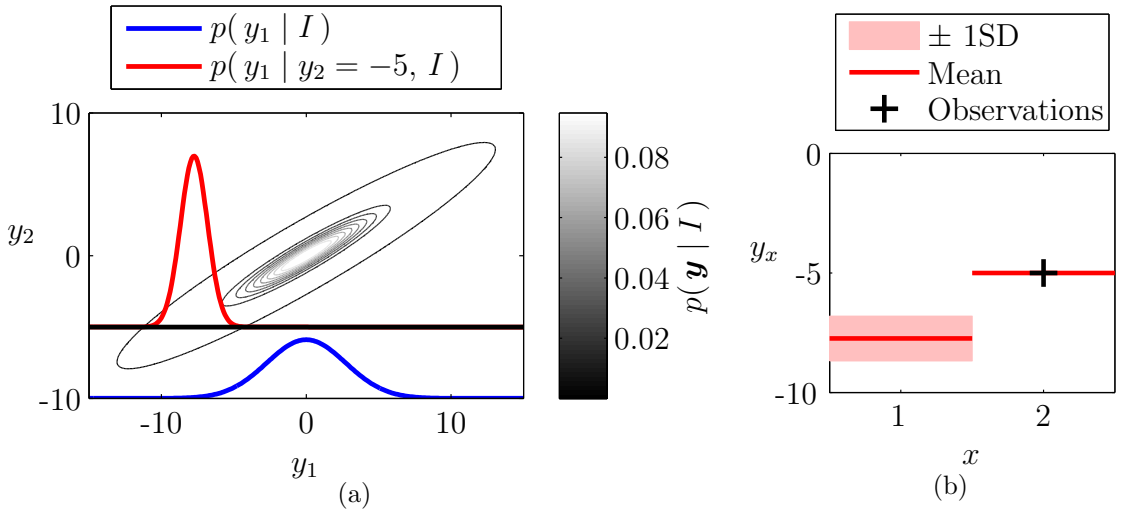


Figure 3.1: (a) Given a bivariate Gaussian for $p(\,\boldsymbol{y} \mid I\,)$ ($\boldsymbol{y} \triangleq [y_1, y_2]$), an illustration that both the marginal distribution $p(\,y_1 \mid I\,)$ and conditional distribution $p(\,y_1 \mid y_2 = -5,\, I\,)$ are themselves Gaussian. Such distributions can be readily determined using the identities in Appendix A.1. Note that the depicted bivariate Gaussian expresses that y_1 and y_2 are strongly correlated, as indicated by its narrow, diagonal shape. As a consequence, in learning y_2, we gain information about y_1 – $p(\,y_1 \mid y_2 = -5,\, I\,)$ is more peaked (and hence informative) than the diffuse $p(\,y_1 \mid I\,)$. (b) gives a different representation of $p(\,\boldsymbol{y} \mid y_2 = -5,\, I\,)$, plotting for each of y_1 and y_2 the means and the regions enclosed by the means plus or minus a single standard deviation. Figure 3.2 displays the generalisation of (b) to a greater number of variables.

conveniently constrained to the space of Gaussians.

We consider a function $y(x)$. As we are uncertain about the actual values our function takes, we must treat them as we would any other 'random variable'. We consider propositions such as $\mathring{y}(x) = y(x)$ – the actual function at x assumes the possible value $y(x)$. We are principally concerned with prediction and hence functions in time, but in all that follows x may equally be interpreted as any arbitrary kind of input. We'll use $\boldsymbol{y}$ to refer to a possible vector of function outputs and $\boldsymbol{x}$, function inputs. I, known as the *context*, will be used to represent the models and background knowledge that we employ in our analysis. Function inputs, like $\boldsymbol{x}$, are almost always known, and hence will usually be incorporated into I for notational convenience. A Bayesian network depicting the possible correlations amongst our variables is depicted

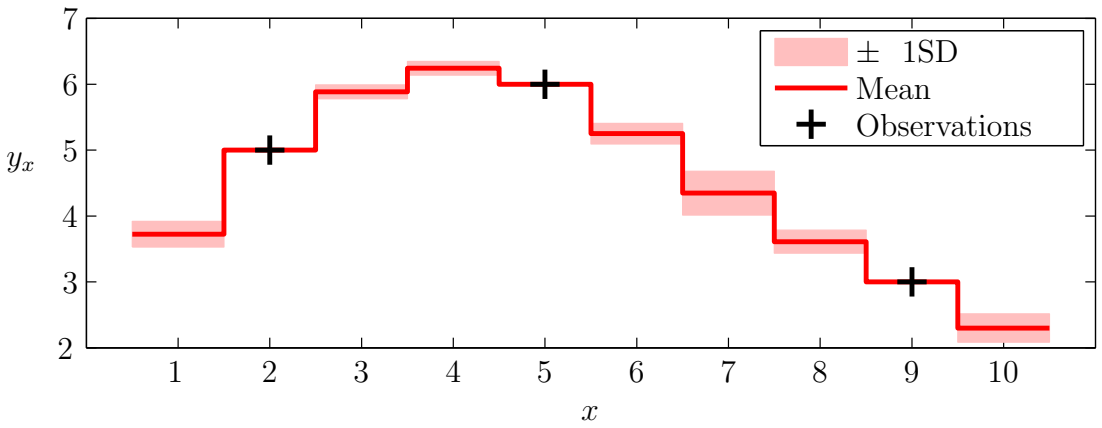


Figure 3.2: An example of the distributions (represented by means and SDs) for y_1 through y_{10} conditional on observations of y_2, y_5 and y_9, assuming a multivariate Gaussian prior distribution over y_1 through y_{10}. A Gaussian process is the generalisation of such a distribution that gives a multivariate Gaussian prior for any set of values $\boldsymbol{y}$.

in Figure 3.3. Hence we define a GP as the distribution

$$p(\,\boldsymbol{y}\mid\boldsymbol{\mu},\,\mathbf{K},\,I\,)\triangleq\mathcal{N}(\boldsymbol{y};\boldsymbol{\mu},\mathbf{K}) \tag{3.1.1}$$

which by the properties of a Gaussian will hold true for some mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$ regardless of the subset represented by $\boldsymbol{y}$ and $\boldsymbol{x}$. Clearly it is the choice of the mean and covariance that defines a GP [MacKay, 2006]; the two respectively defining location and scale in a way identical to any more familiar Gaussian distribution. Where clear, knowledge of both mean and covariance will be included within our context I.

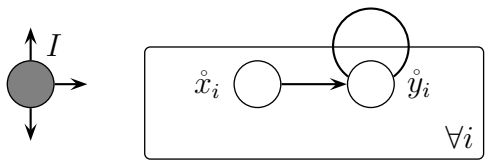


Figure 3.3: Bayesian network for a generic GP – note the value of the function $y(\cdot)$ at a particular input may potentially be related to its value for any other input.

3.2 Mean and Covariance Functions

3.2.1 Hyperparameters

The mean $\boldsymbol{\mu}$ here expresses our expectation about the function values $\boldsymbol{y}$ before having made any observations. This mean vector is generated by a *mean function* $\mu(x;\phi)$, capable of expressing our belief about $y(x)$ for any x. Likewise, the covariance matrix $\mathbf{K}$ must be generated by a *covariance function* $K(x_1, x_2;\phi)$. $\mathbf{K}$ is a square matrix indicating the strength of correlations amongst the entries of $\boldsymbol{y}$. Hence it must have an appropriate entry for every possible pair of admissable inputs x_1 and x_2.

Both mean and covariance functions are specified by a set of *hyperparameters* (that is, parameters of the GP prior), which we collectively denote as ϕ. Marginalising these hyperparameters, effectively nuisance variables, forms the essence of the challenge of GP prediction. We'll return to this topic in Section 3.4. Where we need not perform inference about hyperparameters, we will drop them, as per $\mu(x)$ and $K(x_1, x_2)$. Where we are simultaneously performing inference about multiple functions, we will specify the different means and covariances used with superscripts. For example, the covariance function used to model $y(x)$ will be denoted as $K^{(y)}$, and the mean as $\mu^{(y)}$. We will use the same convention should we wish to distinguish between covariance functions or means for any other reason.

3.2.2 Mean functions

A popular, simple choice for our mean is to take $\mu(x;\phi) = 0$, stating that our initial best guess for the function output at any input is zero. Of course, if we possess better knowledge than this assignment, it should be included. Another possibility is to take $\mu(x;\phi)$ as a non-zero constant for all inputs, where this constant forms a hyperparameter about which we will also possess prior beliefs. Of course, in general, we might take whatever arbitrarily complicated $\mu(x;\phi)$ we require to express our beliefs about $y(x)$. Popular non-constant mean functions include polynomials of various orders.

3.2.3 Covariance functions

The difficulty with parameterising a covariance function is due to the requirement that $\mathbf{K}$ be positive semi-definite. Regardless, many choices exist [Abrahamsen, 1997, Gibbs, 1997, Gneiting, 2002], allowing us to express a wide range of assumptions about our function[1]. A very common expectation we possess about our function is that it be smooth to some extent. That is, the value of a function at x is strongly correlated with the values close to x, these correlations becoming weaker further away. In this case, the covariance will be of the *homogenous* form

$$K(x_1, x_2; h, w) \triangleq h^2 \kappa\big(d(x_1, x_2; w)\big)\,, \tag{3.2.1}$$

where κ is an appropriately chosen function that decreases with increasing d, a metric parameterised by w. For one dimensional inputs x, we will often use

$$d^{(\text{simple})}(x_1, x_2; w) \triangleq \left| \frac{x_1 - x_2}{w} \right| . \tag{3.2.2}$$

In such a context, $h > 0$ and $w > 0$, elements of ϕ, specify the expected length scales of the function in output ('height') and input ('width') spaces respectively. The larger the value of h, the larger the dynamic range we would expect from our data; the larger the value of w, the more rapidly we would expect it to vary with changes in input.

The distance (3.2.2) embodies a further common assumption, that of *stationarity*. A stationary covariance function takes the correlation between points to be purely a function of the difference in their inputs, $x_1 - x_2$. This effectively asserts that our function looks more or less similar throughout its domain. Similarly, we will often want our function to be isotropic, such that it does not have a preferred direction. This quality is expressed by the absolute value used in (3.2.2).

Examples of covariance functions of the form (3.2.1) are illustrated in Figure 3.4. The prototypical stationary covariance function is the squared exponential

$$K^{(\text{SE})}(x_1, x_2; h, w) \triangleq h^2 \exp\left(-\frac{1}{2} d(x_1, x_2; w)^2 \right) , \tag{3.2.3}$$

[1]Kalman Filters, for example, represent another, very special, type of covariance function, as described in Appendix C.

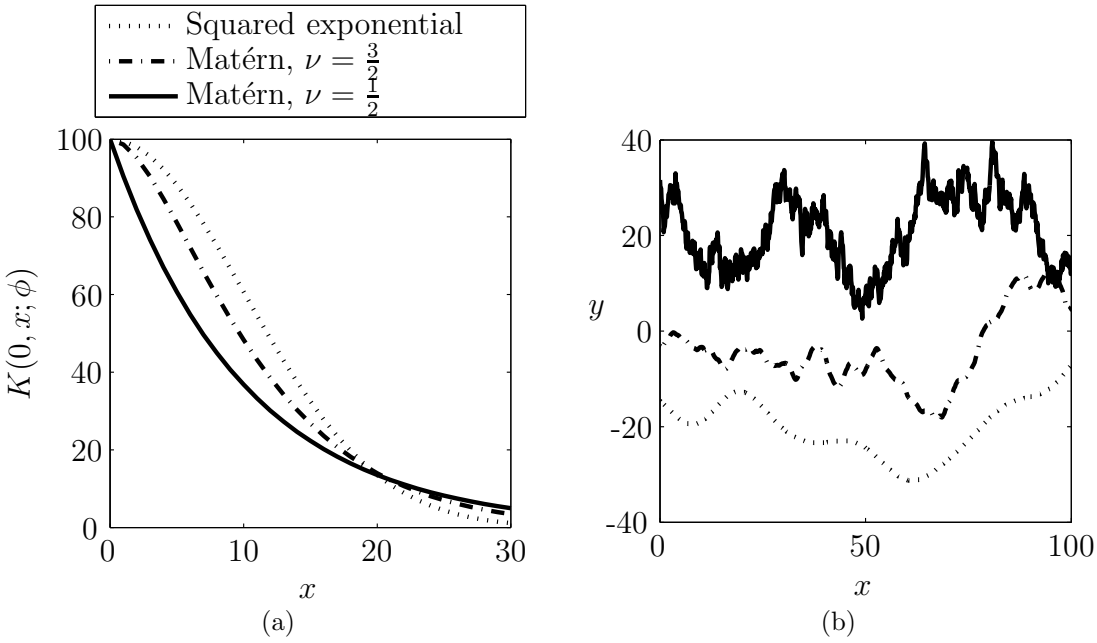


Figure 3.4: For squared exponential (3.2.3) and Matérn (3.2.4) covariance functions, (a) the covariance as a function of separation and (b) example data they might be appropriate for. Note the strong correlations associated with the smoother functions. For all cases, $w = 10$ and $h = 10$.

useful for modelling particularly smooth functions.

However, it has been argued [Stein, 1999] that the strong smoothness that (3.2.3) enforces is an inappropriate assumption for many physical processes. A more flexible alternative is the Matérn class of covariance functions, given by

$$K^{(\mathrm{Mtn})}(x_1, x_2; h, w, \nu) \triangleq h^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\, d(x_1, x_2; w) \right)^{\nu} \mathfrak{K}_{\nu} \left(\sqrt{2\nu}\, d(x_1, x_2; w) \right) \quad (3.2.4)$$

where $\nu > 0$ is a smoothness hyperparameter (larger ν implies smoother functions) and $\mathfrak{K}_{\nu}$ is the modified Bessel function. Fortunately, (3.2.4) simplifies for half-integer ν, to give, for the example of $\nu = 3/2$

$$K^{(\mathrm{Mtn})}(x_1, x_2; h, w, 3/2) = h^2 \left(1 + \sqrt{3}\, d(x_1, x_2; w) \right) \exp\left(-\sqrt{3}\, d(x_1, x_2; w) \right) \quad (3.2.5)$$

A final example of a covariance of the form (3.2.1) is the rational quadratic, parameterised by α,

$$K^{(\mathrm{RQ})}(x_1, x_2; h, w, \alpha) \triangleq h^2 \left(1 + \frac{1}{2\alpha} d(x_1, x_2; w)^2 \right)^{-\alpha}, \quad (3.2.6)$$

which can be seen as an infinite sum of squared exponential covariances with different input scales [Rasmussen and Williams, 2006].

3.3 Posterior Distributions

Given a set of hyperparameters ϕ, then, we can evaluate a covariance $K(x_1, x_2; \phi)$ and mean $\mu(x; \phi)$. Now consider additionally knowing the predictor data, $\boldsymbol{y}_d$ at $\boldsymbol{x}_d$, and being interested in the values of the predictants $\boldsymbol{y}_\star$ at known $\boldsymbol{x}_\star$. As mentioned earlier, we will usually assume that the relevant inputs, such as $\boldsymbol{x}_d$ and $\boldsymbol{x}_\star$, are known to us and as such will assimilate them into I. Using (3.1.1) and the properties of the Gaussian distribution (Appendix A.1), we can write our posterior distribution as

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{y}_d,\, \phi,\, I\,) = \mathcal{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I)\,, \mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I)\big)\,, \tag{3.3.1}$$

where

$$\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I) = \boldsymbol{\mu}(\boldsymbol{x}_\star; \phi) + \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\, \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1} \big(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d; \phi)\big) \tag{3.3.2}$$

$$\mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I) = \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star; \phi) - \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\, \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1}\, \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_\star; \phi)\,, \tag{3.3.3}$$

we also define the more general covariance between two sets of predictants $\boldsymbol{y}_\star$ and $\boldsymbol{y}_*$

$$\mathbf{C}(\mathring{\boldsymbol{y}}_\star,\, \mathring{\boldsymbol{y}}_* \,|\, \boldsymbol{y}_d,\, \phi,\, I) = \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_*; \phi) - \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\, \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1}\, \mathbf{K}_\phi(\boldsymbol{x}_d, \boldsymbol{x}_*)\,. \tag{3.3.4}$$

Notice that, as we discussed in Section 3.1, the choice of a Gaussian process over $y(x)$ has given us a posterior distribution that is conveniently itself a Gaussian. This means, for example, that we can readily update our posterior distribution should we obtain yet further data; that new posterior will be simply another Gaussian. Note, too, that the remarkable marginalisation property of the Gaussian means that this posterior distribution can be obtained despite the fact that there are an infinite number of values of y that are neither predictors nor predictants.

We will also make use of the condensed notation

$$\boldsymbol{m}^{(y)}_{\star|d,\phi} \triangleq \boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I)$$
$$\mathbf{C}^{(y)}_{\star|d,\phi} \triangleq \mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{y}_d,\, \phi,\, I)\,.$$

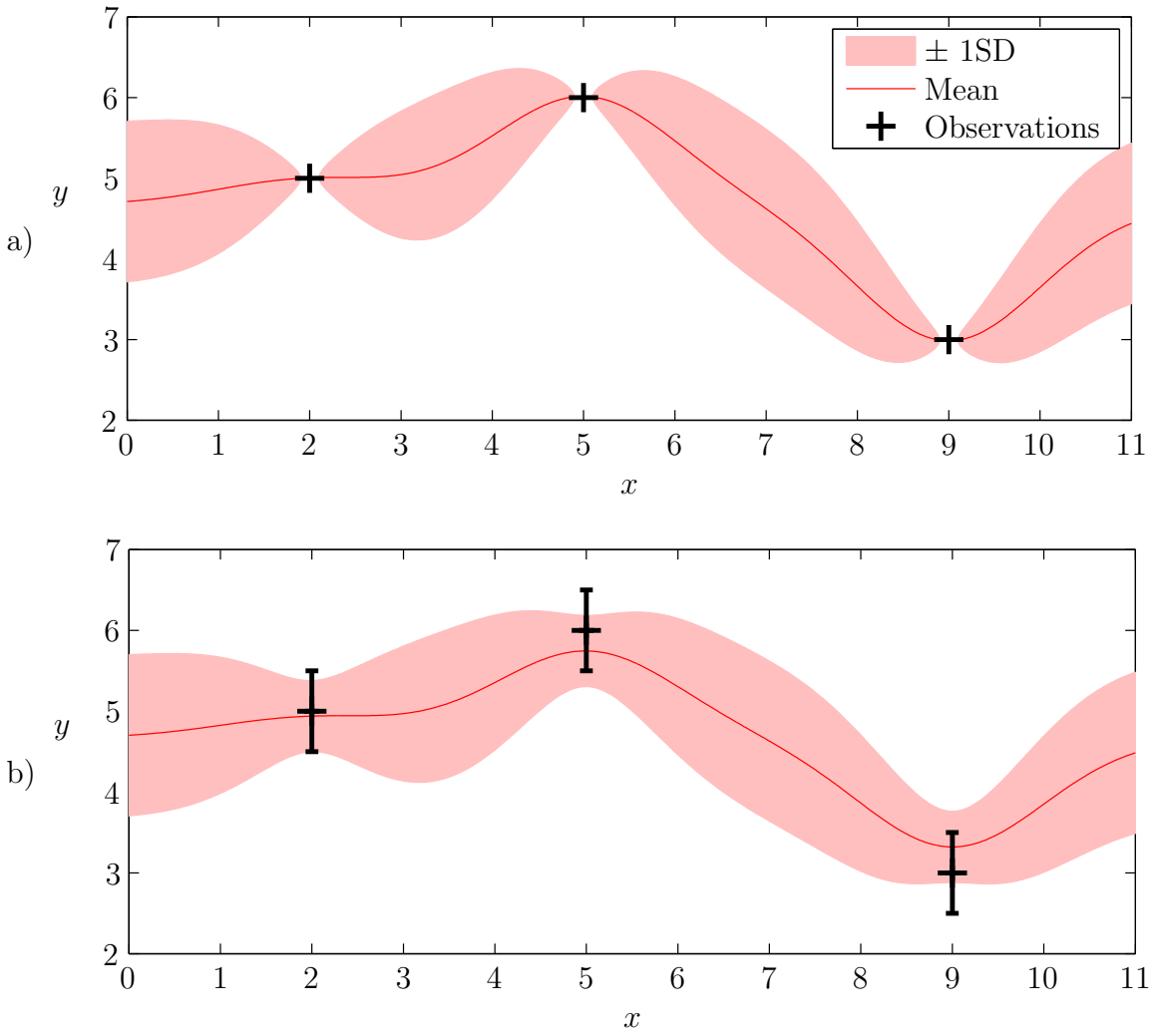


Figure 3.5: GP regression for a continuous function $y(x)$ (compare with the discrete case in Figure 3.2). For this and for all similar figures henceforth, we plot the means (from (3.3.7)) and the regions enclosed by the means plus or minus a single standard deviation (deduced from (3.3.8)) for a large number of y variables. As such, the means and one standard error boundaries appear to be continuous functions. We take a zero mean function and the squared exponential covariance function (3.2.3) with $w = 1$, $h = 1$. For the noise SD, we take (a) $\sigma = 0$ (b) $\sigma = 0.5$. Note that the predictive variance is small close to observations, but grows larger elsewhere.

Where we consider inference about only a single function, we will further abbreviate by dropping the superscript to give $\boldsymbol{m}_{\star|d,\phi}$ and $\mathbf{C}_{\star|d,\phi}$. Where we do not consider inference over hyperparameters, the relevant subscript may be dropped, as per $\boldsymbol{m}^{(y)}_{\star|d}$ and $\mathbf{C}^{(y)}_{\star|d}$.

Equation (3.3.1) forms the cornerstone of GP inference – it permits us to update our beliefs about the predictants $\boldsymbol{y}_\star$, after having obtained data $\boldsymbol{y}_d$. In using GPs for regression, $\boldsymbol{x}_\star$ will typically lie scattered in between the set $\boldsymbol{x}_d$. For prediction in time, we typically use data from past times $\boldsymbol{x}_d$ to make predictions about future times $\boldsymbol{x}_\star$. In particular, we are often interested in performing sequential prediction. At any time x, we make predictions about $y_\star$ at the time $x_\star \triangleq x + \epsilon$. Here ϵ is the *lookahead*, which for simple tracking is zero. When more data is obtained, we update our GP and perform inference about the new lookahead-shifted time. We will discuss practical methods of performing such sequential prediction in Chapter 5.

In most real applications, though, we observe not the underlying variable of interest y, but only some noise-corrupted version of it, z. GPs are unsurprisingly readily extended to allow for Gaussian noise models[2], in which we assume a Gaussian likelihood

$$p(\, \boldsymbol{z} \mid \boldsymbol{y},\, \phi,\, I\,) = \mathcal{N}\big(\boldsymbol{z}; \boldsymbol{y}, \sigma^2 \mathbf{I}\big)\ , \tag{3.3.5}$$

where $\mathbf{I}$ is an identity matrix of appropriate size. That is, we assume independent Gaussian noise contributions of a fixed variance σ^2. This noise variance effectively becomes another hyperparameter of our model and as such will be incorporated into ϕ. To proceed, we define

$$V(x_1, x_2; \phi) \triangleq K(x_1, x_2; \phi) + \sigma^2\, \delta(x_1 - x_2)$$

where $\delta(-)$ is the Kronecker delta function. Hence, by using the properties of the Gaussian distribution (Appendix A.1) once again, we have

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi,\, I\,) = \mathcal{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I)\,, \mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I)\big)\,, \tag{3.3.6}$$

where

$$\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) = \boldsymbol{\mu}(\boldsymbol{x}_\star; \phi) + \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\, \mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1} \big(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d; \phi)\big) \tag{3.3.7}$$

$$\mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) = \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star; \phi) - \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\, \mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1}\, \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_\star; \phi)\,. \tag{3.3.8}$$

[2]Other noise models are possible, at the cost of the loss of the analytic niceties associated with Gaussianity. The study of other noise models forms an active area of research, see Rasmussen and Williams [2006]. We consider one such alternative noise model in Section 4.6.

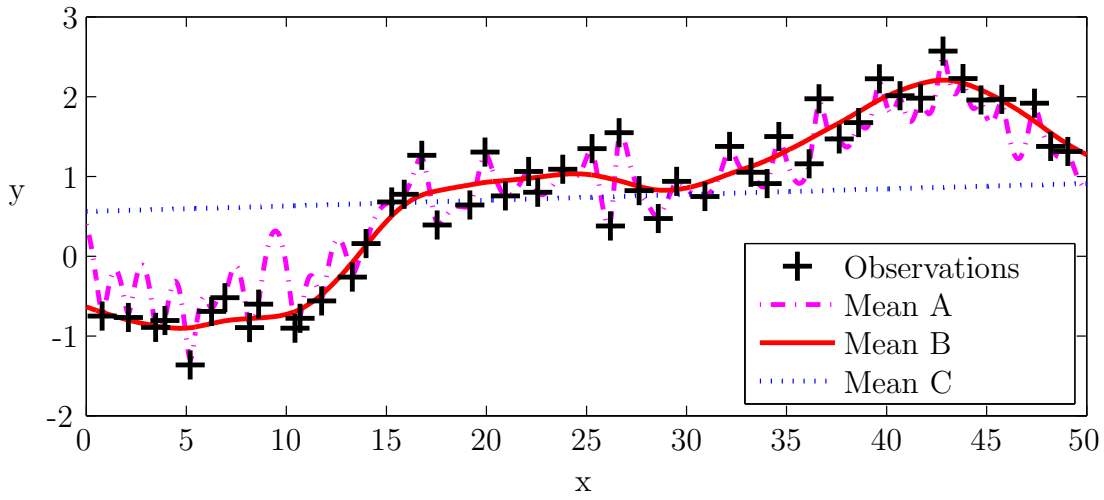


Figure 3.6: Regression using a squared exponential covariance with output scale $h = 1$, unknown input scale w and assuming noisy observations of unknown SD σ. Both h and σ must hence be simultaneously marginalised; we take a broad prior for σ and consider three different priors for w. Mean A depicts the predictive mean given a prior that favours a very small w, in which case the posterior favours a small σ. This represents a description of the data as being all but noiseless, all variation being caused by real fluctuations in the underlying y. Mean C depicts the predictive mean given a prior that favours a very large w, in which case the posterior favours a large σ, essentially explaining all variation of the data as being due to noise. Mean B is the predictive mean given a broad prior for w.

An example of using these equations for inference is demonstrated in Figure 3.5.

It is also possible, if desired, to make predictions about the noise-corrupted variables themselves, as per

$$\boldsymbol{m}(\mathring{\boldsymbol{z}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) = \boldsymbol{m}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) \tag{3.3.9}$$

$$\mathbf{C}(\mathring{\boldsymbol{z}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) = \mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I) + \sigma^2\, \mathbf{I}_\star\,. \tag{3.3.10}$$

3.4 Marginalising Hyperparameters

We now return to the problem of marginalising our hyperparameters. An example of this challenge is illustrated in Figure 3.6. Firstly, we need to assign appropriate priors $p(\,\phi \mid I\,)$. Given these, the expression we need to evaluate is:

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, I\,) = \frac{\int p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi_*,\, I\,)\; p(\,\boldsymbol{z}_d \mid \phi_*,\, I\,)\; p(\,\phi_* \mid I\,)\; \mathrm{d}\phi_*}{\int p(\,\boldsymbol{z}_d \mid \phi_*,\, I\,)\; p(\,\phi_* \mid I\,)\; \mathrm{d}\phi_*} \tag{3.4.1}$$

(3.4.1) performs a great deal of work for us; it is worth examining its powers in more detail. In particular, we have not yet addressed a method for choosing the level of complexity for our model for the mean or covariance. For the mean, should we take a constant, linear or quadratic model? For the covariance, should we use the simple squared exponential (3.2.3) or allow for the additional flexibility afforded by the smoothness parameter of the Matérn class (3.2.4)?

Clearly, complex models include many practical simpler models as subsets. For example, a general quadratic mean function

$$\mu(x; a, b, c) \triangleq a\,x^2 + b\,x + c$$

can express any possible linear or constant model. It is in selecting the right hyperparameters that Bayesian marginalisation demonstrates its worth. If a term, such as the x^2 term in our quadratic mean, is actively unrepresentative, the likelihood $p(\,\boldsymbol{z}_d \mid \phi,\, I\,)$ will be higher for hyperparameter values that eliminate the misleading term, as per $a = 0$. The likelihood will effectively screen this term from making any contributions to (3.4.1).

Typically, however, we will have a range of possible models that fit the data more or less equally. This would not be a problem, except that they tend to produce different predictions. For such multiple possible models, the argument of William of Occam suggests that we should prefer the simplest. It's common that we might possess a prior that favours the simpler model over a more complex one. However, the remarkable action of Bayesian inference is to penalise an unnecessarily complex model, even if our priors are flat [MacKay, 2002].

In explanation, consider that we wish to compare a complex model C and a simple model S. By this, we mean that the hyperparameter space allowable under C is greater than that allowable under S. For notational covenience we'll define a model variable M that may take either the value C or S. As we're only interested in comparing these two models against each other, we take them to be exhaustive.

Moreover, we'll take the flat prior $p(\, M \mid I \,) = \frac{1}{2}$. With this, we can rewrite (3.4.1) as

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, I \,) = \frac{p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, C,\, I \,)\; p(\, \boldsymbol{z}_d \mid C,\, I \,) + p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, S,\, I \,)\; p(\, \boldsymbol{z}_d \mid S,\, I \,)}{p(\, \boldsymbol{z}_d \mid C,\, I \,) + p(\, \boldsymbol{z}_d \mid S,\, I \,)} \tag{3.4.2}$$

Hence the relevant terms here are the evidences $p(\, \boldsymbol{z}_d \mid M,\, I \,)$, whose importance is stressed by Skilling [2006]. These form relative weights for the predictions made under the two models. As such, we're interested in the ratio of their magnitudes

$$\frac{p(\, \boldsymbol{z}_d \mid C,\, I \,)}{p(\, \boldsymbol{z}_d \mid S,\, I \,)} = \frac{\int p(\, \boldsymbol{z}_d \mid \phi,\, C,\, I \,)\; p(\, \phi \mid C,\, I \,)\; \mathrm{d}\phi}{\int p(\, \boldsymbol{z}_d \mid \phi,\, S,\, I \,)\; p(\, \phi \mid S,\, I \,)\; \mathrm{d}\phi} \tag{3.4.3}$$

We take both models to fit the data equally well. That is, there are 'best-fit' configurations of hyperparameters ϕ_M for M such that

$$p(\, \boldsymbol{z}_d \mid \phi_C,\, C,\, I \,) = p(\, \boldsymbol{z}_d \mid \phi_S,\, S,\, I \,) \tag{3.4.4}$$

To get the flavour of how the inference would proceed, we'll take the Laplace approximation for the integrals in (3.4.3). Hence we assume that both ϕ_M represent the respective solitary peaks of the integrands $p(\, \boldsymbol{z}_d \mid \phi,\, M \,)\; p(\, \phi \mid M \,)$. These integrands can also be written as $p(\, \phi \mid \boldsymbol{z}_d,\, M \,)\; p(\, \boldsymbol{z}_d \mid M \,)$ – when integrating over ϕ, the second term is simply a multiplicative constant. Any width of the integrand around its peak is entirely due to $p(\, \phi \mid \boldsymbol{z}_d,\, M \,)$. We take measures of these widths as $\Delta_{\phi|d,M}$; such a width would be simply equal to $\sqrt{\det 2\pi K}$ if $p(\, \phi \mid \boldsymbol{z}_d,\, M \,)$ were Gaussian with covariance K. Hence we can approximate our integrals as being simply the height times the width of the integrands

$$\begin{aligned} \frac{p(\, \boldsymbol{z}_d \mid C,\, I \,)}{p(\, \boldsymbol{z}_d \mid S,\, I \,)} &\simeq \frac{p(\, \boldsymbol{z}_d \mid \phi_C,\, C,\, I \,)\; p(\, \phi_C \mid C,\, I \,)\; \Delta_{\phi|d,C}}{p(\, \boldsymbol{z}_d \mid \phi_S,\, S,\, I \,)\; p(\, \phi_S \mid S,\, I \,)\; \Delta_{\phi|d,S}} \\ &= \frac{p(\, \phi_C \mid C,\, I \,)\; \Delta_{\phi|d,C}}{p(\, \phi_S \mid S,\, I \,)\; \Delta_{\phi|d,S}} \end{aligned} \tag{3.4.5}$$

MacKay [2002] calls the terms $p(\, \phi_M \mid M,\, I \,)\; \Delta_{\phi|d,M}$ *Occam factors*. For an insight into their meaning, we now further assume that the priors $p(\, \phi \mid M,\, I \,)$ are roughly constant over some width $\Delta_{\phi|M}$ and zero elsewhere. Hence

$$\frac{p(\, \boldsymbol{z}_d \mid C,\, I \,)}{p(\, \boldsymbol{z}_d \mid S,\, I \,)} \simeq \frac{\Delta_{\phi|d,C}}{\Delta_{\phi|C}} \Big/ \frac{\Delta_{\phi|d,S}}{\Delta_{\phi|S}} \tag{3.4.6}$$

The width $\Delta_{\phi|d,M}$ is a measure of the sensitivity and flexibility of the model; it represents the volume of hyperparameters consistent with the obtained data. This width is small if the model must be very finely tuned in order to match the data, in which case it will be penalised by the Occam factor. The width $\Delta_{\phi|M}$ is a measure of the complexity of the model; it represents the a priori volume of hyperparameters associated with the model. This width is large if the model has a great number of hyperparameters, in which case it will be penalised by the Occam factor.

Hence we would expect both

$$\Delta_{\phi|d,C} > \Delta_{\phi|d,S} \tag{3.4.7a}$$

$$\Delta_{\phi|C} > \Delta_{\phi|S} \tag{3.4.7b}$$

Hence the Occam factor gives an implicit handicap (3.4.7b) to the complex model, but, if it can justify its flexibility for the received data, it is rewarded (3.4.7a). Effectively, a model will be punished for any 'wasted' hyperparameter space [Gregory, 2005]. The models selected by the marginalisation procedure will have no hyperparameters that are not required to match the data.

The conclusion to be drawn from this is that if we specify a complex model, the machinery inside our Bayesian inference engine will automatically select the simplest sub-model that is sufficiently consistent with the data. In this sense, there is no harm in including as complex and flexible a model as we can dream up – probability theory will do the work for us.

However, the powers of (3.4.1) come at a cost – the integrals are very hard to evaluate! For our Gaussian process framework, as demonstrated by (3.3.6), both $p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi,\, I\,)$ and $p(\,\boldsymbol{z}_d \mid \phi,\, I\,)$ have non-trivial dependence upon ϕ. ϕ is used to generate mean vectors and covariance matrices which are variously inverted, multiplied together and summed, rendering our integrals profoundly nonanalytic. As a consequence, we are forced to use techniques of numerical integration. We'll discuss this at much greater length later in Chapters 7 and 8, but for now it is sufficient to state that we will approximate (3.4.1) by averaging over a set of hyperparameter

samples $\{\phi_i : i \in s\}$. That is, we evaluate our predictions $p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi_i,\, I\,)$ and likelihoods $p(\, \boldsymbol{z}_d \mid \phi_i,\, I\,)$ for each hyperparameter sample ϕ_i. The likelihoods are then used to determine weights $\{\rho_i : i \in s\}$, allowing us to approximate (3.4.1) as a weighted mixture

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, I\,) \simeq \sum_{i \in s} \rho_i \, p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi_i,\, I\,) \;. \tag{3.4.8}$$

The mean and covariance of such a Gaussian mixture are readily determined, as per (A.1.7) and (A.1.8) in Appendix A.1.

A particularly severe approximation is that of *maximum likelihood*. If the likelihood $p(\, \boldsymbol{z}_d \mid \phi,\, I\,)$ is maximised at $\phi_m(\boldsymbol{z}_d)$, maximum likelihood assumes

$$p(\, \boldsymbol{z}_d \mid \phi,\, I\,) \simeq \delta(\phi - \phi_m(\boldsymbol{z}_d)) \;.$$

Similarly, *maximum a posteriori* methods assume the posterior $p(\, \phi \mid \boldsymbol{z}_d,\, I\,)$ is a delta function centered at its maximum. In reality, of course, it is not rare for these quantities to have significant probability 'mass' away from their peaks, as discussed in Section 2.1. Hence both methods are prone to problematic features [MacKay, 2002]. However, note that under both approaches, we would be led to an approximation of the form (3.4.8) with only a single non-zero element in ρ. This single element approximation is a very useful simplification that we will make occasional use of. It finds some justification when we have large amounts of data, such that $p(\, \boldsymbol{z}_d \mid \phi,\, I\,)$ is strongly peaked.

A slightly more sophisticated approach is to take a *Laplace approximation* [MacKay, 2002], which fits a Gaussian around such peaks, resulting in the placement of probability mass away from (although still around) them. Yet further sophistication is displayed by the methods of *Variational Bayes* [Bishop et al., 2006], or *Expectation Propagation* [Minka, 2001], which treat the fitting of probability distributions to the problematic terms in our integrand as an optimisation problem. We do not consider these powerful techniques further, however, due to questions over the relative merits of the objective functions to be used in that process of optimisation. We limit

ourselves to quadrature, in which samples of the integrand are used to construct the integral.

Unfortunately, it is well-known that the difficulty of quadrature (along with the maximisation required for maximum likelihood or *a posteriori* approaches) grows drastically with the dimension of the integral (the 'curse of dimensionality'). In our case, this dimension is equal to the number of hyperparameters to be marginalised. As such, this provides a practical limit on the number and complexity of terms we can include in our mean and covariance functions. Unless we have reason to believe a term is necessary, it should be left out. We should nonetheless remain alert to the possibility that the space spanned by our current model is insufficiently broad to include the studied phenomenon, that our model is just too simple to capture the real dynamics.

The quantity that will act as our meter in this regard is again the evidence, $p(\boldsymbol{z}_d \mid I)$. I here includes the assumptions that underly our selection of a covariance model – given that assigning a prior over models is notoriously difficult and subjective, the model likelihood, represented by the evidence, forms the most informative measure of the efficacy of our model. For exactly the same reasons as discussed above, the evidence will always favour a simpler model if the data is insufficiently discriminatory. If the evidence is small relative to another, we should reconsider our choice of model, its mean and covariance structure. Of course, the evidence forms the denominator of (3.4.1) and hence is a quantity we will necessarily be computing in any case. Skilling [2006] also recommends the publication of the evidence of our final chosen model. This allows any other investigators to objectively compare their methods and models to our own.

Of course, a GP is simply an epistemic probability. Probability theory will ensure that our final results are correct given the specified conditioning information. One consequence of this is that if a (correctly) calculated probability does not reflect our personal beliefs, then only one of two things may be true. Firstly, it is possible that our personal beliefs about the subject are logically inconsistent – our expectations

for the calculation were poorly thought out. Otherwise, it must be the case that the model M we fitted our calculation with was not truly representative of our true beliefs. In principle, if we are honestly at all uncertain about M, it should be marginalised. This would entail specifying a prior distribution and then integrating over the space of all possible models – clearly a difficult task. Hence, in practice, we are usually forced to assume a model to work with.

So, to summarise, if a probability seems wrong, either our inputs to or expectations about the output of the inference engine must be wrong. In either case, we have learned something new! This is the essence of how science progresses [Jaynes, 2003] – we trial different assumptions until we find something that seems to fit, even if the resulting theory's predictions appear initially surprising. As John Skilling points out, this is as good as we can ever possibly do. We can never test any theory for all the infinite possible cases to which it should apply, perform every experiment in every possible place and time, to prove it was absolutely 'true'. If we had the truth sitting in front of us, how could we even possibly tell? All we can do is determine the theory that best fits what we've seen so far, and continue to revise that theory as new evidence arrives. This is exactly what is done for us by probability theory.

Hence, if after evaluating the mean and covariance of our GP they do not match our expectations, only one of two things may be possible. Either our expectations or the model we gave the GP were ill-founded. We are quite within our rights to go back and change our prior distribution (in the form of the mean and covariance of our GP) until we have results that do match our intuition. Equally, it is possible the GP has revealed a surprising but necessary consequence of our initial information.

3.5 Final remark

All the work to follow is built upon the power of GPs. We will, in particular, spend the entirety of Chapter 4 developing methods demonstrating the truly remarkable flexibility of GP models. However, it behooves us to acknowledge that the assumption

of a GP is, nonetheless, still an assumption. To illustrate this, the mean estimate of a GP for a predictant will always be a linear combination of observed predictants, as per (3.3.2). However, it's quite possible for us to possess beliefs about a system such that our mean estimate for a predictant is some non-linear combination of the predictors. Similarly, information of the form 'this function is a pdf', or 'this function is convex', is not readily incorporated into the GP framework. Our algorithm could still be made to perform inference about such functions, but its performance would likely be poorer than another algorithm incorporating our true prior knowledge. In summary, we must always acknowledge our prior information, including the inevitable limits imposed by our models.

Chapter 4

Extending Gaussian Processes

4.1 Modifying Covariance Functions

We now return to the topic of constructing covariance functions appropriate for problems of various types. In particular, we will take the simple covariances presented in Section 3.2.3 as building blocks, and consider various ways to construct valid new covariance functions by modifying and combining them. Such modifications permit us to express more sophisticated beliefs about the structure of the function of interest. For example, if we know that the function under consideration $y(x)$ is actually the sum of independent but unknown functions $a(x)$ and $b(x)$, over which we respectively have GPs with means $\boldsymbol{\mu}^{(a)}$ and $\boldsymbol{\mu}^{(b)}$ and covariances $\mathbf{K}^{(a)}$ and $\mathbf{K}^{(b)}$, the probability calculus gives

$$\begin{aligned} p(\,\boldsymbol{y} \mid I\,) &= \iint \delta(\boldsymbol{y} - (\boldsymbol{a} + \boldsymbol{b}))\; p(\,\boldsymbol{a} \mid I\,)\; p(\,\boldsymbol{b} \mid I\,)\; \mathrm{d}\boldsymbol{a}\, \mathrm{d}\boldsymbol{b} \\ &= \int \mathcal{N}\big(\boldsymbol{a}; \boldsymbol{\mu}^{(a)}, \mathbf{K}^{(a)}\big)\; \mathcal{N}\big(\boldsymbol{y} - \boldsymbol{a}; \boldsymbol{\mu}^{(b)}, \mathbf{K}^{(b)}\big)\, \mathrm{d}\boldsymbol{a} \\ &= \mathcal{N}\big(\boldsymbol{y}; \boldsymbol{\mu}^{(a)} + \boldsymbol{\mu}^{(a)}, \mathbf{K}^{(a)} + \mathbf{K}^{(b)}\big) \end{aligned} \tag{4.1.1}$$

Hence the GP over this $y(x)$ has a covariance that is simply the sum of the covariances for its two constituents. In a similar vein, consider a function $y(x)$ known to be the product of two independent functions $a(x)$ and $b(x)$. Note that this means that $\boldsymbol{y}$ is the Hadamard, or element by element, product $\boldsymbol{y} = \boldsymbol{a} \bullet \boldsymbol{b}$. We can approximate the

distribution of $\boldsymbol{y}$ by taking only its first two moments to give the GP

$$p(\,\boldsymbol{y}\mid I\,)\simeq\mathcal{N}\big(\boldsymbol{x};\boldsymbol{\mu}^{(a)}\bullet\boldsymbol{\mu}^{(a)},\mathbf{K}^{(a)}\bullet\mathbf{K}^{(b)}+\mathbf{K}^{(a)}\bullet\boldsymbol{\mu}^{(b)}\boldsymbol{\mu}^{(b),\mathsf{T}}+\mathbf{K}^{(b)}\bullet\boldsymbol{\mu}^{(a)}\boldsymbol{\mu}^{(a),\mathsf{T}}\big) \tag{4.1.2}$$

So long as the magnitude of our uncertainties, represented by the covariances, are not too large, the approximation is reasonably good[1]. Regardless of the approximation quality, however, the covariance of the right hand side of (4.1.2) is the correct covariance for $p(\,\boldsymbol{y}\mid I\,)$, and is hence a valid covariance. As such, with $\mu^{(a)}=\mu^{(b)}=0$, we have the result that the Hadamard product of two covariance matrices is a valid covariance in its own right, and hence that the product of two covariance functions is a covariance function.

4.2 Multiple Inputs and Outputs

The simple distance (3.2.2) used thus far clearly only allows for the simplest case of a one dimensional input x. In general, however, we assume our input space has finite dimension and write $x^{(e)}$ for the value of the eth element in x. $x_i^{(e)}$ is used for the value of the eth element of x_i.

Fortunately, it is not difficult to extend covariances to allow for multiple input dimensions. Perhaps the simplest approach is to take a covariance function that is the product of one-dimensional covariances over each input (the *product correlation* rule [Sasena, 2002]),

$$K(x_1,x_2;\phi)=\prod_e K^{(e)}\big(x_1^{(e)},\,x_2^{(e)};\phi\big) \tag{4.2.1}$$

where $K^{(e)}$ is a valid covariance over the eth input. As the product of covariances is a covariance (per (4.1.2)), (4.2.1) defines a valid covariance over the multi-dimensional input space.

We can also introduce distance functions appropriate for multiple inputs, such

[1]The third central moment is proportional to $(\boldsymbol{\mu}^{(a)}\otimes\mathbf{K}^{(a)})\bullet(\boldsymbol{\mu}^{(b)}\otimes\mathbf{K}^{(b)})$, where $\otimes$ represents the Kronecker product.

as the Mahalanobis distance

$$d^{(\mathrm{M})}(x_1, x_2; W) \triangleq \sqrt{(x_1 - x_2)^{\mathsf{T}} W^{-1} (x_1 - x_2)}\,, \tag{4.2.2}$$

where W is a covariance matrix of appropriate size. Note that if W is a diagonal matrix, its role in (4.2.2) is simply to provide an individual scale $w^{(e)} \triangleq \sqrt{W(e,e)}$ for the eth dimension. An illustration of what it means to have individual scales in this way is provided by Figure 4.1. However, by introducing off-diagonal elements, we can allow for correlations amongst the input dimensions.

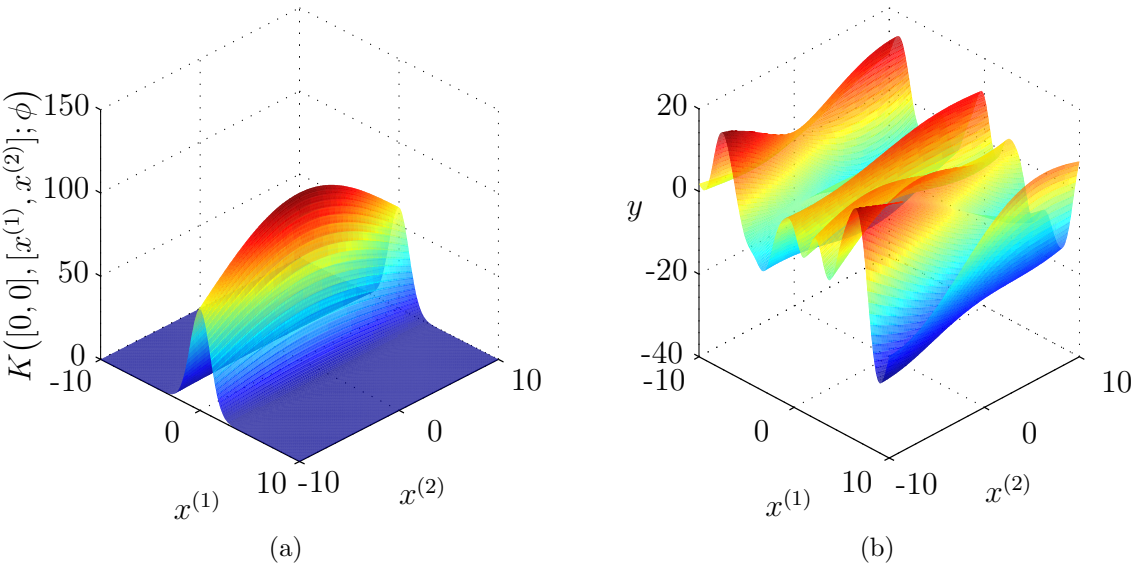


Figure 4.1: For a squared exponential covariance over two inputs $x^{(1)}$ and $x^{(2)}$, (a) the covariance as a function of separation in input space and (b) example data it might be used to model. Here we have taken $h = 10$, $w^{(1)} = 1$ and $w^{(2)} = 10$.

Note that a function with multiple outputs can always be dealt with by instead treating it as having a single output and an additional input, as illustrated in Figure 4.2. This additional (discrete) input, l, can be thought of as a label for the associated output. If our previous inputs were x', we now have inputs $x = [x', l]$; if we previously considered n outputs $[y^{(l)} : l = 1, \ldots, n]$, we now associate the single output $y^{(l)}$ with input $[x', l]$. An advantage of this approach is that we can readily treat the case in which all n outputs are not necessarily observed simultaneously. We now need merely

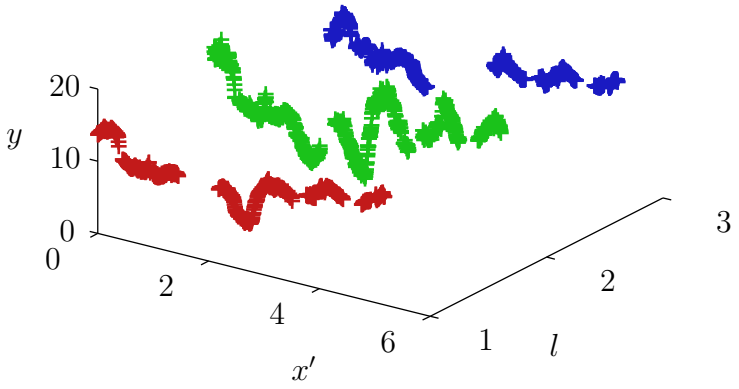


Figure 4.2: An illustration of data with three outputs treated as data with a single output and additional input, l.

to build a covariance over the new input space. One way to do this is to use the form (4.2.1) to give

$$K\big([x_1', l_1], [x_2', l_2]; \phi\big) = K^{(x')}(x_1', x_2'; \phi)\, K^{(l)}(l_1, l_2; \phi) \tag{4.2.3}$$

for appropriate covariances $K^{(x')}$ and $K^{(l)}$.

We now turn to the challenge of finding a general parameterisation for a covariance matrix. If the number n is not too large, we may wish to directly parameterise the covariance matrix $\mathbf{K}^{(l)}([1, \ldots, n], [1, \ldots, n])$. Similar is true, of course, for any problem for which we have a discrete input. Such a parameterisation would also be of use for the covariance matrix W in (4.2.2). Note that any covariance matrix W can be decomposed using the Cholesky factorisation $W = R^{\mathsf{T}} R$, where R is an upper triangular matrix. One approach, then, would be to use the non-zero elements of R as our hyper-parameters; any selection of those hyper-parameters will return a valid positive semi-definite covariance matrix W. Unfortunately, such hyperparameters are not intuitively connected with the elements of W; it would be difficult to assign priors to them given only our expectations of W. Fortunately, other parameterisations exist [Pinheiro and Bates, 1996].

In particular, we consider the *spherical parameterisation*. We write

$$W = S\ \mathrm{diag}(\tau)\,. \tag{4.2.4}$$

Here S is an upper triangular matrix, whose nth column contains the spherical coordinates in $\mathbb{R}^n$ of a point on the hypersphere $\mathbb{S}^{n-1}$, followed by the requisite number of zeros. As an example, S for a four dimensional space is

$$S = \begin{bmatrix} 1 & \cos\phi^{(1)} & \cos\phi^{(2)} & \cos\phi^{(4)} \\ 0 & \sin\phi^{(1)} & \sin\phi^{(2)}\cos\phi^{(3)} & \sin\phi^{(4)}\cos\phi^{(5)} \\ 0 & 0 & \sin\phi^{(2)}\sin\phi^{(3)} & \sin\phi^{(4)}\sin\phi^{(5)}\cos\phi^{(6)} \\ 0 & 0 & 0 & \sin\phi^{(4)}\sin\phi^{(5)}\sin\phi^{(6)} \end{bmatrix} . \tag{4.2.5}$$

This form ensures that $S^\mathsf{T}S$ has ones across its diagonal and hence all other entries may be thought of as akin to correlation coefficients, lying between -1 and 1. Meanwhile, τ is a vector of the scales for each dimension. If W is the covariance matrix parameterising the Mahalanobis distance, τ represents a vector of input scales for each input, and the off-diagonal elements of $S^\mathsf{T}S$ express the correlation amongst inputs. If instead we have used W as a parameterisation of $\mathbf{K}^{(l)}([1, \ldots, n], [1, \ldots, n])$, τ gives a vector of output scales corresponding to the different values $l = [1, \ldots, n]$. $S^\mathsf{T}S$ gives the correlation coefficients for each pair of values l might take.

These intuitive connections provide the motivation for the use of this parameterisation. Finally, note that the total number of hyperparameters required by the spherical parameterisation is $\frac{1}{2}n(n+1)$ if the side length of W is n. Note that this is the same number as would be required to parameterise the non-zero elements of an upper triangular Cholesky factor R directly. The spherical parametrisation hence requires a large number of hyperparameters, but allows us to express any possible covariance matrix.

Where we have correlated outputs, we may also wish to express some degree of translation or latency amongst them. For example, if our outputs are the measurements made by spatially separated sensors (labelled by l) at times x', the measurements made by one sensor may well be highly correlated with time-shifted measurements made by another sensor. This can be trivially achieved by introducing a delay hyperparameter Δ_l for each output l, and then taking

$$K\big([x'_1, l_1], [x'_2, l_2]; \phi\big) = K^{(x')}(x'_1 - \Delta_{l_1}, x'_2 - \Delta_{l_2}; \phi)\, K^{(l)}(l_1, l_2; \phi)\,. \tag{4.2.6}$$

This covariance means that a measurement from variable l_1 at input x'_1 is most strongly correlated with the reading from sensor l_2 at input $x'_2 - \Delta_{l_2} + \Delta_{l_1}$. As we have merely made a simple translation of the input space, (4.2.6) remains a covariance.

4.3 Warping of Inputs

Note that a covariance under any arbitrary map remains a covariance [MacKay, 1998, Schmidt and O'Hagan, 2003].

Theorem 1. *For any function $u : x \to u(x)$, a covariance K defined over its range gives rise to a valid covariance K' over its domain, as per*

$$K'\big(x_1, x_2\big) \triangleq K\big(u(x_1), u(x_2)\big)\,.$$

Proof. Consider the covariance matrix over any vector of inputs $\boldsymbol{x}$ from the domain,

$$K'\big(\boldsymbol{x}, \boldsymbol{x}\big) = K\big(\boldsymbol{u}, \boldsymbol{u}\big)\,,$$

where $\boldsymbol{u} \triangleq u(\boldsymbol{x})$. By assumption, $K\big(\boldsymbol{u}, \boldsymbol{u}\big)$ is a positive semi-definite matrix and hence so is $K'\big(\boldsymbol{x}, \boldsymbol{x}\big)$. □

Hence we can use simple, stationary covariances in order to construct more complex (possibly non-stationary) covariances. A particularly relevant example of this,

$$u(x) \triangleq (u^{(a)}(x), u^{(b)}(x)) \triangleq \Big(\cos\big(2\pi\frac{x}{T}\big), \sin\big(2\pi\frac{x}{T}\big)\Big)\,,$$

allows us to modify our simple covariance functions (3.2.1) to model periodic functions. Over the new transformed variable, a function of the two-dimensional input u, we assume an homogenous covariance (3.2.1) with a simple version of the Mahalanobis distance (4.2.2)

$$d(u_1, u_2; w, T) \triangleq \frac{1}{2w}\sqrt{\big(u^{(a)}(x_1) - u^{(a)}(x_2)\big)^2 + \big(u^{(b)}(x_1) - u^{(b)}(x_2)\big)^2} \qquad (4.3.1)$$

We can now take this covariance over u as a valid covariance over x. As a result, we have the covariance function

$$K^{(\mathrm{per})}(x_1, x_2; h, w, T) \triangleq h^2\kappa\bigg(\frac{1}{w}\sin\Big(\pi\Big|\frac{x_1 - x_2}{T}\Big|\Big)\bigg)\,, \qquad (4.3.2)$$

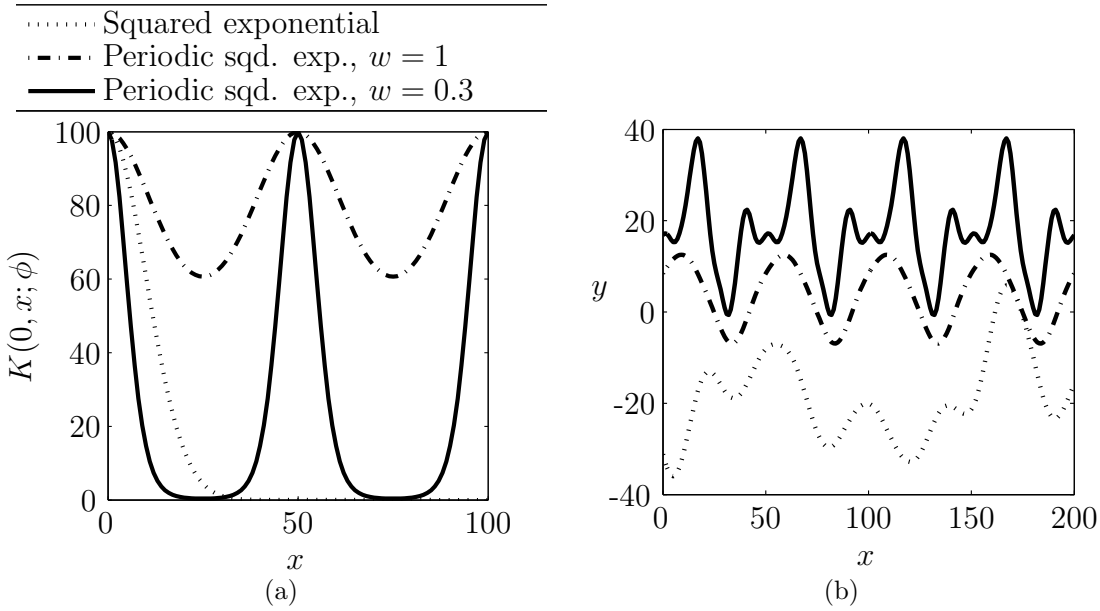


Figure 4.3: For periodic (3.2.3) and non-periodic squared exponential (4.3.3) covariance functions, (a) the covariance as a function of separation and (b) example data they might be appropriate for. For the squared exponential, $w = 10$ and $h = 10$, and for the periodic squared exponential, $T = 50$ and $h = 10$.

which gives, for the example of the squared exponential (3.2.3),

$$K^{(\text{per-SE})}(x_1, x_2; h, w, T) \triangleq h^2 \exp\left(-\frac{1}{2w^2}\sin^2\left(\pi\left|\frac{x_1 - x_2}{T}\right|\right)\right). \tag{4.3.3}$$

In this case the output scale h serves as the amplitude, and T is the period. w is a roughness parameter that serves a similar role to the input scale w in our stationary covariances. With this formulation, we can perform inference about functions of arbitrary roughness and with arbitrary period. Figure 4.3 contrasts squared exponential and periodic squared exponential covariance functions.

4.4 Changepoints

We now describe how to construct appropriate covariance functions for functions that experience sudden changes in their characteristics [Garnett et al., 2009, 2010a]. This section is meant to be expository; the covariance functions we describe are intended as examples rather than an exhaustive list of possibilities. To ease exposition, we assume the input variable of interest x is one-dimensional (e.g. temporal). Our covariances

over this one-dimensional x could be combined with others as per (4.2.1), for example. The covariances will also be readily extended to changepoints in higher dimensional spaces, given an appropriate parameterisation of the boundary between regions of different characteristics. While such a boundary is trivially specified by a single hyperparameter for a one-dimensional input, in higher dimensions we may require more hyperparameters to specify the dividing hyperplane, or whatever boundary is appropriate. We construct appropriate covariance functions for a number of types of changepoint. Some examples of these are illustrated in Figure 4.4.

4.4.1 A drastic change in covariance

Suppose a function of interest is well-behaved except for a drastic change at the point x_c, which separates the function into two regions with associated covariance functions $K^{(\mathrm{a})}(\cdot,\cdot;\phi_a)$ before x_c and $K^{(\mathrm{b})}(\cdot,\cdot;\phi_b)$ after, where ϕ_a and ϕ_b represent the values of any hyperparameters associated with $K^{(\mathrm{a})}$ and $K^{(\mathrm{b})}$, respectively. If the change is so drastic that the observations before x_c are completely uninformative about the observations after the changepoint; that is, if

$$p\big(\,\boldsymbol{y}_{\geq x_c} \mid \boldsymbol{z},\, I\,\big) = p\big(\,\boldsymbol{y}_{\geq x_c} \mid \boldsymbol{z}_{\geq x_c},\, I\,\big)\,,$$

where the subscripts indicate ranges of data segmented by x_c (e.g. $\boldsymbol{z}_{\geq x_c}$ is the subset of $\boldsymbol{z}$ containing only observations after the changepoint), then the appropriate covariance function is trivial. This function can be modelled using the covariance function $K^{(\mathrm{A})}$ defined by

$$K^{(\mathrm{A})}(x_1, x_2; \phi_A) \triangleq \begin{cases} K^{(\mathrm{a})}(x_1, x_2; \phi_a) & (x_1, x_2 < x_c) \\ K^{(\mathrm{b})}(x_1, x_2; \phi_b) & (x_1, x_2 \geq x_c) \\ 0 & (\text{otherwise.}) \end{cases}. \tag{4.4.1}$$

The new set of hyperparameters $\phi_A \triangleq \{\phi_a, \phi_b, x_c\}$ contains knowledge about the original hyperparameters of the covariance functions as well as the location of the changepoint. This covariance function is easily seen to be positive semi-definite and hence admissible.

Theorem 2. *$K^{(A)}$ is a valid covariance function.*

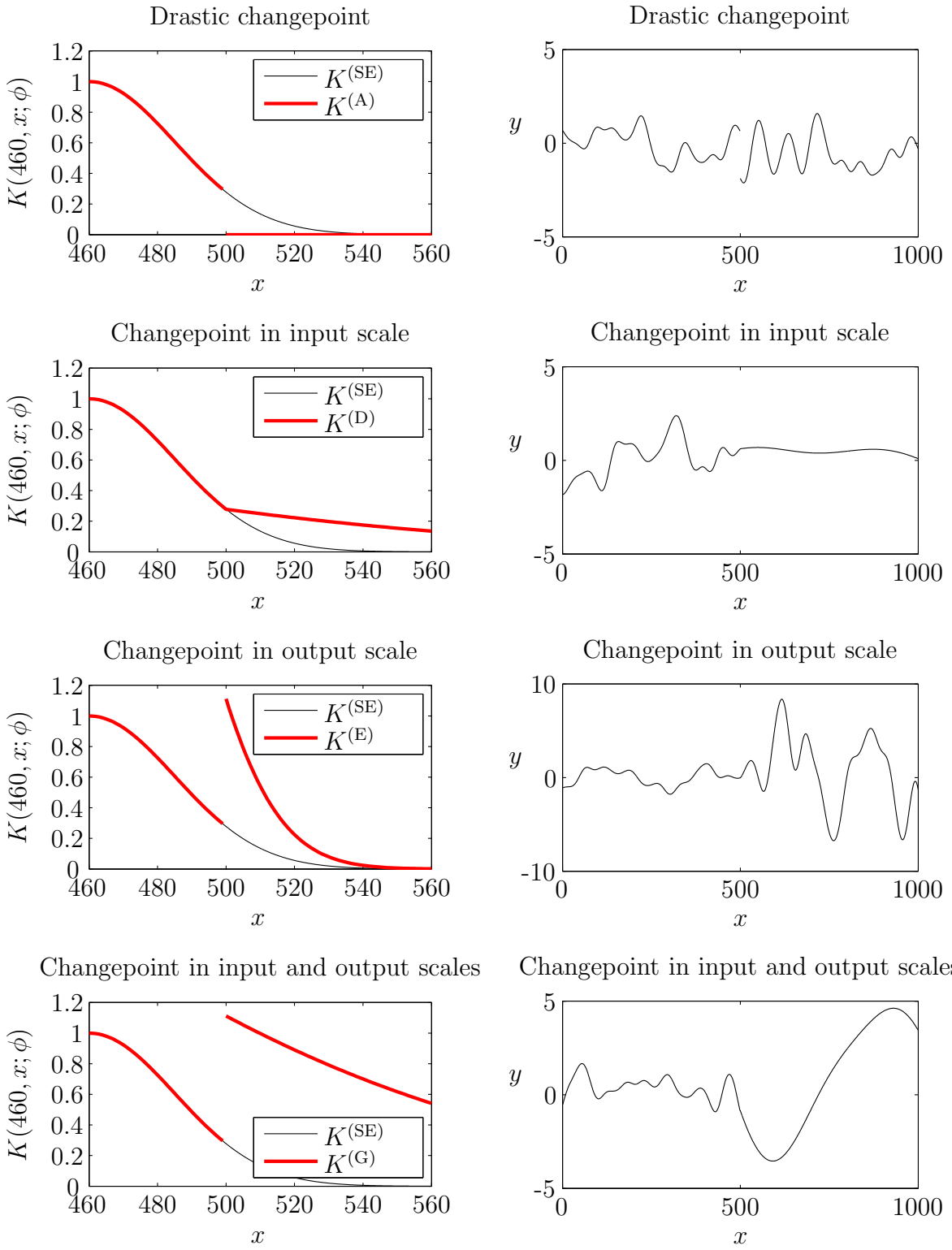


Figure 4.4: Example covariance functions for the modelling of data with changepoints, and associated example data that they might be appropriate for.

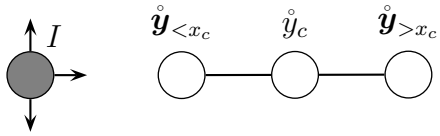


Figure 4.5: Bayesian Network for the smooth drastic change model. I is the context, correlated with all other nodes.

Proof. We show that any covariance matrix given by $K^{(\mathrm{A})}$ is positive semi-definite. Consider an arbitrary set of input points $\boldsymbol{x}$ in the domain of interest. By appropriately ordering the points in $\boldsymbol{x}$, we may write the covariance matrix $K^{(\mathrm{A})}(\boldsymbol{x}, \boldsymbol{x})$ as the block-diagonal matrix

$$\begin{bmatrix} K^{(\mathrm{a})}(\boldsymbol{x}_{<x_c}, \boldsymbol{x}_{<x_c}; \phi_a) & \mathbf{0} \\ \mathbf{0} & K^{(\mathrm{b})}(\boldsymbol{x}_{\geq x_c}, \boldsymbol{x}_{\geq x_c}; \phi_b) \end{bmatrix};$$

the eigenvalues of $K^{(\mathrm{A})}(\boldsymbol{x}, \boldsymbol{x})$ are therefore the eigenvalues of the blocks. Because both $K^{(\mathrm{a})}$ and $K^{(\mathrm{b})}$ are valid covariance functions, their corresponding covariance matrices are positive semi-definite, and therefore eigenvalues of $K^{(\mathrm{A})}(\boldsymbol{x}, \boldsymbol{x})$ are nonnegative. □

4.4.2 A smooth drastic change in covariance

Suppose a continuous function of interest is best modelled by different covariance functions, before and after a changepoint x_c. The function values after the changepoint are conditionally independent of the function values before, given the value at the changepoint itself (the function is continuous across the changepoint). The Bayesian network for this probabilistic structure is depicted in Figure 4.5. This represents an extension to the drastic covariance described above; our two regions can be drastically different, but we can still enforce smoothness across the boundary between them.

The changepoint separates the function into two regions with associated covariance functions $K^{(\mathrm{a})}(\cdot, \cdot; \phi_a)$ before x_c and $K^{(\mathrm{b})}(\cdot, \cdot; \phi_b)$ after, where ϕ_a and ϕ_b represent the values of any hyperparameters associated with $K^{(\mathrm{a})}$ and $K^{(\mathrm{b})}$, respectively. We

may model the function using the covariance function $K^{(\mathrm{B})}$ defined by:

$$K^{(\mathrm{B})}(x_1, x_2; \phi_a, \phi_b) \triangleq$$
$$\begin{cases} K^{(\mathrm{a})}(x_1, x_2; \phi_a) - K^{(\mathrm{a})}(x_1, x_c; \phi_a)\, K^{(\mathrm{a})}(x_c, x_c; \phi_a)^{-1}\, K^{(\mathrm{a})}(x_c, x_2; \phi_a) & (x_1, x_2 < x_c) \\ K^{(\mathrm{b})}(x_1, x_2; \phi_b) - K^{(\mathrm{b})}(x_1, x_c; \phi_b)\, K^{(\mathrm{b})}(x_c, x_c; \phi_b)^{-1}\, K^{(\mathrm{b})}(x_c, x_2; \phi_b) & (x_1, x_2 > x_c) \\ 0 & (\text{otherwise}) \end{cases}, \tag{4.4.2}$$

On either side of the changepoint, this covariance has exactly the form we'd expect given an observation of the function at x_c. Given that observation, however, there is no covariance across the changepoint. We call $K^{(\mathrm{B})}$ the *continuous conditionally independent* covariance function. This covariance function can be extended to multiple changepoints, boundaries in multi-dimensional spaces, and also to cases where function derivatives are continuous at the changepoint.

As a slight extension of $K^{(\mathrm{B})}$, consider a function that undergoes a temporary excursion from an otherwise constant value of zero. This excursion is known to be smooth, that is, it both begins and ends at zero. We define the beginning of the excursion as x_{c_1} and its end as x_{c_2}. Essentially, we have changepoints as considered by (4.4.2) at both x_{c_1} and x_{c_2}. We can hence write the covariance function appropriate for this function as

$$K^{(\mathrm{C})}(x_1, x_2; \phi, x_{c_1}, x_{c_2}) \triangleq$$
$$K(x_1, x_2; \phi) - K\left(x_1, \begin{bmatrix} x_{c_1} \\ x_{c_2} \end{bmatrix}; \phi\right) K\left(\begin{bmatrix} x_{c_1} \\ x_{c_2} \end{bmatrix}, \begin{bmatrix} x_{c_1} \\ x_{c_2} \end{bmatrix}; \phi\right)^{-1} K\left(\begin{bmatrix} x_{c_1} \\ x_{c_2} \end{bmatrix}, x_2; \phi\right), \tag{4.4.3}$$

for $x_{c_1} < x_1 < x_{c_2}$ and $x_{c_1} < x_2 < x_{c_2}$, and $K^{(\mathrm{C})}(x_1, x_2; \phi, x_{c_1}, x_{c_2}) = 0$ otherwise. Here (unsubscripted) K is a covariance function that describes the dynamics of the excursion itself.

4.4.3 A sudden change in input scale

Suppose a function of interest is well-behaved except for a drastic change in the input scale w at time x_c, which separates the function into two regions with different degrees of long-term dependence.

Let w_a and w_b represent the input scale of the function before and after the changepoint at x_c, respectively. Suppose we wish to model the function with a covariance function K of the form (3.2.1) that would be appropriate except for the change in input scale. We may model the function using the covariance function $K^{(\mathrm{D})}$ defined by

$$K^{(\mathrm{D})}(x_1, x_2; h, w_a, w_b, x_c) \triangleq \begin{cases} K(x_1, x_2; h, w_a) & (x_1, x_2 < x_c) \\ K(x_1, x_2; h, w_b) & (x_1, x_2 \geq x_c) \\ h^2 \, \kappa\left(\left|\frac{x_c - x_1}{w_a}\right| + \left|\frac{x_c - x_2}{w_b}\right|\right) & (\text{otherwise}) \end{cases}, \tag{4.4.4}$$

where κ is an appropriate function defining an homogenous covariance function of the form (3.2.1).

Theorem 3. *$K^{(D)}$ is a valid covariance function.*

Proof. Consider the map defined by

$$u(x; x_c) \triangleq \begin{cases} \frac{x}{w_a} & (x < x_c) \\ \frac{x_c}{w_a} + \frac{x - x_c}{w_b} & (x \geq x_c) \end{cases}. \tag{4.4.5}$$

Note that $K^{(\mathrm{D})}(x_1, x_2; h, w_a, w_b, x_c)$ is equal to $K(u(x_1; x_c), u(x_2; x_c); h, 1)$, the original covariance function with equivalent output scale and unit input scale evaluated on the input points after transformation by u. Because K is a valid covariance function, the result follows. □

The function u in the proof above motivates the definition of $K^{(\mathrm{D})}$: by rescaling the input variable appropriately, the change in input scale is removed.

4.4.4 A sudden change in output scale

Suppose a function of interest is well-behaved except for a drastic change in the output scale h at time x_c, which separates the function into two regions.

Let $y(x)$ represent the function of interest and let h_a and h_b represent the output scale of $y(x)$ before and after the changepoint at x_c, respectively. Suppose we wish to model the function with an isotropic covariance function K of the form (3.2.1) that would be appropriate except for the change in output scale. To derive the appropriate

covariance function, we model $y(x)$ as the product of a function with unit output scale, $g(x)$, and a piecewise-constant scaling function, s, defined by

$$s(x; x_c) \triangleq \begin{cases} h_a & x < x_c \\ h_b & x \geq x_c \end{cases}. \tag{4.4.6}$$

Given the model $y(x) = s(x)\big(g(x) - g(x_c)\big)$, the appropriate covariance function for y is immediate. We may use the covariance function $K^{(\mathrm{E})}$ defined by

$$\begin{aligned} K^{(\mathrm{E})}(x_1, x_2; h_a^2, h_b^2, w, x_c) &\triangleq s(x_1; x_c)\, K(x_1, x_2; 1, w)\, s(x_2; x_c) \\ &= \begin{cases} K(x_1, x_2; h_a, w) & (x_1, x_2 < x_c) \\ K(x_1, x_2; h_b, w) & (x_1, x_2 \geq x_c) \\ K(x_1, x_2; (h_a h_b)^{\frac{1}{2}}, w) & (\text{otherwise}) \end{cases}. \end{aligned} \tag{4.4.7}$$

As y is simply a linear transform of g, the form of $K^{(\mathrm{E})}$ follows from the properties of the multivariate Gaussian (see Appendix A.1).

4.4.5 Discussion

The covariance functions above can be extended in a number of ways. They can firstly be extended to handle multiple changepoints. Here we need simply to introduce additional hyperparameters for their locations and the values of the appropriate covariance characteristics, such as input scales, within each segment. Note, however, that at any point in time our model only needs to accommodate the volume of data spanned by the window. In practice, allowing for one or two changepoints is usually sufficient for the purposes of prediction, given that the data prior to a changepoint is typically weakly correlated with data in the current regime of interest. Therefore we can circumvent the computationally onerous task of simultaneously marginalising the hyperparameters associated with the entire data stream. If no changepoint is present in the window, the posterior distribution for its location will typically be concentrated at its trailing edge. A changepoint at such a location will have no influence on predictions; the model is hence able to effectively manage the absence of changepoints.

Additionally, if multiple parameters undergo a change at some point in time, an appropriate covariance function can be derived by combining the above results. For example, a function that experiences a change in both input scale and output scale could be readily modelled by

$$K^{(\mathrm{G})}(x_1, x_2; h_a, h_b, w_a, w_b, x_c) \triangleq s(x_1; x_c)\, K(u(x_1; x_c), u(x_2; x_c); 1, 1)\, s(x_2; x_c), \tag{4.4.8}$$

where u is as defined in (4.4.5) and s is as defined in (4.4.6).

Unlike covariance functions, mean functions are not required to be positive semi-definite, and are hence readily constructed to incorporate changepoints. This will, however, require the addition of further hyperparameters specifying the changepoint type and location. In many cases these will be shared with the relevant covariance function, although of course it is possible to have a changepoint in mean independent of any changepoints in covariance.

4.5 Changepoints in Observation Likelihood

In Section 3.3, we took the observation likelihood $p(\, \boldsymbol{z} \mid \boldsymbol{y}, \phi, I\,)$ as being both constant and of the simple independent form represented in (3.3.5). This was motivated by the analytic niceties afforded by the use of a Gaussian observation likelihood. However, we can benefit from those same properties while using a Gaussian $p(\, \boldsymbol{z} \mid \boldsymbol{y}, \phi, I\,)$ with more sophisticated covariances. In particular, our changepoint covariances from Section 4.4 allow us to consider observation processes that themselves undergo changepoints [Garnett et al., 2010a].

Our principal motivation here is fault detection and removal [Reece et al., 2009]. A sensor fault essentially implies that the relationship between the underlying, or plant, process y and the observed values z is temporarily complicated. In situations where a model of the fault is known, the faulty observations need not be discarded; they may still contain valuable information about the plant process. We distinguish *fault removal*, for which the faulty observations are discarded, from *fault recovery*, for

which the faulty data is utilised with reference to a model of the fault.

We take, as before, a GP over y,

$$p(\,\boldsymbol{y}\mid\phi,\,I\,)\triangleq\mathcal{N}(\boldsymbol{y};\boldsymbol{\mu}(\boldsymbol{x}),\mathbf{K}(\boldsymbol{x},\boldsymbol{x}))$$

and consider the more general observation model

$$p(\,\boldsymbol{z}\mid\boldsymbol{y},\,\phi,\,I\,)=\mathcal{N}\big(\boldsymbol{z};\mathbf{M}(\boldsymbol{x})\,\boldsymbol{y}+\boldsymbol{c}(\boldsymbol{x}),\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x},\boldsymbol{x})\big)\,, \tag{4.5.1}$$

which allows us to consider a myriad of possible types of fault modes. Here μ, K M, c and $K^{(\mathrm{F})}$ are all specified by the hyperparameters, although for notational convenience we do not write the dependence upon ϕ. $K^{(\mathrm{F})}$ is a covariance function associated with the the fault model, which will likely be different from the covariance over y, K. With this model, we have the posteriors

$$p(\boldsymbol{y}_\star|\boldsymbol{z}_d,\,\phi,\,I)=\mathcal{N}\big(\boldsymbol{y}_\star;\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\,\phi,\,I)\,,\mathbf{C}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\,\phi,\,I)\,\big)\,, \tag{4.5.2}$$

where we have

$$\begin{aligned}\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\,\phi,\,I)&=\boldsymbol{\mu}(\boldsymbol{x}_\star)+\\&\qquad\mathbf{K}(\boldsymbol{x}_\star,\boldsymbol{x}_d)\,\mathbf{M}(\boldsymbol{x}_d)^{\mathsf{T}}\mathbf{V}^{(\mathrm{F})}(\boldsymbol{x}_d,\boldsymbol{x}_d)^{-1}\big(\boldsymbol{z}_d-\mathbf{M}(\boldsymbol{x}_d)\,\boldsymbol{\mu}(\boldsymbol{x}_d)-\boldsymbol{c}(\boldsymbol{x}_d)\big)\\\mathbf{C}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\,\phi,\,I)&=\mathbf{K}(\boldsymbol{x}_\star,\boldsymbol{x}_\star)-\mathbf{K}(\boldsymbol{x}_\star,\boldsymbol{x}_d)\,\mathbf{M}(\boldsymbol{x}_d)^{\mathsf{T}}\,\mathbf{V}^{(\mathrm{F})}(\boldsymbol{x}_d,\boldsymbol{x}_d)^{-1}\mathbf{M}(\boldsymbol{x}_d)\,\mathbf{K}(\boldsymbol{x}_d,\boldsymbol{x}_\star)\,,\end{aligned}$$

and

$$\mathbf{V}^{(\mathrm{F})}(\boldsymbol{x}_d,\boldsymbol{x}_d)\triangleq\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x},\boldsymbol{x})+\mathbf{M}(\boldsymbol{x})^{\mathsf{T}}\,\mathbf{K}(\boldsymbol{x},\boldsymbol{x})\,\mathbf{M}(\boldsymbol{x})\,.$$

Note these results also hold even for non-faulty observation likelihoods of the form (4.5.1).

If required, we can also determine the posterior for the fault contributions, defined as $f\triangleq z-y$.

$$\begin{aligned}p(\boldsymbol{f}_\star|\boldsymbol{z}_d,\,\phi,\,I)&=\iint p(\boldsymbol{f}_\star|\boldsymbol{z}_\star,\boldsymbol{y}_\star,\phi,I)\,p(\boldsymbol{z}_\star|\boldsymbol{y}_\star,\phi,I)\,p(\boldsymbol{y}_\star|\boldsymbol{z}_d,\phi,I)\,\mathrm{d}\boldsymbol{y}_\star\mathrm{d}\boldsymbol{z}_\star\\&=\iint\delta(\boldsymbol{f}_\star-(\boldsymbol{z}_\star-\boldsymbol{y}_\star))\,\mathcal{N}\big(\boldsymbol{z}_\star;\mathbf{M}(\boldsymbol{x}_\star)\,\boldsymbol{y}_\star+\boldsymbol{c}(\boldsymbol{x}),\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x},\boldsymbol{x})\big)\\&\qquad\qquad\mathcal{N}\big(\boldsymbol{y}_\star;\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\phi,I)\,,\mathbf{C}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\,\phi,I)\,\big)\mathrm{d}\boldsymbol{z}_\star\,\mathrm{d}\boldsymbol{y}_\star\\&=\mathcal{N}\Big(\boldsymbol{f}_\star;\boldsymbol{m}\Big(\mathring{\boldsymbol{f}}_\star\,\Big|\,\boldsymbol{z}_d,\,\phi,\,I\Big)\,,\mathbf{C}\Big(\mathring{\boldsymbol{f}}_\star\,\Big|\,\boldsymbol{z}_d,\,\phi,\,I\Big)\Big)\,,\end{aligned} \tag{4.5.3}$$

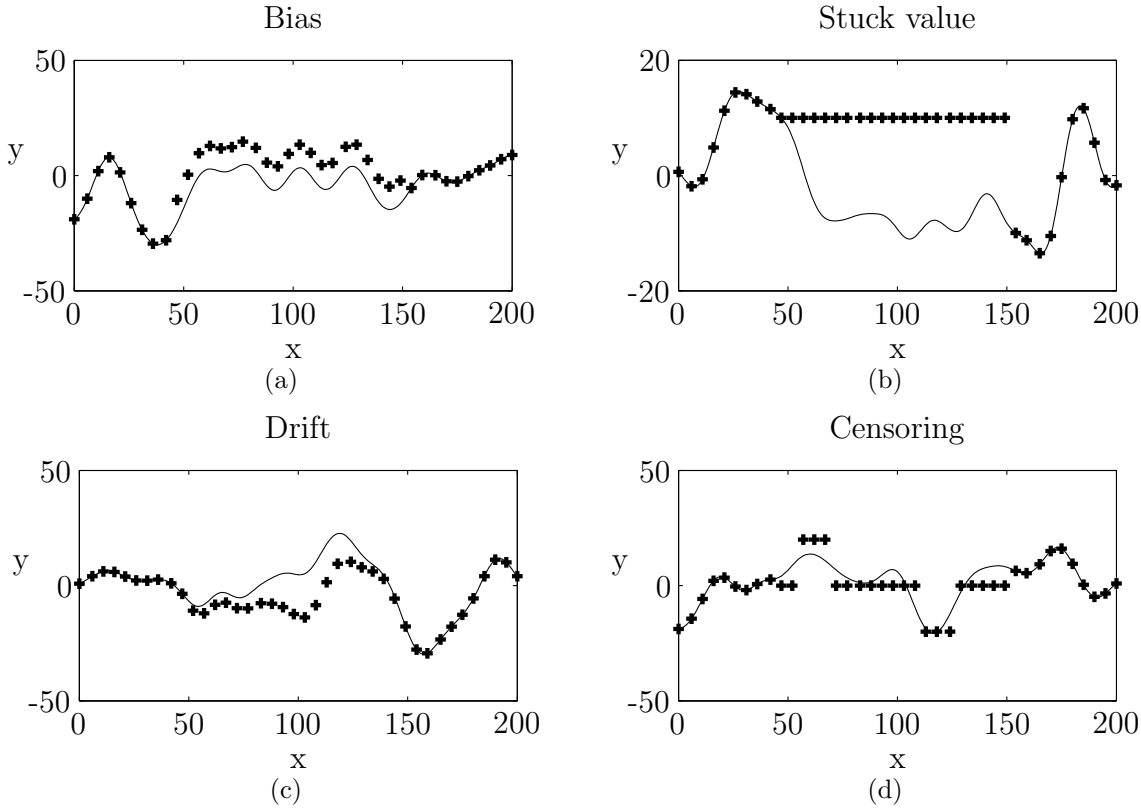


Figure 4.6: Example observations (plotted as crosses) of a function $y(x)$ (plotted as a thin line) that are noiseless for $x < 50$ and $150 \leq x$. For $50 \leq x < 150$, the observations are respectively (a) corrupted by a constant bias, (b) stuck at a constant value, (c) corrupted by drift and (d) rounded to the nearest multiple of 20.

where we have

$$\begin{aligned}
\boldsymbol{m}\Big(\mathring{\boldsymbol{f}}_\star \,\Big|\, \boldsymbol{z}_d,\, \phi,\, I\Big) &= \big(\mathbf{M}(\boldsymbol{x}_\star) - \mathbf{I}_\star\big)\, \boldsymbol{m}\big(\mathring{\boldsymbol{y}}_\star \,\big|\, \boldsymbol{z}_d,\, \phi,\, I\big) + \boldsymbol{c}(\boldsymbol{x}_\star) \\
\mathbf{C}\Big(\mathring{\boldsymbol{f}}_\star \,\Big|\, \boldsymbol{z}_d,\, \phi,\, I\Big) &= \\
&\qquad \mathbf{K}^{(\mathrm{F})}(\boldsymbol{x}_\star, \boldsymbol{x}_\star) + \big(\mathbf{M}(\boldsymbol{x}_\star) - \mathbf{I}_\star\big)\, \mathbf{C}(\mathring{\boldsymbol{y}}_\star \,|\, \boldsymbol{z}_d,\, \phi,\, I)\, \big(\mathbf{M}(\boldsymbol{x}_\star) - \mathbf{I}_\star\big)^{\mathsf{T}},
\end{aligned}$$

where $\mathbf{I}_\star$ is the identity matrix of side length equal to $\boldsymbol{x}_\star$. We now consider some illustrative examples of fault types modelled by this approach.

4.5.1 Bias

Perhaps the simplest fault mode is that of *bias*, in which the readings are simply offset from the true values by some constant amount (and then, potentially, further

corrupted by additive Gaussian noise). An example of this fault mode is illustrated in Figure 4.6a. Clearly knowing the fault model in this case will allow us to extract information from the faulty readings; here we are able to perform fault recovery. In this scenario, $\mathbf{M}(\boldsymbol{x})$ is the identity matrix, $\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x}, \boldsymbol{x})$ is a diagonal matrix whose diagonal elements are identical noise variances (as implicit in the simple IID observation model (3.3.5)) and $\boldsymbol{c}(\boldsymbol{x})$ is a non-zero bias constant for x lying in the faulty period, and zero otherwise. The value of the offset and the start and finish times for the fault are additional hyperparameters to be included in ϕ.

4.5.2 Stuck value

Another simple fault model is that of a *stuck value*, in which our faulty readings return a constant value regardless of the actual plant process, as in Figure 4.6b. We consider the slightly more general model in which those faulty observations may also include a Gaussian noise component on top of the constant value. Here, of course, we can hope only for fault removal; the faulty readings are not at all pertinent to inference about the underlying variables of interest. This model has, as before, $\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x}, \boldsymbol{x})$ equal to a diagonal matrix whose diagonal elements are identical noise variances. $\mathbf{M}(\boldsymbol{x})$ is another diagonal matrix whose ith diagonal element is equal to zero if x_i is within the faulty region, and is equal to one otherwise. $\mathbf{M}(\boldsymbol{x})$ hence serves to select only non-faulty readings. $\boldsymbol{c}(\boldsymbol{x})$, then, is equal to a constant value (the stuck value) if x_i is within the faulty region, and is equal to zero otherwise. Here, as for the biased case, we have additional hyperparameters corresponding to the stuck value and the start and finish times of the fault.

4.5.3 Drift

The final fault we consider is that of drift. Here our sensor readings undergo a smooth excursion from the plant process; that is, they gradually 'drift' away from the real values, before eventually returning back to normality. An example of a drift fault is depicted in Figure 4.6c. Unsurprisingly, here $\mathbf{K}^{(\mathrm{F})}(\boldsymbol{x}, \boldsymbol{x})$ is a drift covariance $K^{(\mathrm{C})}$ as

defined in (4.4.3), with the addition of noise variance terms to its diagonal as required. Otherwise, $\mathbf{M}(\boldsymbol{x})$ is the appropriate identity matrix and $\boldsymbol{c}(\boldsymbol{x})$ is a zero vector. With knowledge of this model, fault recovery is certainly possible. The model requires additional parameters that define the relevant covariance K used in (4.4.3), as well as the fault start and finish times.

4.6 Censored Observations

So far, we have considered only Gaussian observation likelihoods. However, in many contexts, we instead observe *censored* observations [Osborne et al., 2010b]. That is, we might observe that a variable was above or below certain thresholds, but no more.

Examples are rich within the context of weather sensor networks. Float sensors are prone to becoming lodged on sensor posts, reporting only that the water level is below that at which it is stuck. Other observations are problematically rounded to the nearest integer – if we observe a reading of x, we can say only that the true value was between $x - 0.5$ and $x + 0.5$. We extend our sequential algorithms to allow for such a noise model. An example of such rounding is illustrated in Figure 4.6d.

More precisely, we assume that we actually observe bounds $\boldsymbol{b}_c$ that constrain Gaussian-noise corrupted versions $\boldsymbol{z}_c$ of the underlying variables of interest $\boldsymbol{y}_c$ at $\boldsymbol{y}_c$. This framework allows for imprecise censored observations. Note that the noise variance for censored observations may differ from the noise variance associated with other observations. Conditioned on a combination of censored and un-censored observations, the distribution for our variables of interest is

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \boldsymbol{b}_c,\, I\,) = \frac{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \boldsymbol{z}_c,\, \phi,\, I\,)\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d,\, \phi,\, I\,)\, p(\,\boldsymbol{z}_d \mid \phi,\, I\,)\, p(\,\phi \mid I\,)}{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d,\, \phi,\, I\,)\, p(\,\boldsymbol{z}_d \mid \phi,\, I\,)\, p(\,\phi \mid I\,)}\,. \tag{4.6.1}$$

While we cannot determine this full, non-Gaussian distribution easily, we can determine the analytic forms for its mean and covariance. We use the abbreviations $\boldsymbol{m}_{c|d,\phi} \triangleq \boldsymbol{m}(\mathring{\boldsymbol{z}}_c|\boldsymbol{z}_d, \phi, I)$ and $\mathbf{C}_{c|d,\phi} \triangleq \mathbf{C}(\mathring{\boldsymbol{z}}_c|\boldsymbol{z}_d, \phi, I)$. To reflect the influence of our

censored observations, our likelihoods become

$$p(\,\boldsymbol{z}_d,\,\boldsymbol{b}_c\mid\phi,\,I\,)=\mathcal{N}(\boldsymbol{z}_d;\boldsymbol{m}(\mathring{\boldsymbol{z}}_d|\phi,I)\,,\mathbf{C}(\mathring{\boldsymbol{z}}_d|\phi,I))\int_{\boldsymbol{b}_c}\mathrm{d}\boldsymbol{z}_c\,\mathcal{N}\big(\boldsymbol{z}_c;\boldsymbol{m}_{c|d,\phi},\mathbf{C}_{c|d,\phi}\big)\;, \tag{4.6.2}$$

giving the new weights over hyperparameter samples $\boldsymbol{\rho}^{(cd)}$. We can then write our predictive mean as

$$\begin{aligned}\boldsymbol{m}(\mathring{\boldsymbol{y}}_\star\,|\,\boldsymbol{z}_d,\boldsymbol{b}_c,I)=\sum_{i\in s}\rho_i^{(cd)}\Bigg(&\boldsymbol{\mu}(\boldsymbol{x}_\star;\phi_i)+\mathbf{K}(\boldsymbol{x}_\star,[\boldsymbol{x}_c,\boldsymbol{x}_d];\phi_i)\\&\mathbf{V}([\boldsymbol{x}_c,\boldsymbol{x}_d],[\boldsymbol{x}_c,\boldsymbol{x}_d];\phi_i)^{-1}\begin{bmatrix}\frac{\int_{\boldsymbol{b}_c}\mathrm{d}\boldsymbol{z}_c\,\boldsymbol{z}_c\,\mathcal{N}\left(\boldsymbol{z}_c;\boldsymbol{m}_{c|d,\phi_i},\mathbf{C}_{c|d,\phi_i}\right)}{\int_{\boldsymbol{b}_c}\mathrm{d}\boldsymbol{z}_c\,\mathcal{N}\left(\boldsymbol{z}_c;\boldsymbol{m}_{c|d,\phi_i},\mathbf{C}_{c|d,\phi_i}\right)}-\boldsymbol{\mu}(\boldsymbol{y}_c;\phi_i)\\ \boldsymbol{z}_d-\boldsymbol{\mu}(\boldsymbol{x}_d;\phi_i)\end{bmatrix}\Bigg),\end{aligned} \tag{4.6.3}$$

noting that a censored observation is intuitively treated as an uncensored observation equal to the conditional mean of the GP over the bounded region. We have also the predictive covariance

$$\begin{aligned}\mathbf{C}(\mathring{\boldsymbol{y}}_{st}\,|\,\boldsymbol{z}_d,\boldsymbol{b}_c,I)=\sum_{i\in s}\rho_i^{(cd)}\Big(&\mathbf{K}(\boldsymbol{x}_\star,\boldsymbol{x}_\star;\phi_i)-\\&\mathbf{K}(\boldsymbol{x}_\star,[\boldsymbol{x}_c,\boldsymbol{x}_d];\phi_i)\mathbf{V}([\boldsymbol{x}_c,\boldsymbol{x}_d],[\boldsymbol{x}_c,\boldsymbol{x}_d];\phi_i)^{-1}\mathbf{K}([\boldsymbol{x}_c,\boldsymbol{x}_d],\boldsymbol{y}_\star;\phi_i)\Big)\,.\end{aligned} \tag{4.6.4}$$

We now have the problem of determining the integrals

$$\int_{\boldsymbol{b}_c}\mathrm{d}\boldsymbol{z}_c\,\mathcal{N}\big(\boldsymbol{z}_c;\boldsymbol{m}_{c|d,\phi},\mathbf{C}_{c|d,\phi}\big)\qquad\text{and}\qquad\int_{\boldsymbol{b}_c}\mathrm{d}\boldsymbol{z}_c\,\boldsymbol{z}_c\,\mathcal{N}\big(\boldsymbol{z}_c;\boldsymbol{m}_{c|d,\phi},\mathbf{C}_{c|d,\phi}\big)\;, \tag{4.6.5}$$

which are non-analytic. With only a single censored observation, we can use standard functions for such integrals. By assuming that our censored observations are independent of one another (conditioned on the other observations $\boldsymbol{z}_d$), we can also make use of such results. Alternatively, techniques of numerical quadrature can be used to approximate such integrals, as will be discussed later in Chapter 7. This does, however, introduce a practical limit to the number of censored observations (equal to the dimensionality of the required integrals) we can simultaneously consider.

4.7 Derivative and Integral Observations

We now highlight a very useful property of a GP [Murray-Smith and Pearlmutter, 2005]. As per Appendix A.1, any variables over which we have a multivariate Gaussian

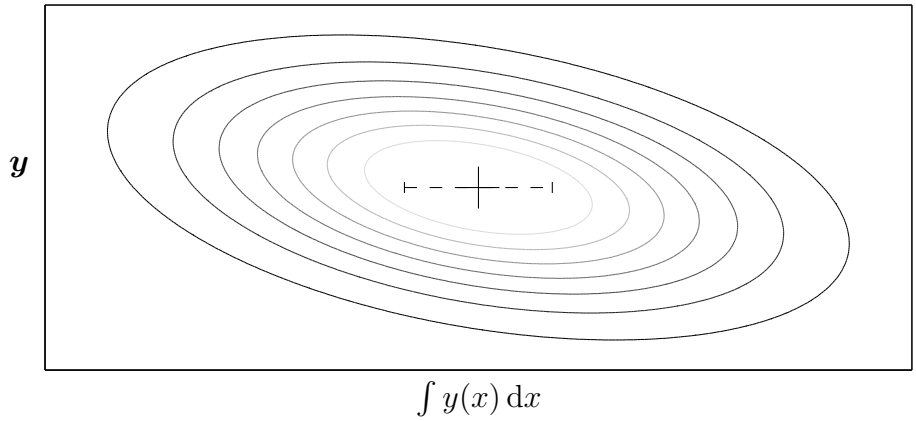


Figure 4.7: If we have a GP over function $y(x)$, any set of values $\boldsymbol{y}$ of that function are jointly Gaussian with an integral over the function, $\int y(x)\,\mathrm{d}x$.

distribution are joint Gaussian with any affine transformations of those variables. For illustrative purposes, take a multivariate Gaussian over the column vector $\boldsymbol{y} = [y_1,\, y_2,\, y_3]^\mathsf{T}$,

$$p(\,\boldsymbol{y} \mid I\,) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \mathbf{K})\ .$$

We now define the matrices

$$\mathbf{S} \triangleq \Delta \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}\ ,$$

and

$$\mathbf{D} \triangleq \frac{1}{\Delta} \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

for a 'separation' parameter Δ. Using (A.1.4), we then have a joint Gaussian over the variables resulting from the transformations $\mathbf{S}$ and $\mathbf{D}$

$$p(\,\mathbf{S}\,\boldsymbol{y},\, \boldsymbol{y},\, \mathbf{D}\,\boldsymbol{y} \mid I\,) = \mathcal{N}\left(\begin{bmatrix} \mathbf{S}\,\boldsymbol{y} \\ \boldsymbol{y} \\ \mathbf{D}\,\boldsymbol{y} \end{bmatrix}; \begin{bmatrix} \mathbf{S}\,\boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \mathbf{D}\,\boldsymbol{\mu} \end{bmatrix}, \begin{bmatrix} \mathbf{S}\,\mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{S}\,\mathbf{K} & \mathbf{S}\,\mathbf{K}\,\mathbf{D}^\mathsf{T} \\ \mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{K} & \mathbf{K}\,\mathbf{D}^\mathsf{T} \\ \mathbf{D}\,\mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{D}\,\mathbf{K} & \mathbf{D}\,\mathbf{K}\,\mathbf{D}^\mathsf{T} \end{bmatrix} \right)\ .$$

Now, imagine that $y_1 \triangleq y(x_1)$, $y_2 \triangleq y(x_1{+}\Delta)$ and $y_3 \triangleq y(x_1{+}2\Delta)$ are but three of the possible values of some (sensible) function $y(\cdot)$. We now include more variables in the vector $\boldsymbol{y} = [\ldots, y_{i-1}, y_i, y_{i+1}, \ldots]$, define Δ as the ever-diminishing separation between their inputs and inflate $\mathbf{S}$ and $\mathbf{D}$ appropriately. In such a limit, the transformed

variables $\mathbf{S}\,\boldsymbol{y}$ approach the definition of the definite (Riemann) integrals and $\mathbf{D}\,\boldsymbol{y}$, the derivatives of the function. The number of rows possessed by $\mathbf{S}$ or $\mathbf{D}$ respectively are equal to the number of integrals or derivatives we are interested in performing inference about, and the number of columns, for both matrices, is equal to the number of elements in $\boldsymbol{y}$.

Slightly more formally, imagine we have a GP over a function $y : \mathbb{R} \to \mathbb{R}$, such that over all possible function values, $\boldsymbol{y}_:$, we have

$$p(\,\boldsymbol{y}_: \mid I\,) = \mathcal{N}(\boldsymbol{y}_:; \boldsymbol{\mu}, \mathbf{K})\ .$$

Consider some vector of inputs, $\boldsymbol{x} = [x_1, \ldots, x_N]$. We re-define $\mathbf{S}$ as being the corresponding column 'vector' of integral functionals

$$\mathbf{S} \triangleq \left[\int_{-\infty}^{x_1} \cdot\, \mathrm{d}x,\ \ldots,\ \int_{-\infty}^{x_N} \cdot\, \mathrm{d}x\right]^\mathsf{T}$$

and $\mathbf{D}$ as being the column 'vector' of derivative functionals

$$\mathbf{D} \triangleq \left[\frac{\mathrm{d}}{\mathrm{d}x} \cdot \Big|_{x=x_1},\ \ldots,\ \frac{\mathrm{d}}{\mathrm{d}x} \cdot \Big|_{x=x_N},\ \ldots\right]^\mathsf{T}$$

Essentially, we are going to exploit the linearity of integration and differentiation to use $\mathbf{S}$ and $\mathbf{D}$ as we would a linear transform. As before, $\mathbf{S}$ or $\mathbf{D}$ respectively have a number of rows equal to the number of integrals or derivatives we are interested in performing inference about, here equal to N, the number of elements in $\boldsymbol{x}$. As there are an uncountably infinite number of possible elements of $\boldsymbol{y}_:$, $\mathbf{S}$ and $\mathbf{D}$ do not have columns as such; instead, each row is a functional, acting over the entirety of $\boldsymbol{y}_:$.

The column vector of integrals

$$\mathbf{S}\,\boldsymbol{y}_: \triangleq \left[\int_{-\infty}^{x_1} y(x)\,\mathrm{d}x,\ \ldots,\ \int_{-\infty}^{x_N} y(x)\,\mathrm{d}x\right]^\mathsf{T}$$

and derivatives

$$\mathbf{D}\,\boldsymbol{y}_: \triangleq \left[\frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{x=x_1},\ \ldots,\ \frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{x=x_N}\right]^\mathsf{T},$$

are, then, all joint Gaussian with the vector of all function values, $\boldsymbol{y}_:$,

$$p(\,\mathbf{S}\,\boldsymbol{y}_:,\ \boldsymbol{y}_:,\ \mathbf{D}\,\boldsymbol{y}_: \mid I\,) = \mathcal{N}\left(\begin{bmatrix}\mathbf{S}\,\boldsymbol{y}_: \\ \boldsymbol{y}_: \\ \mathbf{D}\,\boldsymbol{y}_:\end{bmatrix};\ \begin{bmatrix}\mathbf{S}\,\boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \mathbf{D}\,\boldsymbol{\mu}\end{bmatrix},\ \begin{bmatrix}\mathbf{S}\,\mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{S}\,\mathbf{K} & \mathbf{S}\,\mathbf{K}\,\mathbf{D}^\mathsf{T} \\ \mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{K} & \mathbf{K}\,\mathbf{D}^\mathsf{T} \\ \mathbf{D}\,\mathbf{K}\,\mathbf{S}^\mathsf{T} & \mathbf{D}\,\mathbf{K} & \mathbf{D}\,\mathbf{K}\,\mathbf{D}^\mathsf{T}\end{bmatrix}\right)\ .$$

As such, we can treat observations of the derivatives or integrals of a function just as we would observations of the function itself, albeit with slightly modified mean and covariance functions.

The construction of, for example, $\mathbf{D}\,\mathbf{K}\,\mathbf{S}^\mathsf{T}$, means that the derivative functionals will act on the first argument of the covariance function, and the integral functionals on the second. Explicitly, given a covariance function $K(\cdot,\cdot)$, the covariance between an observation of the derivative at x_i and an observation of the definite integral from $-\infty$ to x_j is

$$K^{(\mathrm{D,S})}(x_i, x_j) = \int_{-\infty}^{x_j} \frac{\partial}{\partial x} K(x, x') \Big|_{x=x_i} \mathrm{d}x' \,.$$

Similarly, the covariance between a derivative observation at x_i and one of the function itself at x_j is given by

$$K^{(\mathrm{D})}(x_i, x_j) = \frac{\partial}{\partial x} K(x, x_j) \Big|_{x=x_i} \,,$$

and the covariance between two derivative observations at x_i and x_j is given by

$$K^{(\mathrm{D,D})}(x_i, x_j) = \frac{\partial}{\partial x'} \frac{\partial}{\partial x} K(x, x') \Big|_{x=x_i} \Big|_{x'=x_j} \,.$$

Figure 4.8 illustrates examples of such covariance functions. Similarly, we have the covariance between an observation of the definite integral from $-\infty$ to x_i and one of the function itself at x_j

$$K^{(\mathrm{S})}(x_i, x_j) = \int_{-\infty}^{x_i} K(x, x_j) \,\mathrm{d}x \,,$$

and the covariance between an observation of the definite integral from $-\infty$ to x_i and an observation of the definite integral from $-\infty$ to x_j is given by

$$K^{(\mathrm{S,S})}(x_i, x_j) = \int_{-\infty}^{x_j} \int_{-\infty}^{x_i} K(x, x') \,\mathrm{d}x \,\mathrm{d}x' \,.$$

The same approach yields similar results for any other derivatives or integrals; all are joint Gaussian. In particular, we have joint Gaussianity for derivatives of higher order, partial derivatives of multi-input functions, and integrals with respect to other (integrable) functions.

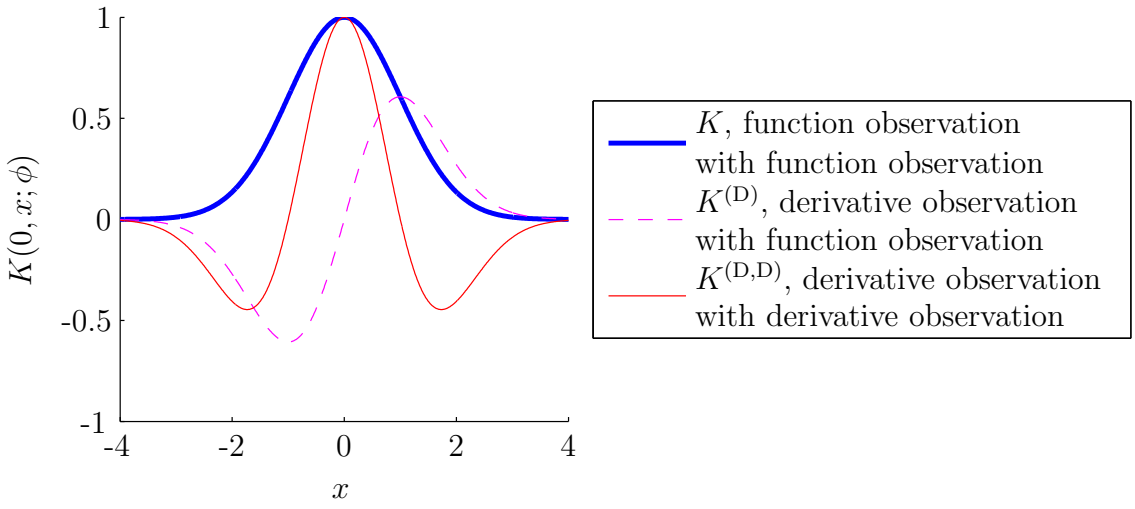


Figure 4.8: The squared exponential covariance $K^{(\mathrm{SE})}$ between observations, of either the function or its derivative, separated by x.

4.8 Metrics over sets

We now consider the challenge of constructing homogenous covariance functions to perform inference about a function defined over point sets [Garnett et al., 2010b]. To avoid confusion, we refer to such a function as an *objective function* throughout the remainder of this section. We assume that the objective function is smooth, in the sense that it will have similar values for 'related' sets.

As one motivating example, consider the objective function f that is the predictive performance of a sensor network. Here the inputs to the objective function are a set of sensor locations. For this example, we should expect *a priori* that moving each sensor by a small amount will typically not greatly affect its performance. Our ultimate goal is to use our ability perform inference over such an objective function in order to optimise it; that is, to select the optimal locations for a set of sensors. We'll return to this problem in Section 6.8.

To formalise the meaning of 'related' in this context, we define a distance between two sets of points in terms of an underlying distance between singletons. This distance can then be incorporated into an off-the-shelf covariance (3.2.1) for Gaussian process inference. Suppose we have a domain X of interest with an associated

parametrised metric $d'(x_1, x_2; \phi)$ defined on $X \times X$. We assume that the chosen metric would be useful for performing inference about the objective function if we restricted the domain to singleton sets, and extend this distance to arbitrary subsets of X. The distance d' can be readily obtained for sets of spatial coordinates; for example, the usual Euclidean distance (or great-circle distance on the earth) usually suffices.

If we additionally have a Gaussian process for a relevant field over X (which is, for example, measured by a sensor network), we have several more options for the distance d'. In such a case, the underlying prediction Gaussian process will have a parametrised covariance $K'(x_1, x_2; \phi)$ defined on $X \times X$. If known, such a covariance encapsulates much information about the unknown field, information that could be employed in our determination of the distance between sets. This might simply involve the sharing of a characteristic length scale between K' and d'. If the covariance is more sophisticated, reflecting, for example, periodicity or changepoints in our field, we might instead build a pairwise distance as

$$d'(x_1, x_2; \phi) \triangleq \sqrt{K'(x_1, x_1; \phi) + K'(x_2, x_2; \phi) - 2K'(x_1, x_2; \phi)}\,.$$

Returning to the general problem, we seek to define a parametrised metric $d(A, B; \phi)$ for $A, B \subseteq X$. We write $A \triangleq \{a_i;\, i = 1, \ldots, m\}$ and $B \triangleq \{b_j;\, j = 1, \ldots, n\}$.

This metric between sets should clearly be selected as appropriate for the task at hand. We will introduce a metric suitable for constructing covariance functions for objective functions similar to those for our sensor network example, f. Before we introduce our proposed metric d, we introduce some examples, depicted in Figure 4.9, to motivate our choices.

Figure 4.9a illustrates the first intuition to which we appeal – if $d'(a_i, b_j; \phi)$ is large for all $a_i \in A$ and $b_j \in B$, we should not expect the evaluation of the objective function at A to be strongly correlated with its evaluation at B. In such a case, $d(A, B; \phi)$ should be large.

Next, Figure 4.9b illustrates the case in which $d'(a_i, b_j; \phi)$ is large for a single

$b_j \in B$ and every $a_i \in A$. In this case, we also want $d(A, B; \phi)$ to be somewhat large. That single b_j may be in a location such that $f(B)$ will be significantly different from $f(A)$.

Figure 4.9c has, for every $a_i \in A$, $d'(a_i, b_j; \phi)$ small for some $b_j \in B$. In such a case, $d(A, B; \phi)$ should be small, because A and B, despite relabeling, have very similar locations overall (according to d'). For determining the distances between A and B in this case, the distance should depend solely on the proximity of each point in A to the *closest* point in B. It is additionally clear that any permutation of sensor labels should never be important.

It seems reasonable that adding a point very close to an existing point should not dramatically change the performance of a set. Figure 4.9d illustrates the sets from Figure 4.9c with a_3 added close to a_2 and b_3 added close to b_2. In such a case, the distance between A and B should remain close to that from Figure 4.9c.

Finally we attempt to draw out our intuitions for how the distance should behave over sets of unequal size. Figure 4.9e has $d'(a_1, b_j; \phi)$ large for all $b_j \in B$, and $d'(b_i, b_j; \phi)$ small for all b_i, $b_j \in B$. In this case,

$$d(\{a_1\}, \{b_1\}; \phi) \simeq d(\{a_1\}, \{b_1,\, b_2\}; \phi)\,.$$

Given that b_1 and b_2 are very close, the addition of b_2 should not dramatically change the distance between A and B.

We now propose a metric that satisfies these desiderata.

The metric to which we appeal is the earth mover's distance (EMD), which is well-known and widely used in image processing [Rubner et al., 2000, Levina and Bickel, 2001]. The earth mover's distance is defined for two 'signatures' of the form $\big\{(x_i, w_i)\big\}$, where the $\{x_i\}$ are points in space (in our case, points on the sphere S^2), and the $w_i \in \mathbb{R}^+$ are positive real weights. When the total weight of each signature sums to unity (that is, each signature represents a discrete probability distribution), the EMD is equivalent to the first Wasserstein or Mallows distance [Levina and Bickel, 2001]. We will assume henceforth that each signature normalises in this manner.

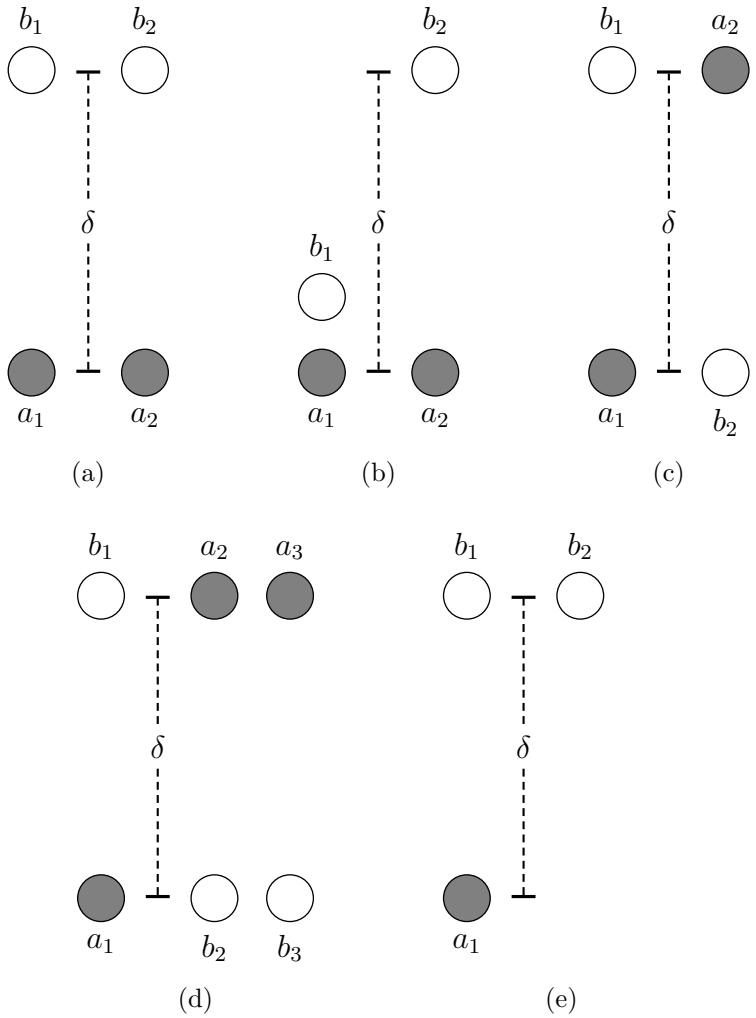
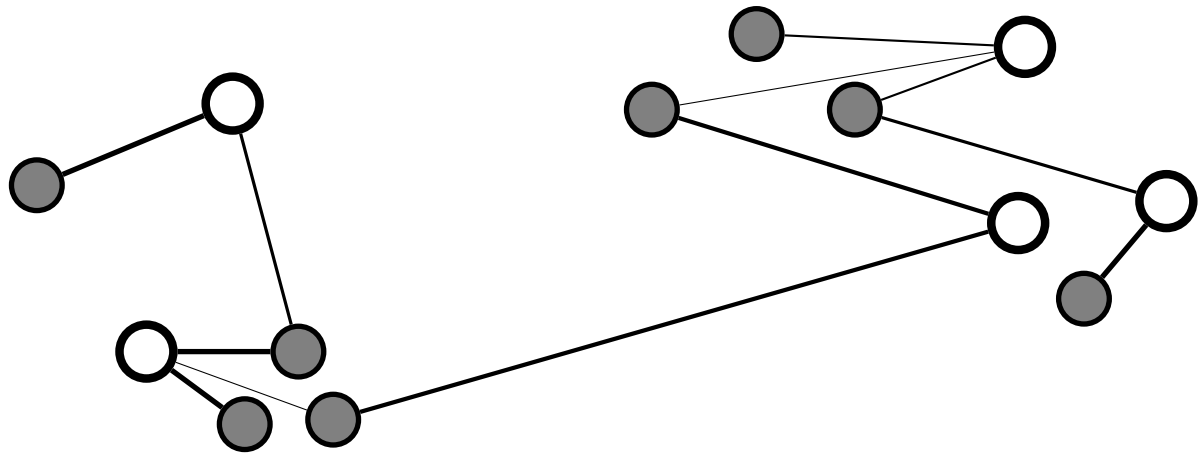


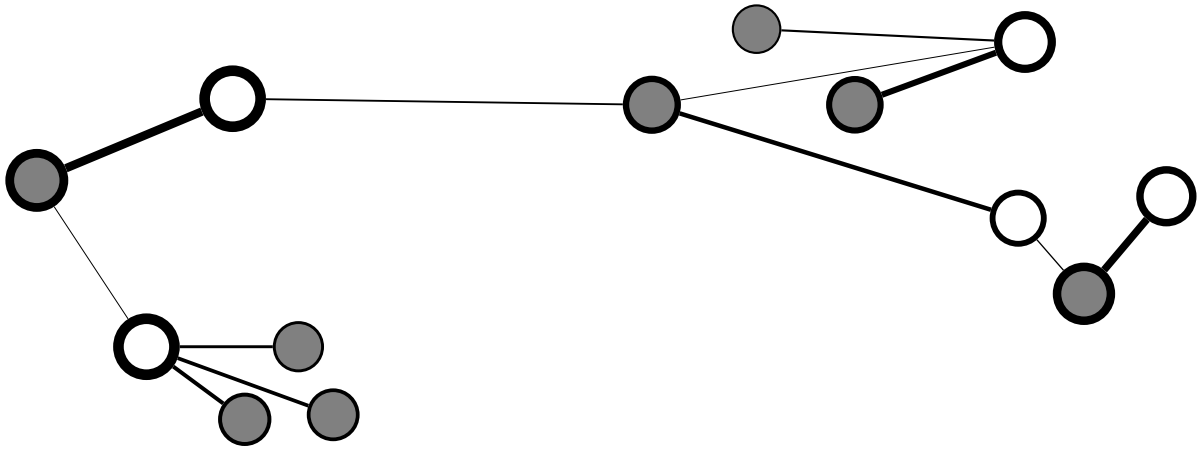
Figure 4.9: For δ sufficiently large, the distance $d(A, B; \phi)$ between sets A and B is (a) large, (b) somewhat large, (c) small, (d) small and (e) large.

Given two signatures $A \triangleq \left\{(a_i, w_i)\right\}_{i=1}^{m}$ and $B \triangleq \left\{(b_j, v_j)\right\}_{j=1}^{n}$, the EMD is defined intuitively as the minimum amount of work required to move the points in A to be aligned with the points in B. Here the amount of work for moving a point is proportional to its weight and the distance traveled. Considering each signature as a collection of mounds of dirt, the EMD is the minimum amount of earth (times distance) one would have to move to transform one signature into the other, hence its name.

More formally, the EMD is the optimum of the following linear program [Rubner et al.,



(a)



(b)

Figure 4.10: An example of the EMD between two sets, differentiated by their shading. The distance is the sum of the lengths of the depicted node connections, multiplied by the edge weighting, indicated here by the line-width of the edge. These edge weights are determined with reference to the node weights, similarly indicated by the linewidth of the node. For (a), the total (unit) weight for each set is evenly split between all nodes. For (b), the weight for each node is assigned according to its importance, using (4.8.2). Note that nodes that might be expected to be of greater importance to a sensor network, those that are far separated from other sensors in the network, are assigned greater weight.

2000]:

$$d^{(\mathrm{EMD})}(A, B) \triangleq \min \sum_i \sum_j f_{ij}\, d'(a_i, b_j) \tag{4.8.1}$$

subject to the following constraints

$$\begin{aligned} f_{ij} &\geq 0 && (1 \leq i \leq m, 1 \leq j \leq n) \\ \sum_j f_{ij} &= w_i && (1 \leq i \leq m) \\ \sum_i f_{ij} &= v_j && (1 \leq j \leq n) \\ \sum_i \sum_j f_{ij} &= 1, \end{aligned}$$

where $d'(a_i, b_j)$ is the underlying pairwise distance discussed previously. The solution to this optimisation problem can be found in polynomial time ($O(n^3)$ when $|A| = |B| = n$) using the well-known Hungarian algorithm for transportation problems.

In the case of sensor networks, corresponding two sets of sensors to the 'signatures' discussed above is mostly straightforward, except the assignment of weights to each point. Naïvely we might try a simple uniform weighting for each point in each set. The resulting distance achieves nearly all the heuristic goals set forth in the previous section. In particular, the uniformly weighted earth mover's distance has the desired behavior for Figures 4.9a, 4.9b, 4.9c, and 4.9e. Figure 4.9d presents a problem, however: using a uniform weighting, the optimal flow pairs the two closest opposite-colored points at the top, b_1 and a_2, and similarly the two closest opposite-colored points at the bottom, a_1 and b_2. The remaining two points, a_3 and b_3, are then compelled to share an edge of length δ, and the overall distance will be approximately $\delta/3$, assuming the vertical distance dominates the horizontal. This behavior is undesirable; simply by increasing the vertical distance, we can arbitrarily increase the distance between these two sets, but their performance should be expected to be similar for any distance δ.

In this case, of course, not all sensors are created equal. Assuming sufficient range, and isotropy, the sensors a_2 and a_3 will have significant overlap, as will b_2 and

b_3. In the limit where a_2 and a_3 lie completely on top of each other, they might as well be a single sensor, as one has been made completely redundant. With this insight, we modify the weights for the two signatures, such that the nearby sensors a_2 and a_3 share nearly half the total weight of the A set (as they, combined, act essentially as a single sensor), and b_2 and b_3 half the total weight of the B set. With this modification, the EMD will now pair both a_2 and a_3 with b_1, both sets now each having nearly the same weight. The overall distance will now be small (and in the limit not depend on δ), as expected.

Depending on the nature of the sensors and the prediction scheme used, any number of weighting schemes taking this notion of redundancy into account could be used; see, for example, [Gao et al., 2003] for a discussion of redundancy in sensor networks with identical isotropic sensors. The chosen scheme might also incorporate further information about the range of each sensor; for example, if sensors a_2 and a_3 were actually directional sensors pointing away from each other, it would be inappropriate to reduce their weighting as discussed above as they now have very little overlap, despite their proximity.

We offer an appropriate weighting scheme when the underlying prediction algorithm is implemented using a Gaussian process, with covariance K'. We imagine using observations $\boldsymbol{z}_A$ of that field taken at at locations $\boldsymbol{x}_A$ given by our set elements A, in order to make predictions about the field elsewhere, $\boldsymbol{x}_\star$. The mean prediction is then given by that in (3.3.2), a weighted sum of those observations,

$$\boldsymbol{\mu}(\boldsymbol{x}_\star) + \mathbf{K}'(\boldsymbol{x}_\star, \boldsymbol{x}_A)\mathbf{V}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}(\boldsymbol{z}_A - \boldsymbol{\mu}(\boldsymbol{x}_A))\,.$$

With this in mind, we take the weights associated with those observations as weights on their locations, the set elements themselves. Rather than choosing a single set of observations, we will average over all possible $\boldsymbol{x}_\star$, covering every point in the domain. By an appeal to symmetry, integrating over the entire (infinite) domain forces the average value $\mathbf{K}'(\boldsymbol{x}_\star, \boldsymbol{x}_a)$ to be identical for every element $a \in A$. Therefore we take

the weights on the elements of A

$$\boldsymbol{w}_A = \frac{\mathbf{1}_A^\top \mathbf{K}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}}{\mathbf{1}_A^\top \mathbf{K}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1} \mathbf{1}_A}, \tag{4.8.2}$$

where $\mathbf{1}_A$ is a column vector of appropriate size whose every element is equal to one. Note that we have normalised our weights to unity.

These weights, the average weights obtained in making predictions given $\boldsymbol{x}_A$, naturally reflect the behaviour we require. Elements far away from all others will receive a large quantity of weight, whereas tight clusters of elements will essentially have the weight of a single element at that position shared amongst them. We use exclusively the EMD distance that results from the use of this weighting.

Figure 4.10 demonstrates the difference between the unweighted and weighted EMD distances. In the weighted version, clusters of nearby sensors receive appropriate weights, and the overall distance for the weighted version is more natural than the unweighted version, which is forced to resort to incorporate long, highly weighted, awkward edges.

Chapter 5

Gaussian Processes for Practical, Online Inference

5.1 Conditioning

Thus far, we have presented theory that will allow us to perform Gaussian process inference for a variety of different problems. However, implementing such theory in practice will usually require some care. In particular, the implementation of our GP posterior equations, (3.3.6) and (4.5.2), is particularly sensitive to the condition number of the covariance matrix $\mathbf{V}$ – this is the ratio of its largest eigenvalue to its smallest. Unfortunately, a smooth covariance is particularly prone to poor conditioning. Smoothness requires that two nearby points are strongly correlated, and so the two rows/columns in the covariance matrix corresponding to those points will both be very similar. If we imagine those two points becoming progressively closer, the matrix will consequently tend towards singularity. This problem becomes particularly acute in the otherwise desirable case that we have a large, finely separated set of almost noiseless data.

One way to counterract this problem is to tell the GP that there is more noise than there actually is, by adding a small positive quantity known as *jitter* [Neal, 1997] to the diagonals of $\mathbf{V}$. Alternatively, we can use a less smooth covariance function – this gives another reason for the use of the flexibly smooth Matérn covariance. Of course, we could choose to discard data as necessary to obtain a data set that is well-

separated as adjudged by the input scale $\mathbf{w}$. Use of a Moore–Penrose pseudoinverse, in which small singular values are zeroed in the singular value decomposition, is also possible, but is prohibitively slow to compute. These options will all artificially decrease the condition number and reduce associated errors.

5.1.1 Cholesky factors

However, easily the most important action to take in reducing the consequences of poor conditioning is to avoid the explicit computation of the matrix inverse. A vastly more stable alternative is the computation of the Cholesky decomposition $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)$ of $\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d) = \mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^\mathsf{T} \mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)$. We denote the Cholesky operation as $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d) = \text{chol}\big(\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d)\big)$. This upper triangular factor can then be used to efficiently solve our required triangular sets of linear equations . For $\mathbf{A}$ triangular, we define $\boldsymbol{y} = \mathbf{A} \setminus \boldsymbol{b}$ as the solution to the equations $\mathbf{A}\,\boldsymbol{y} = \boldsymbol{b}$ as found by the use of backwards or forwards substitution. Note that this operation must be performed twice for (3.3.6), to give two terms we define as

$$\boldsymbol{\mathfrak{d}}_d \triangleq \mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^\mathsf{T} \setminus \big(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d)\big) \tag{5.1.1}$$

$$\boldsymbol{\mathfrak{K}}_{\star,d} \triangleq \mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^\mathsf{T} \setminus \mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_\star)\,. \tag{5.1.2}$$

Rewriting (3.3.6),

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{y}_d,\, \phi,\, I\,) = \mathcal{N}\big(\boldsymbol{y}_\star; \boldsymbol{\mu}(\boldsymbol{x}_\star) + \boldsymbol{\mathfrak{K}}_{\star,d}^\mathsf{T}\, \boldsymbol{\mathfrak{d}}_d, \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star) - \boldsymbol{\mathfrak{K}}_{\star,d}^\mathsf{T}\, \boldsymbol{\mathfrak{K}}_{\star,d}\big)\,.$$

Clearly, (4.5.2) can be rewritten using similar terms.

5.1.2 Derivative observations

Covariance matrices over derivative observations are typically better conditioned than those over observations of the function itself. Figure 4.8 makes clear that, under a squared exponential covariance, a derivative is only weakly correlated with the function very nearby, sidestepping conditioning problems due to close observations. It does, however, provide useful information about the function at remoter locations

around $r = \pm 1$. Relative to function observations, derivative observations are more weakly correlated with each other, meaning derivatives can be observed closer to each other at the same conditioning "cost." Similar conclusions are obtained under similarly shaped covariance functions, such as the rational quadratic and Matérn classes.

Roughly, these covariance functions assume that our function will be wiggling between plus-and-minus a few output scales around the mean, and so any observation of the function is reasonably informative about other nearby points on the function. This wiggling, however, can see the slope of the function changing very abruptly from negative to positive, and by a large amount. This means that a derivative observation is not particularly strongly correlated with the derivative at nearby points – in fact, it is quite likely to be anti-correlated. Neither is it particularly strongly correlated with nearby values of the function; of course, the derivative at a particular point is entirely independent of the value of the function at that same point.

Given this fact, our covariance matrix would be better conditioned if we had taken derivative observations in the place of function observations [Osborne et al., 2009]. We take this as the inspiration for a heuristic to improve conditioning. Whenever we take an function evaluation (x_d, y_d) that is closer than δ to any existing observation (x_+, y_+), we simply do not incorporate it into our covariance matrix. In its place, we include a directional derivative observation that forces the slope at the midpoint of (x_d, y_d) and (x_+, y_+) to be that of the line that connects them. That is, our new derivative observation is $(\frac{1}{2}(x_+ + x_d), y_+ - y_d)$ along the direction $x_+ - x_d$. Given that x_+ and x_d were close, and that the GP retains knowledge of y_+, we can expect to lose very little information in making this linear approximation. Of course, we do not discard y_d altogether – it can be used for the purposes of computing future derivative observations, as illustrated in Figure 5.1. y_d also forms part of $\boldsymbol{y}_0$, used for the purposes of computing η, for example. This use of derivative observations in the place of function observations sacrifices very little information, while giving rise to a much better conditioned algorithm.

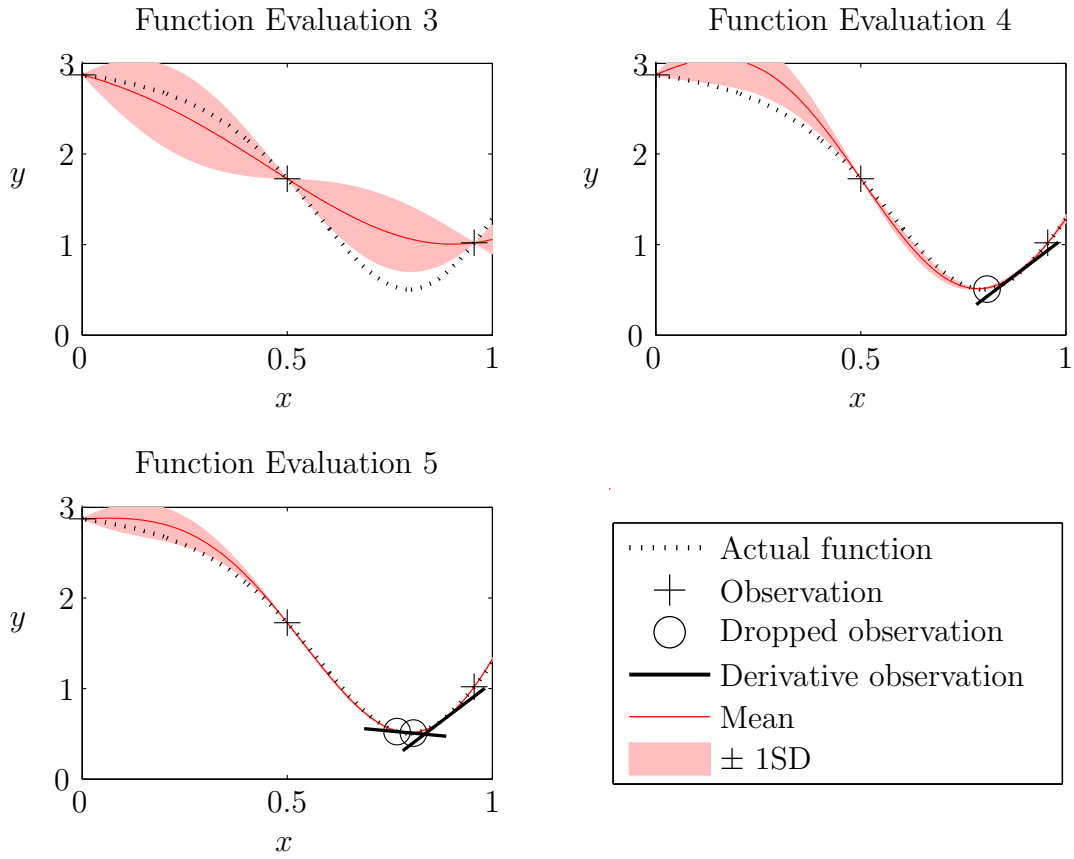


Figure 5.1: An illustration of the use of derivative observations as a replacement for the very close function observations required in locating a minimum. Both function evaluation 4 and 5 are replaced by appropriately calculated derivative observations.

5.2 Updating Gaussian Processes

In many real problems, we do not have a fixed data set. Rather, we receive data one or a few observations at a time, the total information available to us perpetually growing incrementally. Moreover, updates are often required every hour, minute or even second – time is critical. Using the above methods as is for this case would rapidly incur prohibitive computational overheads. Each time we wish to update our beliefs in the light of a new observation, we would be forced to perform the expensive Cholesky decomposition (the complexity of which grows as $O(\frac{1}{3}n^3)$ in the dimension n of the matrix) of an ever-bigger $\mathbf{V}$ matrix. Fortunately, there are a number of tricks we can employ to improve the efficiency of the iterative updates [Osborne et al., 2008, 2010b].

The following proposals all exploit the special structure of this problem. When we receive new data, our $\mathbf{V}$ matrix is changed only in the addition of a few new rows and columns. Hence most of the work that went into computing its Cholesky decomposition at the last iteration can be recycled, if we are careful. To reflect the iterative nature of our data, we denote by x_n the location (input) of the nth datum, and by $\boldsymbol{x}_{1:n}$ the locations (inputs) of all data obtained between the first and nth iterations, inclusively.

Given $\mathbf{R}(\boldsymbol{x}_{1:n-1}, \boldsymbol{x}_{1:n-1})$, then, we can use the procedures described in Appendix B.1 to efficiently determine $\mathbf{R}(\boldsymbol{x}_{1:n}, \boldsymbol{x}_{1:n})$. Similarly, given $\mathfrak{d}_{1:n-1}$, Appendix B.2 describes how to find $\mathfrak{d}_{1:n}$. Note that in a sequential prediction context, $\mathfrak{K}_{\star,1:n}$ is not suitable for such rules. This is due to its dependence on the selection of predictants, denoted by the $\star$. In prediction and tracking, the objects of our interest will be constantly shifting, implying no recycling is possible.

So, given these update rules, it's time to weigh the ledger to determine just what we have gained. The most expensive operations above, such as the determination of $\mathfrak{K}_{\star,1:n}$, (5.1.2), involve the solution of a set of n triangular equations for a dataset of size n, the cost of which will in general scale as $O(n^2)$. As such, we have reduced the

overall cost of an update from $O(n^3)$ to $O(n^2)$, with no loss of accuracy.

We now turn to another important term – the hyperparameter likelihoods $p(\, \boldsymbol{z}_d \mid \phi,\, I\,)$. Likelihoods are commonly smaller than machine precision and so for numerical reasons we prefer to instead work with log-likelihoods. Naturally, these will then be exponentiated whenever required. Noting that the determinant of any matrix can be expressed as the product of its eigenvalues, which for a triangular matrix are simply its diagonal elements, we can rewrite (3.1.1) to give the log-likelihood as

$$\begin{aligned}\log p(\, \boldsymbol{z}_{1:n} \mid \phi,\, I\,) &= -\frac{1}{2}\log\det\left(2\pi\,\mathbf{V}(\boldsymbol{x}_{1:n}, \boldsymbol{x}_{1:n})\right) - \frac{1}{2}\boldsymbol{\mathfrak{d}}_{1:n}^{\mathsf{T}}\boldsymbol{\mathfrak{d}}_{1:n} \\ &= -\operatorname{tr}\log\left(\sqrt{2\pi}\,\mathbf{R}(\boldsymbol{x}_{1:n}, \boldsymbol{x}_{1:n})\right) - \frac{1}{2}\boldsymbol{\mathfrak{d}}_{1:n}^{\mathsf{T}}\boldsymbol{\mathfrak{d}}_{1:n} \qquad (5.2.1)\end{aligned}$$

Hence it can be seen that an update of the log-likelihood of our hyperparameters is also possible, as per Appendix B.3. As per (3.4.8), these updated log-likelihoods then give rise to updated weights ρ over our hyperparameter samples.

These rules hold for any arbitrary covariance. However, further savings are possible if we take certain special covariance functions. The prototypical example is the Kalman Filter covariance, as per Appendix C.

5.3 Discarding Data

In practice, a GP requires very small amounts of data to produce good estimates. In the context of prediction, often only several close data points are necessary to produce reasonable estimates for the predictants. One reason for this can be seen in the very light tails of the commonly used covariance functions displayed in Figure 3.4. Hence only points within a handful of input scales will prove at all relevant to the point of interest.

Further, reducing the size of our data set is desirable for a number of reasons. As mentioned in Section 5.1, a smaller data set produces a better conditioned covariance and can be expected to give rise to smaller associated errors. Secondly, despite our efforts in Section 5.2, the cost of an update remained on the order of $O(n^2)$. While

this certainly represents an improvement, given a finite computer, it still imposes a distinctly finite limit on the number of time steps we can possibly consider. Clearly this is unsatisfactory – we'd like our prediction algorithms to be capable of running more or less indefinitely. Finally, our finite computer will also only have finite memory; as such we have a certain limit to the amount of data we can store, regardless of any wishes otherwise. Hence, in the context of sequential prediction, we seek sensible ways of discarding information once it has been rendered 'stale'.

One pre-eminently reasonable measure of the value of data is the uncertainty we still possess after learning it. In particular, we are interested in how uncertain we are about $\boldsymbol{y}_\star$, as given by the covariance of our Gaussian mixture (A.1.8). One approach is to take a loss function that induces the algorithm to store as little data as possible while still maintaining the uncertainty below some pre-specified threshold. That is, our loss is infinite if the uncertainty is greater than the threshold, and zero otherwise. We also have some fixed loss associated with each data point, such that we are induced to try and minimise the size of our dataset.

We now face the challenge of how to select which data to drop so as to minimise this loss. In considering how the uncertainty of (A.1.8), and hence how (3.3.8), varies with data $\boldsymbol{x}_d$, we find that the only terms of significance are of the form $K(\boldsymbol{x}_\star, \boldsymbol{x}_d)\, V(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}\, K(\boldsymbol{x}_d, \boldsymbol{x}_\star)$ (which, as $V(\boldsymbol{x}_d, \boldsymbol{x}_d)$ is semi-positive definite, is non-negative). To keep that term large, and hence the uncertainty small, we want to maximise the covariance between our predictants $\boldsymbol{x}_\star$ and data $\boldsymbol{x}_d$. Of course, this simplistic analysis ignores the influence our weights ρ and data covariance $K(\boldsymbol{x}_d, \boldsymbol{x}_d)$ have on our uncertainty, and how those terms themselves vary with data $\boldsymbol{x}_d$. A truly sophisticated search strategy would determine which subset of data is least relevant as measured by our full uncertainty. We might even consider the long-term consequences of our actions, such as how quickly the increase in uncertainty due to dropped data is likely to be ameliorated by the addition of future data. For simplicity, however, we limit ourselves to discarding data that is poorly correlated with the current predictants.

The covariances over time we usually consider will be of the form (3.2.1) with simple distance (3.2.2). For such covariances, correlations decrease with separation in time. In this case, we can simply discard data well removed from the current predictant until our uncertainty becomes unacceptably large. This is particularly simple for data over a single, temporal input, for which we wish to make predictions about the present or future – we drop the oldest data. For other covariances, such as those incorporating a periodic term, old data might remain pertinent, and the search for poorly correlated data becomes slightly more challenging. However, it is common that periodic terms be modulated by a 'forgetting' covariance term, such as a squared exponential. No real periodic signal is so periodic that exceedingly old data is equally as informative as new data. In such a case, dropping old data is again justified.

Let's see, then, how we can drop old data without having to completely redetermine all the quantities in our model. In effect, we now have do perform a downdate for each hyperparameter sample, the opposite of what we were trying to achieve in Section 5.2. Appendix B.4 describes the procedures required to determine $\mathbf{R}(\boldsymbol{x}_{1:n-1}, \boldsymbol{x}_{1:n-1})$ given $\mathbf{R}(\boldsymbol{x}_{1:n}, \boldsymbol{x}_{1:n})$. We'll also need to downdate $\mathfrak{d}$, but unfortunately, there is no way to do this more efficient than the direct solution

$$\mathfrak{d}_{2:n} = \mathbf{R}(\boldsymbol{t}_{2:n}, \boldsymbol{t}_{2:n})^{\mathsf{T}} \setminus (\boldsymbol{y}_{2:n} - \mu(\boldsymbol{t}_{2:n})) \tag{5.3.1}$$

Naturally there is no computational benefit here to be gained in downdating our log-likelihood, whose dimension remains constant regardless of the size of the dataset. Hence, at no cost, we can retain all knowledge of otherwise dropped data that is pertinent to our marginalisation of hyperparameters[1]. This is important as it means that as we progress, we can continue to refine our weights over the hyperparameter samples, improving our model. The somewhat glib statement made earlier to the effect that a GP needs only a few data points is true only if it has sufficient knowledge

[1]It should be noted, however, that after we have dropped a data point, all subsequent updates to the log-likelihood will be made using covariances computed over the data set without the dropped data point. Of course these covariances should embody only weakly correlations with dropped data, and so the effects of discarding data should be small.

of the hyperparameters. Fortunately, we can achieve this by simply continuing to update our log-likelihoods.

We have just seen that downdating involves a reasonable amount of effort for a single hyperparameter sample. As such, it may not be feasible to evaluate the full Gaussian mixture uncertainty (3.4.8), involving downdating for all hyperparameter samples, each time we wish to trial the discarding of a datum. A more practical approach is to compute the uncertainty in terms of only one or a few of the higher weighted samples. Once it has been determined how much data is to be dropped, a single downdate can be performed for each hyperparameter sample, a time step advanced and the prediction algorithm continued.

In conclusion, this proposed method of discarding data will allow us to retain only a manageable data set. At the same time, using a permissible threshold of uncertainty as a criterion ensures that discarding will not have a significant effect on our predictions. This allows a principled way of introducing a 'windowing' approach to a temporal data series. Of course, the same methods of downdating can be used for other measures – for example, we may wish to put in place a maximum allowable size on our data set, such that it never exceed available memory. This represents just as valid a loss function as that of using uncertainty – indeed, we are constantly forced to make decisions with constraints on how long we have available for computation. We can use these methods to ensure that the computational cost of each iteration reaches a plateau. Hence we can envisage our GP algorithm continuing indefinitely, producing good predictions within allowable time constraints.

5.4 Active Data Selection

Active data selection (ADS) is the task of selecting only the most informative observations of a system under study. Specifically, we aim to select observations that render us as certain as possible about some variable(s) of interest. As a motivating context, we consider networks of sensors that are all capable of making observations

of various correlated variables, such as the air temperatures at the different sensor locations. Our goals in this scenario are twofold: we want to select observations that minimise our uncertainty about those variables, while also minimising the required number of those observations. Note that in practice, taking observations is usually associated with some cost, such as the battery energy required to power a sensor or the computational cost associated with additional data. An observation might also be associated with an increased risk of detection, if our system's goal is to perform covert surveillance.

Consider the decision task facing the central agent controller of the network, at an arbitrary time t. There are two components to this decision. First, we must decide whether to take an observation at all at that time. If we do, we must then select a type of observation to be made at that time – if there are multiple sensors, we must select one to draw an observation from. As previously mentioned, our goals in making this decision are likewise two-fold: we aim to both minimise the uncertainty we possess about variables of interest, and also to minimise the number of observations required to be taken. We define the objects of our interest at t as $\boldsymbol{y}_\star \triangleq \{y_l,\, l = 1,\, \ldots,\, L\}$, where y_l is the value of our variable at input $[l, t + \epsilon]$, that is, at the location of sensor l for the lookahead-shifted time $t + \epsilon$. Should the uncertainty about any other points be relevant to a problem (for example, at other points we are unable to directly observe), these could also be readily incorporated into our set. We choose the sum of the standard deviations for each $\boldsymbol{y}_\star$ as our loss – we want to minimise that measure of uncertainty. We can now define the expected loss of choosing an observation from sensor l_c as

$$\begin{aligned}&\Lambda(l_c \mid \boldsymbol{z}_d,\, I)\\ &\triangleq \int \left(\sum_{l=1}^{L} \sqrt{\mathbf{C}(\mathring{y}_l \,|\, z_c,\, \boldsymbol{z}_d,\, I)} \right) p(z_c|\boldsymbol{z}_d,\, I)\, \mathrm{d}z_c + C_c\,, \end{aligned} \tag{5.4.1}$$

where we have marginalised over the possible observations z_c received from sensor l_c. This integral is non-analytic, and as such will be approximated by application of Bayesian Quadrature techniques, as will be discussed in Section 7.1.

The term C_c denotes the cost of the observation, representing a "conversion factor" between the loss associated with sensing and with the expected loss due to uncertainty. The cost represents the average increase in the standard deviation at our pertinent locations we would require before an additional observation would become worthwhile. It is trivially possible (both theoretically and computationally) to take a different C_c for different types of observation. However, the applications we consider require only a fixed cost C, and therefore we consider only this possibility.

Note that we treat the standard deviation associated with each variable equally, combining them all with a sum. As another simple extension, the various standard deviations could be weighted against one another if the monitoring of some variables was more critical than others. In some applications, we might also wish to include the cross-correlation terms of the full covariance over all objects of interest into (5.4.1).

We also have the loss associated with not taking an observation at all

$$\Lambda(\emptyset \mid \boldsymbol{z}_d,\, I) \triangleq \sum_{l=1}^{L} \sqrt{\mathbf{C}(\mathring{y}_l \,|\, \boldsymbol{z}_d,\, I)}\,. \tag{5.4.2}$$

So, defining

$$l_m \triangleq \operatorname*{argmin}_{l_c=1,\,\ldots,\,L} \Lambda\left(l_c \mid \boldsymbol{z}_d,\, I\right)\,,$$

if $\Lambda\left(l_m \mid \boldsymbol{z}_d,\, I\right) < \Lambda(\emptyset \mid \boldsymbol{z}_d,\, I)$, we sample at l_m; otherwise, we do not sample at all. Of course, it is usually not desirable to have to evaluate this policy at every time it is possible to sample. Instead, after taking an observation, we can use any optimisation algorithm to determine the future time at which a sample will next become necessary.

We will use GPs to model our variables, giving a Gaussian for $p(z_c|\boldsymbol{z}_d,\, I)$ and an analytic form for $\mathbf{C}(\mathring{y}_l \,|\, \boldsymbol{z}_d,\, I)$ (from Section 3.3). However, the loss function proposed above is applicable to active data selection with any probabilistic model.

Importantly, the variances in (5.4.1) and (5.4.2) capture our underlying uncertainty about the correct model. This is due to the principled marginalisation undertaken, as discussed in Section 3.4. In particular, in (5.4.1), the potential observation z_c may influence our beliefs about the correct model, as expressed by the weights over hyperparameter samples used when constructing $\sqrt{\mathbf{C}(\mathring{y}_l \,|\, z_c,\, \boldsymbol{z}_d,\, I)}$. For

example, the uncertainty associated with a sensor being faulty, as discussed in Section 4.5, and the uncertainty associated with the potential characteristics of that fault, all influence the variances we are trying to minimise. More appropriate observations will be scheduled if there is a significant degree of uncertainty about the model. Note also that while we make a decision at time t solely with the objective of minimising our uncertainty about the state at $t + \epsilon$, the smoothness of a GP means that our scheduled observations will also give low uncertainty for subsequent times.

It is worth noting the differences between our loss function and that used in previous approaches. In Rogers et al. [2008] and Osborne et al. [2008, 2010b], a loss function was taken that was infinite if the variance associated with any object of interest become greater than a pre-specified threshold. This is a problematic choice when our sensors are subject to faults. Should a sensor become faulty, the uncertainty associated with it is likely to increase beyond the threshold, regardless of the samples we take. Once this has happened, the algorithm will request observations from it constantly – clearly, this is undesirable behaviour. Our proposed loss function suffers from no such shortcoming.

Chapter 6

Gaussian Processes for Global Optimisation

6.1 Introduction

Optimisation has been an important area of mathematical study since the creation of the digital computer. A great deal of research has been devoted to solving dozens of sub-problems in this field [Nocedal and Wright, 2006], with application areas ranging from economics to mechanical design. In this chapter we approach global optimisation from the viewpoint of Bayesian theory, and frame the problem as a sequential decision problem. Imagine that we have an unknown and expensive-to-evaluate objective function $y(x)$ that we seek to minimise. The cost associated with computing $y(x)$ compels us to select the location of each new evaluation very carefully. For testing purposes, we reflect this cost by restricting ourselves to a limited number of function evaluations. Our problem's ultimate task is to return a final point x_m in the domain, and we define the loss associated with this choice, $\lambda(x_m)$, to be $y(x_m)$. As we aim to minimise our losses, we aim to minimise $y(x_m)$.

Building upon the proposal of Section 5.4, we will use a Gaussian process to model the objective function. Of course, the idea of using such a surrogate or *response surface* for optimisation has been well-explored [Jones, 2001]. Essentially, the goal of such an approach is to build a statistical picture of the function's overall form given our observations of it. Given the GP model of our expensive function, we then

determine where best to evaluate it in future.

This context allows us to make full use of the iterative GP framework developed in Chapter 3. After each new function evaluation, we can efficiently update our predictions in light of the new information received. Consider a step of the sequential optimisation problem, and define I_0 as the conjunction of I and whatever observations of the function we have gathered so far, $(\boldsymbol{x}_0, \boldsymbol{y}_0)$. Note that in this chapter the location of function observations will attain a greater importance, and as such we temporarily abandon our convention of taking function inputs as being always implicit within I. Given I_0, as per (3.4.8), we approximately marginalise hyperparameters by taking a weighted mixture

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{x}_\star,\, I_0\,) \simeq \sum_{i \in s} \rho_i \, \mathcal{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}(\mathring{\boldsymbol{y}}_\star | \phi_i, I_0), \mathbf{C}(\mathring{\boldsymbol{y}}_\star | \phi_i, I_0)\big)\,,$$

where the weights ρ will continue to be revised as new data is gathered.

6.2 One-step Lookahead

To begin, imagine that we have only one allowed function evaluation remaining before we must report our inferred function minimum. We constrain the returned (x_m, y_m) to the set of actual gathered points. Essentially, we require that the final reported minimum supplied by our algorithm must be known with the utmost confidence (although we will look at slightly relaxing this constraint later in Section 6.5). Suppose $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ are the function evaluations gathered thus far and define $\eta \triangleq \min \boldsymbol{y}_0$. Given this, we can define the loss of evaluating the function this last time at x and its returning y

$$\lambda(y) \triangleq \begin{cases} y; & y < \eta \\ \eta; & y \geq \eta \end{cases}. \tag{6.2.1}$$

That is, after having observed y, our loss is simply the new minimum of the set of observed points, $\min(y, \eta)$, which we would report as y_M.

Now, given our GP over y, we can determine an analytic form for the expected

loss of selecting x given that we know I_0 and have only one evaluation remaining

$$\Lambda_1(\, x \mid I_0\,) \triangleq \int \lambda(y)\, p(\, y \mid x,\, I_0\,)\, \mathrm{d}y = \sum_{i \in s} \rho_i \mathrm{V}(\, x \mid \phi_i,\, I_0\,)\ ,$$

where, denoting the usual Gaussian cumulative distribution function as Φ,

$$\begin{aligned}\mathrm{V}(\, x \mid \phi_i,\, I_0\,) &\triangleq \eta \int_{\eta}^{\infty} \mathcal{N}(y; m_i, C_i)\, \mathrm{d}y + \int_{-\infty}^{\eta} y\, \mathcal{N}(y; m_i, C_i)\, \mathrm{d}y \\ &= \eta + (m_i - \eta)\, \Phi(\eta; m_i, C_i) - C_i\, \mathcal{N}(\eta; m_i, C_i)\ . \end{aligned} \tag{6.2.2}$$

Here we have abbreviated $m(\mathring{y}|\phi_i, I_0)$ as m_i and $C(\mathring{y}|\phi_i, I_0)$ as C_i. Note that the recommendations of Jones et al. [1998] are close to but differ from this Bayesian expected loss criterion[1]. Our criteria $\mathrm{V}(\, x \mid \phi_i,\, I_0\,)$ gives the *expected minimum* after taking the next function evaluation.

The location where our expected loss is lowest gives the optimal location for our next function evaluation. Note that (6.2.2) will decrease as m_i becomes lower than η, and also as C_i increases. This first fact ensures exploitation, the second exploration. As such, the minimisation of our expected loss gives a very natural balancing of these two concerns.

Of course, we have merely shifted the minimisation problem from one over the objective function $y(x)$ to one over the expected loss function $\Lambda_1(\, x \mid I_0\,)$. Fortunately, the expected loss function is continuous and computationally inexpensive to evaluate, as are the analytic expressions for its gradient and Hessian. We hence have a range of routines suitable for this minimisation.

6.3 Multi-step Lookahead

Note that the situation described in Section 6.2 can also be used as a myopic approximation to the case in which we actually have many evaluations remaining. Essentially, this means we assume that we are so uncertain about how this current decision will affect our future decisions that we can ignore them. Put another way, given our great

[1] Their stated expression is $\eta + (m_i - \eta)\, \Phi(\eta; m_i, C_i) - \sqrt{C_i}\, \mathcal{N}(\eta; m_i, C_i)$, which is dimensionally invalid.

uncertainty about function evaluations made in the far future (we are not even sure where they will be made), we ignore them and attempt instead to minimise over simply the very next function evaluation. Such an approximation typically gives good results, as we'll see later.

Nonetheless, proper probabilistic reasoning will allow us to relax this short-sighted assumption. We define our successive remaining function evaluations as $\{(\boldsymbol{x}_j, \boldsymbol{y}_j) : j = 1, \ldots, n\}$. The totality of our information up to and including the jth evaluation we denote using I_j, the conjunction of I_0 and $\{(\boldsymbol{x}_{j'}, \boldsymbol{y}_{j'}) : j' = 1, \ldots, j\}$. In general, we use $\Lambda_n(\, x_1 \mid I_0\,)$ to denote the expected loss of selecting x_1 given that we know I_0, considering n evaluations into the future:

$$
\begin{aligned}
&\Lambda_n(\, x_1 \mid I_0\,) \\
&\triangleq \int \lambda(y_n)\, p(y_n|x_n, \phi, I_{n-1})\, p(x_n|I_{n-1}) \ldots p(y_2|x_2, \phi, I_1)\, p(x_2|I_1)\, p(y_1|x_1, \phi, I_0) \\
&\qquad \mathrm{d}y_1 \ldots \mathrm{d}y_n \mathrm{d}x_2 \ldots \mathrm{d}x_n\, p(\boldsymbol{y}_0|\boldsymbol{x}_0, \phi, I)\, p(\phi|I)\, \mathrm{d}\phi \Big/ \int p(\boldsymbol{y}_0|\boldsymbol{x}_0, \phi, I)\, p(\phi|I)\, \mathrm{d}\phi \\
&\qquad\quad = \sum_{i \in s} \rho_i \int \mathrm{V}(\, x_n \mid \phi_i,\, I_{n-1}\,)\; \delta(x_n - \mathrm{argmin}_x \Lambda_1(\, x \mid I_{n-1}\,)) \ldots \\
&\qquad\qquad\quad \mathcal{N}\big(y_2; m(\mathring{y}_2|\phi_i, I_1), C(\mathring{y}_2|\phi_i, I_1)\big)\; \delta(x_2 - \mathrm{argmin}_x \Lambda_{n-1}(\, x \mid I_1\,)) \\
&\qquad\qquad\qquad \mathcal{N}\big(y_1; m(\mathring{y}_1|\phi_i, I_0), C(\mathring{y}_1|\phi_i, I_0)\big) \mathrm{d}y_1 \ldots \mathrm{d}y_{n-1} \mathrm{d}x_2 \ldots \mathrm{d}x_n\,. \qquad (6.3.1)
\end{aligned}
$$

The probabilistic model underlying (6.3.1) is illustrated in Figure 6.2, and represented in pseudocode in Algorithm 1. To evaluate (6.3.1), we successively sample y_1 through y_{n-1} from their respective Gaussians, $\mathcal{N}\big(y_1; m(\mathring{y}_1|\phi_i, I_0), C(\mathring{y}_1|\phi_i, I_0)\big)$ through $\mathcal{N}\big(y_{n-1}; m(\mathring{y}_{n-1}|\phi_i, I_{n-2}), C(\mathring{y}_{n-1}|\phi_i, I_{n-2})\big)$.
Given each, we can then determine the best choice for the next sample location by minimising the appropriate expected loss function $\Lambda_{n-j}(\, x_{j+1} \mid I_j\,)$, for $j = 1, \ldots, n-1$. Note these loss functions are themselves integrals over ϕ. These will be approximated as weighted sums over hyperparameter samples ϕ_i, with updated weights that will differ from the ρ_i given the differing sets of information I_j. Unfortunately, only for $\Lambda_1(\, x_n \mid I_{n-1}\,)$ can this required minimisation benefit from analytic derivatives. Given this fact, and the degree of sampling required, it is clear that multi-step lookahead

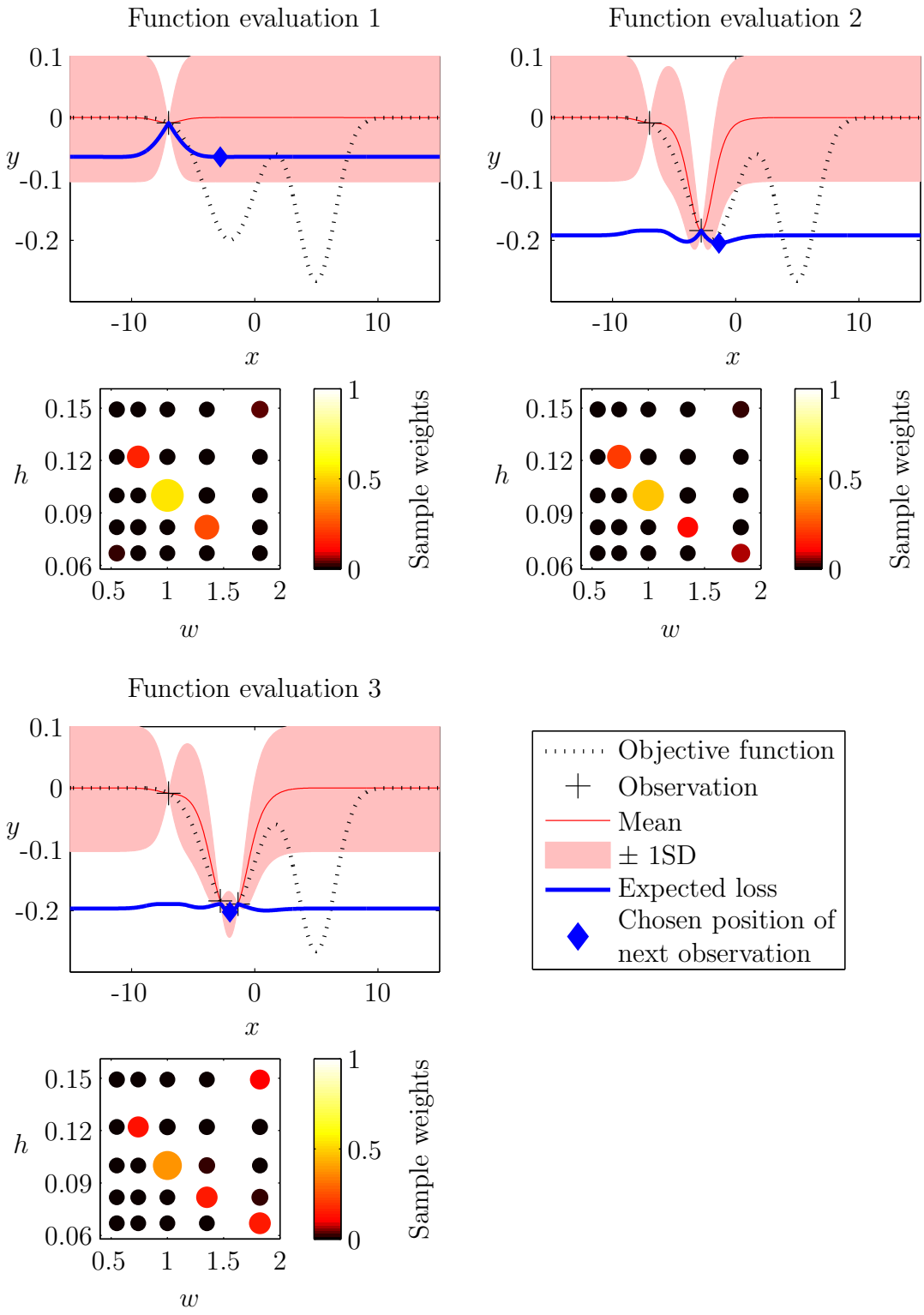
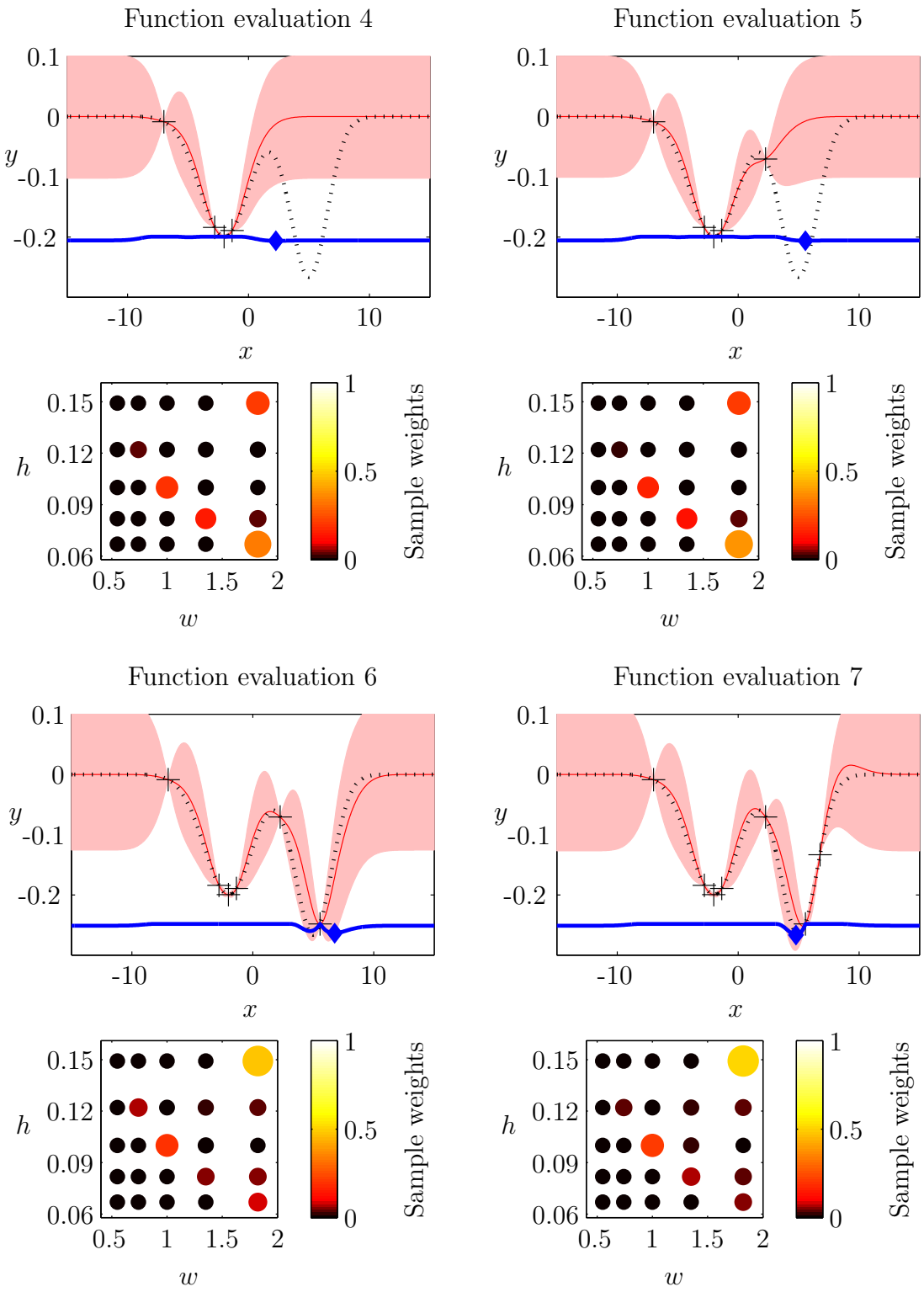


Figure 6.1: An illustration of our algorithm balancing the exploitation of the current local minimum and exploration of unvisited areas. The chosen covariance was the squared exponential (3.2.3), giving us two scale hyperparameters, w and h. The weights associated with the samples over those hyperparameters are indicated...



Continued Figure 6.1: ...both by the shading and the radii of the circles. Weight is transferred from the prior mean, at $w = 1$ and $h = 0.1$, towards larger and more representative scales. This gives rise to a smoother GP fit with larger error bars, which then influences the selection of future evaluations.

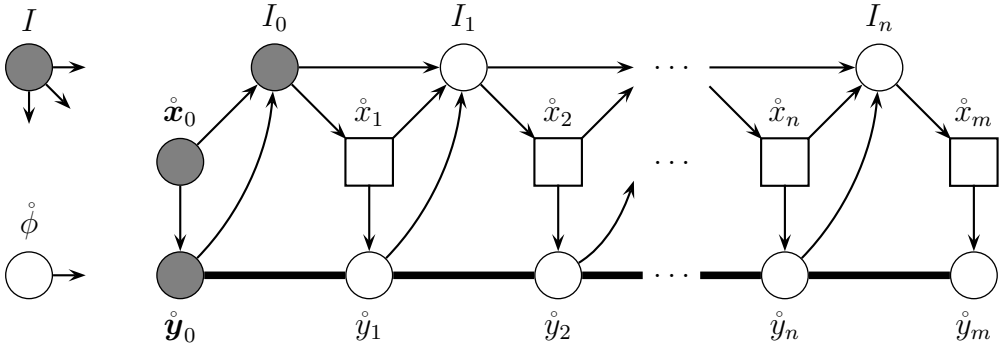


Figure 6.2: A Bayesian network for the optimisation problem. Shaded nodes are known, and square nodes represent the results of a decision. All $\mathring{y}$ nodes are correlated with one another, $\mathring{\phi}$ is correlated with all $\mathring{y}$ nodes and the context I is correlated with all nodes.

is a more computationally demanding exercise. However, this cost can be justified if the function $y(x)$ we aim to minimise is itself even more computationally expensive. We effectively need merely to select the n appropriate for the problem at hand.

6.4 Conditioning Problems

As discussed in 5.1, a common problem with GPs is the poor conditioning of covariance matrices. This occurs when we have observations that are strongly correlated (giving multiple rows/columns in the covariance matrix that are very similar) such as is the case for highly proximate observations. This is a particular concern for optimisation, in which we commonly want to take many observations close to a minimum. Of course, this problem is ameliorated slightly by our procedure, which recognizes that sampling close to an existing observation rarely produces much new information. The problem is also less severe in high dimension, in which there is volume enough to avoid existing observations.

In order to improve conditioning, we firstly make use of Cholesky factors, as per Section 5.1.1. We also ensure that each new function observation is sufficiently far from all existing function observations; the separation δ we enforce is a single input scale. If our procedure stipulates an observation closer than this, we instead take an appropriate derivative observation, as described in Section 5.1.2. Figure 5.1 gives an

Algorithm 1 GPGO-Expected-Loss$(x_1, \boldsymbol{x}_0, \boldsymbol{y}_0, n, \boldsymbol{\phi}_s)$

// Returns the expected loss $\Lambda_n(\, x_1 \mid I_0\,)$ of evaluating the objective function at x_1.
// x_1: Potential next evaluation location.
// $\{(\boldsymbol{x}_0, \boldsymbol{y}_0)\}$: Initial data.
// n: Lookahead. Only the cases $n \in \{1, 2\}$ are considered below.
// $\boldsymbol{\phi}_s \triangleq \{\phi_i\}_{i \in s}$: Hyperparameter samples.

$\boldsymbol{\rho}_s \leftarrow$ Determine-Hypersample-Weights$(\boldsymbol{\phi}_s, \boldsymbol{x}_0, \boldsymbol{y}_0)$
// We will discuss in Section 7.3 how to determine the weights over our hyperparameter samples, $\boldsymbol{\phi}_s$.
$\Lambda_n(\, x_1 \mid I_0\,) \leftarrow 0$.
for $i \in s$ **do**
  if $n = 1$ **then**
    $\mathrm{V}_n(\, x_1 \mid \phi_i,\, I_0\,) \leftarrow \mathrm{V}(\, x_1 \mid \phi_i,\, I_0\,)$ // $\mathrm{V}(\, x_1 \mid \phi_i,\, I_0\,)$ as in (6.2.2).
  else if $n = 2$ **then**
    $\mathrm{V}_n(\, x_1 \mid \phi_i,\, I_0\,) \leftarrow 0$.
    $\boldsymbol{y}_{1,T} \leftarrow$ Sample-Gaussian$(x_1, \boldsymbol{x}_0, \boldsymbol{y}_0, \phi_i)$ // Take samples (perhaps at the quantiles) of the Gaussian $\mathcal{N}\big(\cdot; m(\mathring{y}_1|\phi_i, I_0), C(\mathring{y}_1|\phi_i, I_0)\big)$.
    $\boldsymbol{w}_{1,T} \leftarrow$Determine-Sample-Weights$(\boldsymbol{y}_{1,T}, x_1, \boldsymbol{x}_0, \boldsymbol{y}_0, \phi_i)$
    // We will discuss in Section 7.1 how to determine the weights over our sample values for y_1, $\boldsymbol{y}_{1,t}$.
    for $t \in T$ **do**
      $x_2 \leftarrow \mathrm{argmin}_{x_2}$ GPGO-Expected-Loss$(x_2, (\boldsymbol{x}_0, x_1), (\boldsymbol{y}_0, y_{1,t}), 1, \boldsymbol{\phi}_s)$
      // A simple optimisation routine is used to evaluate this argmin.
      $\mathrm{V}_n(\, x_1 \mid \phi_i,\, I_0\,) \leftarrow \mathrm{V}_n(\, x_1 \mid \phi_i,\, I_0\,) +$
        $w_{1,t}$ GPGO-Expected-Loss$(x_2, (\boldsymbol{x}_0, x_1), (\boldsymbol{y}_0, y_{1,t}), 1, \boldsymbol{\phi}_s)$.
    end for
  else
    ... // This case is not explicitly considered here.
  end if
  $\Lambda_n(\, x_1 \mid I_0\,) \leftarrow \Lambda_n(\, x_1 \mid I_0\,) + \rho_i \mathrm{V}_n(\, x_1 \mid \phi_i,\, I_0\,)$
end for
Return: $\Lambda_n(\, x_1 \mid I_0\,)$

example of the GPGO algorithm operating with such a heuristic.

Note that our method is equally capable of exploiting observations of derivatives or anti-derivatives obtained by any other means. Such an approach has been taken within the EGO framework by Leary et al. [2004]. If this additional information is available, it can be expected to only improve the performance of our algorithm over the case where it is not. An interesting avenue of future work is to investigate the comparative performances of traditional optimisation procedures utilising derivative

information (such as simple gradient descent and its cousins) and our own. For now, we address the broader class of problems where such information is unavailable.

6.5 Confidence in Output

Note that, even if we do not sample exactly at a minimum, its location will often be evident in light of other nearby observations. Hence an obvious possibility is to allow the algorithm to report a suspected – rather than evaluated – minimum.

As such, we relax the constraint on our final returned value, $x_m \in \boldsymbol{x}_0$. Instead, we merely require that $p(\, y_m \mid I_0 \,)$ has standard deviation equal to or less than some pre-specified threshold ε. Essentially, we quantify by ε exactly how confident we need to be in the returned value y_m. Of course, if ε is sufficiently small, usually only elements of $\boldsymbol{y}_0$ will be eligible to be returned by the algorithm – only points at which the function has actually been evaluated. This suggestion replaces η with

$$\eta_i' \triangleq \min_{x : C_i < \varepsilon^2} m_i \,,$$

now dependent on the hyperparameter values ϕ_i. This change will improve the performance of our algorithm by obviating the necessity to expend a sample simply to return a function value we are already very confident in. As an example, in minimising a function known to be quadratic, with the appropriate covariance function, any three observations away from the minimum will exactly determine where that minimum is. The use of η requires that we then evaluate the function at that point, whereas this new proposal allows us to end the procedure immediately.

This proposal also allows us to counter a possible side-effect of our proposal (to enforce a separation of δ) in Section 6.4, that of being unable to sample exactly at the suspected location of minimum. Now we can return our estimate for such a minimum, about which, with an evaluation less than δ away and derivative observations in addition, we can be very confident.

6.6 Noisy Optimisation

Many applications [Carter et al., 2001] require an effective means of noisy optimisation. In such problems, rather than observing the objective function $y(x)$ directly, we instead observe $z(x)$, which differs from $y(x)$ in the addition of some noise component. For now, we restrict attention to Gaussian noise. In this case, we can easily apply our methods through a trivial modification to the GP's covariance, as per Section 4.5. In the simple example of IID Gaussian noise, we need simply add the noise variance σ^2 to the diagonal of the covariance matrix over such observations[2]. As such, $\boldsymbol{m}(\mathring{\boldsymbol{z}}_\star \,|\, \phi,\, I'_0)$ is equal to (3.3.9) and $\mathbf{C}(\mathring{\boldsymbol{z}}_\star \,|\, \phi,\, I'_0)$ is equal to (3.3.10), where I'_j is the conjunction of I and our noisy observations of the function, $\{(\boldsymbol{x}_{j'}, \boldsymbol{z}_{j'}) : j' = 1, \ldots, j\}$. We must also slightly modify our hyperparameter sample weights to $\boldsymbol{\rho}'$, to allow our likelihoods to incorporate the new covariance.

The noise variance is commonly unknown, in which case it can be treated just as any other hyperparameter. As before, σ is treated as a member of ϕ, over which we now have samples $\boldsymbol{\phi}_{s'}$. Of course, we should always exploit our prior knowledge about the problem at hand. If we suspect the objective function may be corrupted by noise, we should specify a prior over the noise variance and then marginalise it as usual.

Of course, here the considerations of Section 6.5 take a more central importance. If the noise is considerable, we may in fact need to take several function evaluations at, or very near, the same point before we become sufficiently confident in the function's value there. Note that our ultimate loss remains the lowest function value known with sufficient confidence – there is no need for it to be subjected to any heuristic modifications [Huang et al., 2006]. As such, we have the loss function

$$\lambda'(z_j; \phi) \triangleq \min_{x\,:\,C(y|\phi,\, I'_j) < \varepsilon^2} m(y|\phi,\, I'_j)\,,$$

[2]Note that this modification has the fortunate side-effect of alleviating any conditioning problems.

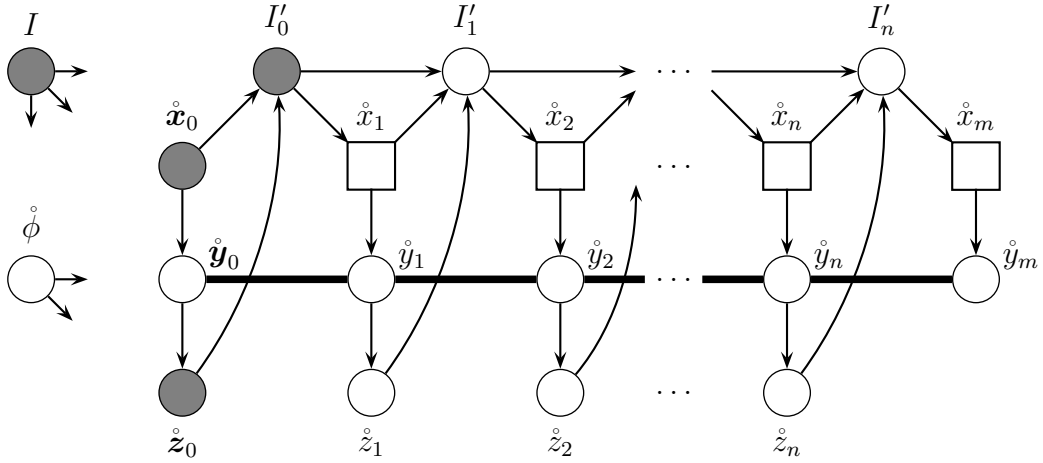


Figure 6.3: A Bayesian network for the noisy optimisation problem. Shaded nodes are known, and square nodes represent the results of a decision. All $\mathring{y}$ nodes are correlated with one another, the context I is correlated with all nodes and $\mathring{\phi}$ is correlated with all $\mathring{y}$ and $\mathring{z}$ nodes.

and, from Figure 6.3, the new n-step lookahead expected loss

$$
\begin{aligned}
&\Lambda'_n(\, x_1 \mid I'_0\,) \\
&\triangleq \int \lambda'(z_n;\phi)\, p\big(z_n\big|x_n,\phi,I'_{n-1}\big)\, p\big(x_n\big|I'_{n-1}\big) \ldots p(z_2|x_2,\phi,I'_1)\, p(x_2|I'_1)\, p(z_1|x_1,\phi,I'_0) \\
&\qquad\qquad \mathrm{d}z_1 \ldots \mathrm{d}z_n \mathrm{d}x_2 \ldots \mathrm{d}x_n\, p(\boldsymbol{z}_0|\boldsymbol{x}_0,\phi,I)\, p(\phi|I)\, \mathrm{d}\phi \Big/ \int p(\boldsymbol{z}_0|\boldsymbol{x}_0,\phi,I)\, p(\phi|I)\, \mathrm{d}\phi \\
&= \sum_{i\in s'} \rho'_i \int \lambda'(z_n;\phi_i) \\
&\qquad \mathcal{N}\big(z_n; m(\mathring{z}_n|\phi_i,I'_{n-1}), C(\mathring{z}_n|\phi_i,I'_{n-1})\big)\delta\big(x_n - \mathrm{argmin}_x \Lambda'_1\big(\, x_\star \;\big|\; I'_{n-1}\big)\big) \ldots \\
&\qquad\qquad \mathcal{N}\big(z_2; m(\mathring{z}_2|\phi_i,I'_1), C(\mathring{z}_2|\phi_i,I'_1)\big)\, \delta\big(x_2 - \mathrm{argmin}_x \Lambda'_{n-1}(\, x_\star \mid I'_1\,)\big) \\
&\qquad\qquad\qquad \mathcal{N}\big(z_1; m(\mathring{z}_1|\phi_i,I'_0), C(\mathring{z}_1|\phi_i,I'_0)\big)\mathrm{d}z_1 \ldots \mathrm{d}z_n \mathrm{d}x_2 \ldots \mathrm{d}x_n\,.
\end{aligned}
$$

As with those described in Section 6.3, these integrals must be approximated as weighted sums. Unfortunately, the analytic result that gave us (6.2.2) does not hold in the noisy case. Of course, if the noise is not too large relative to the scale of variation of the objective function, (6.2.2) will continue to be effective. We assume that we will be sufficiently confident about the function at our new observation that it will be eligible to be returned by the algorithm. Further, we assume that this observation does not promote and demote any other observations to or from such

eligibility. Under this assumption, the 1-step lookahead expected loss is

$$\Lambda'_1(\, x_1 \mid I'_0\,) \approx \sum_{i \in s'} \rho'_i\big(\eta'_i + (m'_i - \eta'_i)\,\Phi(\eta'_i; m'_i, C'_i) - C'_i\,\mathcal{N}(\eta'_i; m'_i, C'_i)\,\big)\,, \qquad (6.6.1)$$

where we have abbreviated $m(z_1|\phi_i, I'_0)$ as m'_i, $C(z_1|\phi_i, I'_0)$ as C'_i and, as before,

$$\eta'_i = \min_{x_1 : C'_i < \varepsilon^2} m'_i\,.$$

6.7 Optimisation of Dynamic Functions

A further challenging problem is the optimisation of functions that change in time. The problem we consider is one in which we have a pre-defined time at which we must return our minimum (x_m, y_m). Until that time, we have a certain number of exploratory evaluations to make. We may be able to select the times of our evaluations, or we may be forced to evaluate at constrained timesteps. Either way, what distinguishes dynamic optimisation from the contexts considered above is that we cannot choose an evaluation that is before any evaluation we have already taken – we cannot travel back in time. With this caveat, the problem is otherwise identical to that considered above. Time is simply taken as an additional input to our objective function, meaning that our inputs will now be of the form $[x, t]$. This requires the incorporation of time into whatever covariance we have used. Our covariance over time is likely to induce additional (but reasonable) uncertainty about the current value at input x, even if we have an evaluation at x in the past. As such, we must make use of the noisy optimisation framework presented above: we cannot be certain about such past observations. This means that before returning the final y_m, we will likely be compelled to revisit the function's minima in order to clarify how they have changed with time.

6.8 Set Selection

We consider now the optimisation of an objective function over sets, as discussed in 4.8. As before, we are motivated by the problem of sensor network arrangement. For

example, we might aim to place a small number of sensors with the goal of making accurate global predictions about a spatial field with temporal variation, such as air temperature. The prediction quality of our set of sensors forms our 'black-box' objective function – a measure of the efficacy of a particular sensor layout that is supplied to us after having tried it. This measure could be any arbitrarily complicated function, and might potentially be provided to us by an external agency, such as a user of the sensor network.

We now briefly describe how GPGO could be used for optimising such a function. Suppose we have a discrete set X and a real-valued function $f\colon \mathcal{P}(X) \to \mathbb{R}$ defined on the power set of X. Suppose further that we have selected an appropriate distance $d\colon \mathcal{P}(X)\times\mathcal{P}(X) \to \mathbb{R}^+$ between subsets of X, as per Section 4.8. We use d to build an homogenous covariance of the form (3.2.1). Given this covariance, and an appropriate mean function over $\mathcal{P}(X)$, the application of GPGO to our problem is straightforward. At each step of the optimisation process, we evaluate f on a particular subset of X. We update our GP over $\mathcal{P}(X)$ with this observation. Next we evaluate the GPGO expected loss function (for set selection problems, we will consider only the use of the one-step lookahead (6.2.2)) on candidate subsets in $\mathcal{P}(X)$. The minimum among these becomes the subset used for the next evaluation of f. We continue in this manner for as long as desired.

In many cases, the objective function f might exhibit dynamism; that is, it may be a function of both time and the chosen subset. In this case, we adjust our inputs (as per Section 6.7) to be of the form $[S, t]$. Here S is a subset of X and t is time. We also allow for the presence of noise in our measurements of the objective function. As per Sections 6.6 and 6.7, the noise and dynamics in our objective function impact upon our choice of the final point to return as the minimum. We'll constrain the returned x_m to the set of actual evaluated points. At such points, we are likely to have only a small amount of uncertainty about the objective function. We further need to limit the candidate points to return to those at which we are still reasonably confident about the objective: as we are optimising a dynamic function, its current

value at input x is likely to be different from an evaluation at x made in the past. We enforce this final constraint by stipulating that the returned value must be at an x at which we have an evaluation made not more than ϵ units of time in the past.

Imagine that we have only one allowed function evaluation remaining before we must report our inferred function minimum, at the final time t_m. If

$$([\boldsymbol{S}_d, \boldsymbol{t}_d], \boldsymbol{f}_d) \triangleq \left\{([S_j, t_j], f_j) : j \in d\right\}$$

are the evaluations of the function we have gathered so far, these constraints above require the redefinition

$$\eta_i' \triangleq \min_{j \in d:\, t_j > t_m - \epsilon} m\left(\mathring{f}_j \,\middle|\, \phi_i, \boldsymbol{f}_d, I\right) .$$

As such, our method is able to cope with the potentially dynamic nature of our objective function, changing the selection of subset over time as appropriate. Pseudo-code for N iterations of our algorithm is provided in Algorithm 2.

Of course, the size of $\mathcal{P}(X)$ grows exponentially in the size of X, and evaluating our expected loss function at every point quickly becomes infeasible. However, the nature of this expected loss function suggests various search heuristics that can be adopted to guide the search. In particular, the expected loss function is likely to be minimised at either a point that is close to an observation we have already made (exploitation), or far away from every observation (exploration). We can speed up our search by limiting our search to sets with these properties. Of course, evaluating the distance itself between all pairs of subsets can also become quickly infeasible, but we may apply further heuristics, for example encouraging exploitation by including subsets that differ from our current observations by one point and encouraging exploration by including a random selection of the remaining subsets. The effect to Algorithm 2 is that the search over the set S' is constrained to these heuristically chosen points.

Algorithm 2 GPGO-Sets($\textsc{Obj}(\cdot,\cdot), S_1, N, \boldsymbol{\phi}_s$)

// Returns the set that optimises objective function.
// $\textsc{Obj}(\cdot,\cdot)$: Objective function, takes a set and a time as arguments.
// S_1: Initial set.
// N: Number of objective function evaluations permitted.
// $\boldsymbol{\phi}_s$: Hyperparameter samples.

1: **for** $i = 1 \dots N$ **do**
2: $f_i \leftarrow \textsc{Obj}(S_i, t_i)$
3: $S_{i+1} \leftarrow \text{argmin}_{S'}$
GPGO-Expected-Loss$\Big((S', t_{i+1}), \big((S_1, t_1), \dots, (S_i, t_i)\big), (f_1, \dots, f_i\}, 1, \boldsymbol{\phi}_s\Big)$
// Using Algorithm 1.
4: **end for**
5: **Return:** $S(N+1)$

Chapter 7

Bayesian Quadrature

7.1 Simple Bayesian Quadrature

Bayesian quadrature [O'Hagan, 1991, Rasmussen and Ghahramani, 2003] is a means of performing Bayesian inference about the value of a potentially nonanalytic integral

$$\mathring{\varphi} \triangleq \int f(\phi)\, p(\,\phi \mid I\,)\, \mathrm{d}\phi\,. \tag{7.1.1}$$

In particular, we consider integrating with respect to a Gaussian prior

$$p(\,\phi \mid I\,) \triangleq \mathcal{N}\big(\phi; \nu^{(\phi)}, \Lambda^{(\phi)}\big)\ . \tag{7.1.2}$$

Quadrature inevitably involves evaluating $f(\phi)$ at a set of sample points $\boldsymbol{\phi}_s$, giving $\mathring{\boldsymbol{f}}_s \triangleq f(\boldsymbol{\phi}_s)$. Of course, this evaluation is a computationally expensive operation. Defining the vector of all possible function inputs as $\boldsymbol{\phi}_:$, we clearly can't afford to evaluate $\boldsymbol{f}_:$. We view our sparse sampling as introducing a form of uncertainty.

As ever in the face of uncertainty, we address the estimation of the value of our integral as itself a problem of Bayesian inference [O'Hagan, 1992]. In considering any problem of inference, we need to be clear about both what information we have and which uncertain variables we are interested in. In our case, both the values $f(\boldsymbol{\phi}_s)$ and their locations $\boldsymbol{\phi}_s$ represent valuable pieces of knowledge. As discussed by O'Hagan [1987], traditional, frequentist Monte Carlo, which approximates as

$$\int f(\phi)\, p(\,\phi \mid I\,)\, \mathrm{d}\phi \simeq \frac{1}{|s|} \sum_{i=1}^{|s|} f(\phi_i)\,, \tag{7.1.3}$$

effectively ignores the information content of $\boldsymbol{\phi}_s$, leading to several unsatisfactory features.

As with our convention above, we will take knowledge of sample locations $\boldsymbol{\phi}_s$ to be implicit within I. However, as we don't know $f(\phi)$ for any $\phi \notin \boldsymbol{\phi}_s$, we are uncertain about the function $f(\cdot)$. As a consequence, we are also uncertain about the value of the integral φ. As such, we possess probability distributions over both $f(\cdot)$ and φ. The resulting Bayesian network is depicted in Figure 7.1.

In particular, for f, we choose a zero-mean GP prior with a Gaussian covariance function

$$K^{(f)}(\phi_1, \phi_2) \triangleq h^{(f)} \mathcal{N}\big(\phi_1; \phi_2, W^{(f)}\big) \,. \tag{7.1.4}$$

Here the scales $h^{(f)}$ and $W^{(f)}$ are *quadrature hyperparameters*, hyperparameters that specify the GP used for Bayesian quadrature. We'll refer to the set of quadrature hyperparameters as θ. We'll also take θ as given and incorporate it into I. Note that it will later become apparent that our inference for (7.1.1) is independent of the $h^{(f)}$ quadrature hyperparameter.

As integration is a projection (see Section 4.7), we can use our computed samples $\boldsymbol{f}_s$ to perform analytic Gaussian process inference about the value of integrals over $f(\phi)$. In the example of φ, we have

$$\begin{aligned}
p(\,\varphi \mid \boldsymbol{f}_s,\, I\,) &= \int p(\,\varphi \mid \boldsymbol{f}_:,\, I\,)\, p(\,\boldsymbol{f}_: \mid \boldsymbol{f}_s,\, I\,)\, \mathrm{d}\boldsymbol{f}_: \\
&= \int \delta\bigg(\varphi - \int f(\phi)\, p(\,\phi \mid I\,)\, \mathrm{d}\phi\bigg)\, \mathcal{N}\Big(\boldsymbol{f}_:; \boldsymbol{m}^{(f)}_{:|s}, \mathbf{C}^{(f)}_{:|s}\Big)\, \mathrm{d}\boldsymbol{f}_: \\
&= \lim_{\varepsilon \to 0} \int \mathcal{N}\bigg(\varphi; \int f(\phi)\, p(\,\phi \mid I\,)\, \mathrm{d}\phi, \varepsilon\bigg)\, \mathcal{N}\Big(\boldsymbol{f}_:; \boldsymbol{m}^{(f)}_{:|s}, \mathbf{C}^{(f)}_{:|s}\Big)\, \mathrm{d}\boldsymbol{f}_: \\
&= \mathrm{N}\big(\varphi; \boldsymbol{m}(\mathring{\varphi} \,|\, \boldsymbol{f}_s,\, I)\,, \mathbf{C}(\mathring{\varphi} \,|\, \boldsymbol{f}_s,\, I)\big)\,,
\end{aligned}$$

where

$$
\begin{aligned}
m(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I) \;&\triangleq\; \int p(\,\phi_* \mid I\,)\; m\Big(\mathring{f}_* \,\Big|\, \boldsymbol{f}_s, I\Big)\; \mathrm{d}\phi_* \\
&= \mathfrak{y}_s^{(f)\,\mathsf{T}} \mathbf{K}^{(f)}\big(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s\big)^{-1} \boldsymbol{f}_s \qquad (7.1.5) \\
C(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I) \;&\triangleq\; \iint p(\,\phi_* \mid I\,)\; \mathbf{C}\Big(\mathring{f}_*,\, \mathring{f}_\star \,\Big|\, \boldsymbol{f}_s,\, I\Big)\; p(\,\phi'_\star \mid I\,)\; \mathrm{d}\phi_* \mathrm{d}\phi'_\star \\
&= \mathfrak{C}^f - \mathfrak{y}_s^{(f)\,\mathsf{T}} \mathbf{K}^{(f)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \mathfrak{y}_s^{(f)}\,, \qquad (7.1.6)
\end{aligned}
$$

having used the standard GP posterior equations (3.3.2) and (3.3.4), and defining

$$
\begin{aligned}
\mathfrak{y}_s^{(f)} \;&\triangleq\; \Big[\int p(\,\phi \mid I\,)\; K^{(f)}(\phi, \phi_i)\, \mathrm{d}\phi;\; i \in s\Big]^{\mathsf{T}} \\
&=\; \big[h^{(f)}\, \mathcal{N}\big(\phi_i; \nu^{(\phi)}, \Lambda^{(\phi)} + W^{(f)}\big)\,;\, i \in s\big]^{\mathsf{T}} \qquad (7.1.7) \\
\mathfrak{C}^{(f)} &= h^{(f)} \iint p(\,\phi \mid I\,)\; K^{(f)}(\phi, \phi')\, p(\,\phi' \mid I\,)\; \mathrm{d}\phi\, \mathrm{d}\phi' \\
&= \Big(\det 2\pi\Big(2\,\Lambda^{(\phi)} + W^{(f)}\Big)\Big)^{-\frac{1}{2}}. \qquad (7.1.8)
\end{aligned}
$$

Note that the form of our 'best estimate' for φ, (7.1.5), is a linear combination of the samples $\boldsymbol{f}_s$, identical to that of traditional quadrature or Monte Carlo techniques.

Note also that $h^{(f)}$ represents a simple multiplicative factor to both $\mathfrak{y}_s^{(f)}$ and $\mathbf{K}^{(f)}\big(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s\big)$, and as such cancels out of (7.1.5).

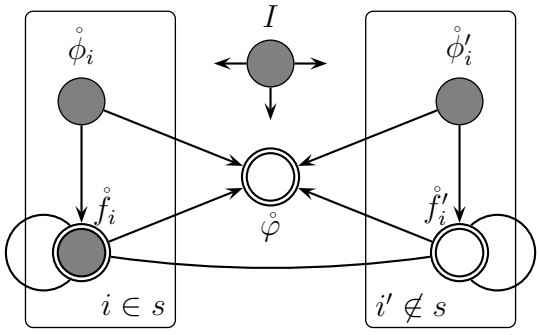


Figure 7.1: Bayesian network for Bayesian Quadrature.

7.2 Second-order Probabilities

It is not unusual for the integral(s) we are performing inference about to give a probability. Note that if $f(\phi)$ is a likelihood, e.g. $p(\,y \mid \phi,\, I\,)$, (7.1.1) will be a

probability distribution itself. In such cases, we have a second-order probability for our integral(s), as discussed in Section 2.5. Assume we have such a probability (or pdf) φ determined by some combination of integrals over unknown function f. In the event that φ is determined by integrals over multiple unknown functions, e.g. $\int f_1(\phi) f_2(\phi) \mathrm{d}\phi$, we simply allow f multiple outputs/inputs as required. Without loss of generality, we assume that we are interested in the probability (or pdf) for some y conditioned on some information I. However, we are unable to analytically evaluate the required integrals over f. We have observations $\boldsymbol{f}_s$, some subset of the total possible values of f, all of which would be required in order to be certain about our probability. We write $\boldsymbol{f}_{\overline{s}}$ for the complement of $\boldsymbol{f}_s$; together, they represent all possible values of the function, $\boldsymbol{f}_:$. As such, we can write

$$\mathring{\varphi} = p(\, y \mid \boldsymbol{f}_s,\, \boldsymbol{f}_{\overline{s}},\, I\,)$$

Note that $\boldsymbol{f}_{\overline{s}}$ forms the additional information J we considered in Section 2.5. Of course, we don't actually know $\boldsymbol{f}_{\overline{s}}$, so the quantity of interest is, as before

$$\begin{aligned} p(\, y \mid \boldsymbol{f}_s,\, I\,) &= \iint p(\, y \mid \varphi\,)\; p(\, \varphi \mid \boldsymbol{f}_s,\, \boldsymbol{f}_{\overline{s}},\, I\,)\; p(\, \boldsymbol{f}_{\overline{s}} \mid \boldsymbol{f}_s,\, I\,)\; \mathrm{d}\boldsymbol{f}_{\overline{s}} \mathrm{d}\varphi \\ &= \int \varphi\, p(\, \varphi \mid \boldsymbol{f}_s,\, I\,)\; \mathrm{d}\varphi \\ &= m(\mathring{\varphi} \,|\, \boldsymbol{f}_s,\, I) \end{aligned} \tag{7.2.1}$$

where we have taken $p(\, y \mid \varphi\,) = \varphi$. Hence inference about y is only ever dependent on the mean of $p(\, \varphi \mid \boldsymbol{f}_s,\, I\,)$, $m(\mathring{\varphi} \,|\, \boldsymbol{f}_s,\, I)$. We'll use this fact extensively in the sections to follow. The higher moments of $p(\, \varphi \mid \boldsymbol{f}_s,\, I\,)$, do, however, gain pertinence when considering the selection of the locations of $\boldsymbol{f}_s$. This idea will be pursued in greater depth in Chapter 8.

7.3 Predictive Bayesian Quadrature

We return now to the numerical evaluation of the ratio of integrals (3.4.1)

$$p(\boldsymbol{y}_\star | \boldsymbol{z}_d, I) = \frac{\int p(\boldsymbol{y}_\star | \boldsymbol{z}_d, \phi_*, I)\, p(\boldsymbol{z}_d | \phi_*, I)\, p(\phi_* | I)\; \mathrm{d}\phi_*}{\int p(\boldsymbol{z}_d | \phi_*, I)\, p(\phi_* | I)\; \mathrm{d}\phi_*}, \tag{7.3.1}$$

in which we have marginalised ϕ_*. We emphasise that this very general form is not limited to GP regression, but is also encountered in the generic problems of prediction, classification, optimisation and others. For the ease of exposition, we'll limit ourselves again to the Gaussian prior for our hyperparameters ϕ, (7.1.2), although we'll relax this constraint in Section 7.5. The methods we will henceforth describe hold for any arbitrary likelihood function $p(\,\boldsymbol{z}_d \mid \phi_*, I\,)$ and prediction model $p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi_*, I\,)$[1]. We consider the case in which those terms potentially exhibit non-trivial dependence upon ϕ_* such that our integrals are non-analytic. Defining the two quantities

$$\mathring{q}_{\star,*} \triangleq q(\boldsymbol{y}_\star,\, \phi_*) \triangleq p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi_*,\, I\,) \tag{7.3.2}$$

$$\mathring{r}_* \triangleq r(\phi_*) \triangleq p(\,\boldsymbol{z}_d \mid \phi_*,\, I\,)\,, \tag{7.3.3}$$

we evaluate them at samples $\boldsymbol{\phi}_s$, giving

$$\mathring{\boldsymbol{q}}_{\star,s} \triangleq q(\boldsymbol{y}_\star,\, \boldsymbol{\phi}_s)$$

$$\mathring{\boldsymbol{r}}_s \triangleq r(\boldsymbol{\phi}_s)\,.$$

Our evaluation of both q and r at the same set of hyperparameter samples $\boldsymbol{\phi}_s$ is not absolutely necessary, but results in some computational savings, as we'll see later. Note that the more complex our model, and hence the greater the number of hyperparameters, the higher the dimension of the hyperparameter space we must sample in. The complexity of models we can practically consider is therefore limited by the curse of dimensionality.

Using such samples, traditional Monte Carlo would approximate as

$$\frac{\int q(\boldsymbol{y}_\star, \phi_*)\, r(\phi_*)\, p(\phi_*|I)\ \mathrm{d}\phi_*}{\int r(\phi_*)\, p(\phi_*|I)\ \mathrm{d}\phi_*} \simeq \frac{1}{|s|}\sum_{i=1}^{|s|} q(\boldsymbol{y}_\star, \phi_i)\,, \tag{7.3.4}$$

where with reference to (7.1.3), we substitute $q(\boldsymbol{y}_\star, \phi_*)$ for $f(\phi_*)$ and

$$p(\,\phi_* \mid \boldsymbol{r}_:,\, \boldsymbol{z}_d,\, I\,) \triangleq \frac{r(\phi_*)\, p(\phi_*|I)}{\int r(\phi)\, p(\phi|I)\ \mathrm{d}\phi} \tag{7.3.5}$$

[1]In the case of optimisation using GPGO, note that although the 'prediction model' is actually an expected minimum such as $\mathrm{V}(\,x \mid \phi_i,\, I_0\,)$ from (6.2.2), the methods described below are equally applicable. We need only transform the expected minimum so it is a non-negative *expected improvement* upon the current best minimum, by considering instead $\eta - \mathrm{V}(\,x \mid \phi_i,\, I_0\,)$.

for $p(\,\phi_* \mid I\,)$. Of course, such an approach ignores relevant information and privileges irrelevant information [O'Hagan, 1987]. Instead, we will investigate the use of Bayesian Quadrature techniques for this problem.

As such, we now assign GP priors to the logarithms

$$\tilde{q} \triangleq \log\left(\frac{q}{q_\mu}\right) \tag{7.3.6}$$

$$\tilde{r} \triangleq \log\left(\frac{r}{r_\mu}\right) \tag{7.3.7}$$

These choices of prior distributions over logarithms are motivated by the fact that both q and r are strictly positive and possess a large dynamic range. For both $\tilde{q}$ and $\tilde{r}$ we take Gaussian covariances of the form (7.1.4). We effectively consider $\tilde{q}$ as a function solely of ϕ, although of course it is also dependent upon $\boldsymbol{y}_\star$ and, implicitly, $\boldsymbol{x}_\star$. We can consider $\boldsymbol{x}_\star$ fixed in any instance (our integrals are employed in making predictions about a particular point) and hence ignore it henceforth. As our observations are usually of the form of predictions for all $\boldsymbol{y}_\star$ (that is, a prediction function) for a particular ϕ, we need never perform inference over $\boldsymbol{y}_\star$ and can hence safely ignore it too[2].

Here $q_\mu \triangleq \min_i q_{i,\star}$ and $r_\mu \triangleq \min_i r_i$ are scaling factors such that we can justifiably assume a zero prior mean for our GPs over $\tilde{q}$ and $\tilde{r}$. Essentially, we assume that far away from our existing observations, the means for q and r are equal to values suitably close to the minima observed. This is not unreasonable for r – we do not expect there to be significant likelihood at all values of ϕ. For q, this assumption is less reasonable (although, of course, convenient). However, the negative consequences of this assumption should be limited, as q is only ever considered in conjunction with r. The prior mean has significant influence only for locations well-removed from existing observations. Here, as our mean for r will be effectively zero, the product of q and r that appears in our numerator integrand will be very small regardless of our mean for q.

[2]In more detail, with a product covariance for $\tilde{q}$ of the form (4.2.1) over the two variables $\boldsymbol{y}_\star$ and ϕ, our results will prove to be the product of terms over $\boldsymbol{y}_\star$ and ϕ. With our dense observations over $\boldsymbol{y}_\star$, we obtain the results stated below.

We now use these priors to perform inference about our integrals[3] over q and r,

$$\mathring{\varrho} \triangleq p\big(\boldsymbol{y}_\star \big| \tilde{\boldsymbol{q}}_{\star,:},\, \tilde{\boldsymbol{r}}_{:},\, \boldsymbol{z}_d,\, I\big) = \frac{\int q(\boldsymbol{y}_\star, \phi)\, r(\phi)\, p(\phi|I)\, \mathrm{d}\phi}{\int r(\phi)\, p(\phi|I)\, \mathrm{d}\phi}\,.$$

Note that here $r(\phi)$ appears in both our numerator and denominator integrals, introducing correlations between the values we estimate for them. For this reason, we must consider the ratio as a whole rather than performing inference for the numerator and denominator separately, as would be performed should we employ simple Bayesian Quadrature for this problem. The correlation structure of this system is illustrated in Figure 7.2.

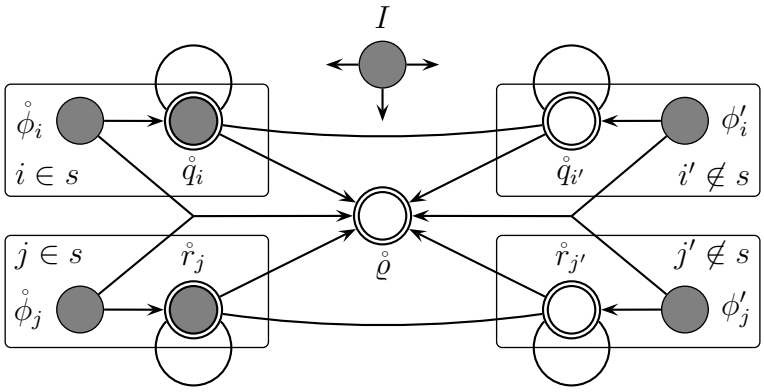


Figure 7.2: Bayesian network for marginalising hyperparameters using BMC.

In considering any problem of inference, we need to be clear about both what information we have and which uncertain variables we are interested in. In this case, both function values, $\boldsymbol{q}_{\star,s}$ and $\boldsymbol{r}_s$, and their locations, $\boldsymbol{\phi}_s$, represent valuable pieces of knowledge. As before, we take $\boldsymbol{\phi}_s$ to be implicit within I. For later convenience, we define

$$m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,s},\, \boldsymbol{r}_{:},\, I\big) \triangleq \frac{\int m^{(q_\star)}_{*|s}\, r_*\, p(\phi_*|I)\, \mathrm{d}\phi_*}{\int r_*\, p(\phi_*|I)\, \mathrm{d}\phi_*}\,. \tag{7.3.8}$$

[3]In fact, there is a different variable $\mathring{\varrho}_\star$ for different values of $\boldsymbol{y}_\star$ and $\boldsymbol{x}_\star$. To aid exposition, we consider fixed $\boldsymbol{x}_\star$ and $\boldsymbol{y}_\star$, as above, and so will drop the $\star$ subscript on $\mathring{\varrho}$ (and similar variables) henceforth.

The ultimate quantity of our interest is then

$$\begin{aligned} p\big(\boldsymbol{y}_\star \big| \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, \boldsymbol{z}_d,\, I\big) &= \iiint p(\boldsymbol{y}_\star|\varrho)\; p\big(\varrho \big| \tilde{\boldsymbol{q}}_{\star,:},\, \tilde{\boldsymbol{r}}_:, I\big)\; p\big(\tilde{\boldsymbol{q}}_{\star,:} \big| \tilde{\boldsymbol{q}}_{\star,s}, I\big) \\ &\qquad p(\tilde{\boldsymbol{r}}_:|\tilde{\boldsymbol{r}}_s, I)\; \mathrm{d}\varrho\, \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\ &= \iiint \varrho\, \delta(\varrho - \mathring{\varrho})\; \mathcal{N}\Big(\tilde{\boldsymbol{q}}_{\star,:}; \boldsymbol{m}_{:|s}^{(\tilde{q}_\star)}, \mathbf{C}_{:|s}^{(\tilde{q}_\star)}\Big) \\ &\qquad \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}_{:|s}^{(\tilde{r})}, \mathbf{C}_{:|s}^{(\tilde{r})}\Big)\; \mathrm{d}\varrho\, \mathrm{d}\boldsymbol{q}_{\star,:}\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\ &\simeq \iiint \varrho\, \delta(\varrho - \mathring{\varrho})\; \mathcal{N}\Big(\boldsymbol{q}_{\star,:}; \boldsymbol{m}_{:|s}^{(q_\star)}, \mathbf{C}_{:|s}^{(q_\star)}\Big) \\ &\qquad \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}_{:|s}^{(\tilde{r})}, \mathbf{C}_{:|s}^{(\tilde{r})}\Big)\; \mathrm{d}\varrho\, \mathrm{d}\boldsymbol{q}_{\star,:}\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\ &= \int m\big(\mathring{\varrho}\,\big|\, \boldsymbol{q}_{\star,s},\, \boldsymbol{r}_:,\, I\big)\; \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}_{:|s}^{(\tilde{r})}, \mathbf{C}_{:|s}^{(\tilde{r})}\Big)\; \mathrm{d}\tilde{\boldsymbol{r}}_:\,, \end{aligned} \tag{7.3.9}$$

where, as per Section 7.2, $p(\boldsymbol{y}_\star|\varrho) = \varrho$, and we have approximated as

$$\mathcal{N}\Big(\tilde{\boldsymbol{q}}_{\star,:}; \boldsymbol{m}_{:|s}^{(\tilde{q}_\star)}, \mathbf{C}_{:|s}^{(\tilde{q}_\star)}\Big) \left|\frac{\mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}}{\mathrm{d}\boldsymbol{q}_{\star,:}}\right| \simeq \mathcal{N}\Big(\boldsymbol{q}_{\star,:}; \boldsymbol{m}_{:|s}^{(q_\star)}, \mathbf{C}_{:|s}^{(q_\star)}\Big) \tag{7.3.10}$$

in order to render one of our integrals analytic. This new GP over $q_\star$ employs a zero-mean prior and Gaussian covariance (7.1.4). For appropriate selections of $W^{(\tilde{q}_\star)}$ and $W^{(q_\star)}$, our assumption can be made reasonable. Of course, we do not correctly account for the uncertainty in $\tilde{q}_{\star,*}$. However, this uncertainty can be expected to be large at locations where we also have large uncertainty in $\tilde{r}_*$, that is, for ϕ_* far removed from $\boldsymbol{\phi}_s$. As such, our proper treatment of the uncertainty in $\tilde{r}_*$ will result in the exploration of areas where the uncertainty in $\tilde{q}_{\star,*}$ is also large, as desired. Note that the use of simple Bayesian Quadrature for (7.3.1) would require not only (7.3.10) but also a similar assumption for r, ignoring the positivity of both q and r. Figure 7.3 illustrates some of the consequences of our approximation. Note that the approximated mean is a faithful representation of the real mean (for well-learned $W^{(\tilde{q}_\star)}$ and $W^{(q_\star)}$). Unfortunately, of course, the error bars of our approximate distribution incorrectly assign non zero probability to negative q.

7.3.1 MAP approximation for log-likelihood function

At this point, unfortunately, our integration over r becomes nonanalytic. One simple solution is to take an additional maximum *a posteriori* (MAP) approximation for r,

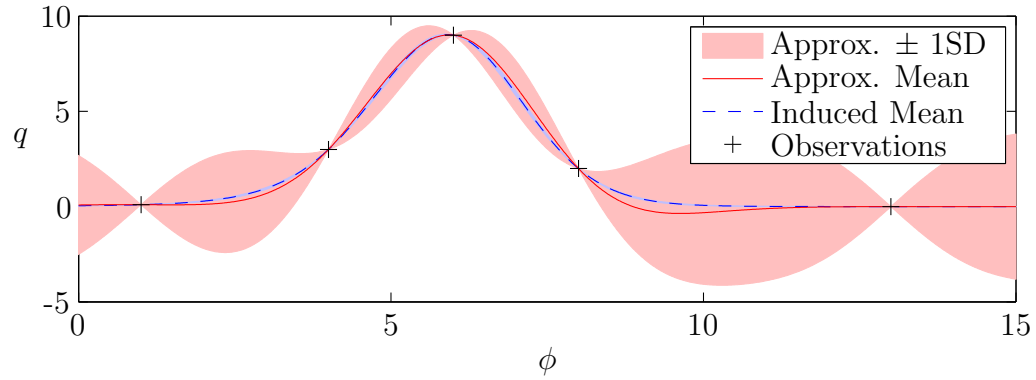


Figure 7.3: An illustration of the approximation (7.3.10). In blue and with a dashed mean, we have an example distribution for the likelihood function $\boldsymbol{q}_{\star,:}$ induced by $\mathcal{N}\left(\tilde{\boldsymbol{q}}_{\star,:}; \boldsymbol{m}_{:|s}^{(\tilde{q}_\star)}, \mathbf{C}_{:|s}^{(\tilde{q}_\star)}\right) |\mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}/\mathrm{d}\boldsymbol{q}_{\star,:}|$. In red, the distribution we approximate it by, $\mathcal{N}\left(\boldsymbol{q}_{\star,:}; \boldsymbol{m}_{:|s}^{(q_\star)}, \mathbf{C}_{:|s}^{(q_\star)}\right)$. The $\pm 1SD$ region for the blue distribution, while drawn, is largely invisible at the scale plotted.

which takes

$$\mathcal{N}\left(\tilde{\boldsymbol{r}}_{:}; \boldsymbol{m}_{:|s}^{(\tilde{r})}, \mathbf{C}_{:|s}^{(\tilde{r})}\right) \simeq \delta\left(\boldsymbol{r}_{:} - \boldsymbol{m}_{:|s}^{(r)}\right) . \tag{7.3.11}$$

This represents the assumption that our uncertainty about r is small, and that, just as for our (7.3.10), a distribution over a log-transformed variable, here $\tilde{r}$, can be approximated by a distribution over the untransformed variable, here r. This new GP for r as a function of ϕ again employs a zero-mean prior and Gaussian covariance (7.1.4).

With this approximation, we arrive at

$$p\big(\, \boldsymbol{y}_\star \,\big|\, \boldsymbol{q}_{\star,s},\, \boldsymbol{r}_s,\, \boldsymbol{z}_d,\, I \,\big) \simeq \frac{\boldsymbol{q}_{\star,s}^{\mathsf{T}} \mathbf{K}^{(q_\star)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \mathfrak{Y}_{s,s}^{(q_\star,r)} \mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \boldsymbol{r}_s}{\mathfrak{y}_s^{(r)\,\mathsf{T}}\, \mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s} \tag{7.3.12}$$

wher $\mathfrak{y}_s^{(r)\,\mathsf{T}}$ is of the form (7.1.7) and we define for $(i,j) \in (v_{\text{left}} \times v_{\text{right}})$,

$$\begin{aligned} \mathfrak{Y}_{u_{\text{left}}, v_{\text{right}}}^{(f,g)}(i,j) &\triangleq \int K^{(f)}(\phi_i, \phi_*)\, p(\, \phi_* \mid I \,)\; K^{(g)}(\phi_*, \phi_j)\, \mathrm{d}\phi_* \\ &= \left(h^{(f)} h^{(g)}\right)^2 \mathcal{N}\left(\begin{bmatrix} \phi_i \\ \phi_j \end{bmatrix}; \begin{bmatrix} \nu^{(\phi)} \\ \nu^{(\phi)} \end{bmatrix}, \begin{bmatrix} \Lambda^{(\phi)} + W^{(f)} & \Lambda^{(\phi)} \\ \Lambda^{(\phi)} & \Lambda^{(\phi)} + W^{(g)} \end{bmatrix} \right), \end{aligned} \tag{7.3.13}$$

7.3.2 Quadrature over log-likelihood functions

Without such a MAP approximation, we are forced to perform another stage of Bayesian quadrature in order to resolve (7.3.9). We assign another zero-mean GP with Gaussian covariance (7.1.4) to express how we expect $m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,s},\,\boldsymbol{r}_{:},\,I\big)$ (7.3.8) to vary as a function of $\tilde{\boldsymbol{r}}_{:}$. Recall that $\tilde{\boldsymbol{r}}_{:}$ is a complete log-likelihood function – the (infinite) vector of log-likelihoods for all possible values of the hyperparameters. To approximate our integral, we are required to evaluate the integrand at a number of different values of the variable of integration, $\tilde{\boldsymbol{r}}_{:}$. With a sample set containing many possible log-likelihood functions, $(\tilde{\boldsymbol{r}}_{:})_l$, and defining

$$\begin{aligned}\mathring{\alpha} &\triangleq p\big(\boldsymbol{y}_\star\big|\boldsymbol{q}_{\star,s},\,\boldsymbol{r}_s,\,\boldsymbol{z}_d,\,I\big)\\ a(\tilde{\boldsymbol{r}}_{:}) &\triangleq m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,s},\,\boldsymbol{r}_{:},\,I\big)\\ \boldsymbol{a}_l \triangleq a\big((\tilde{\boldsymbol{r}}_{:})_l\big) &\triangleq \big[m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,s},\,\boldsymbol{r}_{:},\,I\big)\,;\,\tilde{\boldsymbol{r}}_{:}\in(\tilde{\boldsymbol{r}}_{:})_l\big]^{\mathsf{T}}\end{aligned}$$

we have

$$p\big(\boldsymbol{y}_\star\big|\tilde{\boldsymbol{q}}_{\star,s},\tilde{\boldsymbol{r}}_s,\boldsymbol{z}_d,\boldsymbol{a}_l,I\big) = m(\mathring{\alpha}\,|\,\boldsymbol{a}_l,I) = \mathfrak{y}_l^{(a)\,\mathsf{T}}\mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_{:})_l,(\tilde{\boldsymbol{r}}_{:})_l\big)^{-1}\boldsymbol{a}_l \tag{7.3.14}$$

where $\mathfrak{y}_l^{(a)\,\mathsf{T}}$ is of the form (7.1.7) with $\nu^{(\tilde{\boldsymbol{r}}_{:})} = \boldsymbol{m}^{(\tilde{r})}_{:|s}$ and $\Lambda^{(\tilde{\boldsymbol{r}}_{:})} = \mathbf{C}^{(\tilde{r})}_{:|s}$. Note that (7.3.14) is our 'best' estimate for (7.3.1).

We now have the problem of selecting sample log-likelihood functions. Of course, our evaluated likelihoods $\boldsymbol{r}_s$ constrain somewhat the form that our likelihood function could take. Our approach is to take sample functions of the form of a GP conditional mean (3.3.2) for the likelihood, motivated by the consequent analytic niceties. We choose to condition those GP means on the values of the likelihood at our existing samples $\boldsymbol{\phi}_s$ (as the prior for functions $\tilde{\boldsymbol{r}}_{:}$ that do not pass through these points is zero). We additionally condition on the simulated likelihoods at a set of *candidate* points, $\boldsymbol{\phi}_c$, to allow for exploration of likelihood function space. That is, for such a sample log-likelihood function, $i \in l$

$$(\tilde{\boldsymbol{r}}_{:})_i \triangleq \log\left(\frac{\boldsymbol{m}(\mathring{\boldsymbol{r}}_{:}\,|\,\boldsymbol{r}_s,\,(\boldsymbol{r}_c)_i,\,I)}{r_\mu}\right) = \log\left(\frac{\boldsymbol{m}^{(r)}_{:|[s,c_i]}}{r_\mu}\right) \tag{7.3.15}$$

and

$$a_i = a\big((\tilde{\boldsymbol{r}}_{:})_i\big) = \frac{\int m_{*|s}^{(q_\star)} m_{*|[s,c_i]}^{(r)}\, p(\phi_*|I)\,\mathrm{d}\phi_*}{\int m_{*|[s,c_i]}^{(r)}\, p(\phi_*|I)\,\mathrm{d}\phi_*}$$
$$= \frac{\boldsymbol{q}_{\star,s}^{\mathsf{T}} \mathbf{K}^{(q_\star)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1} \mathfrak{Y}_{s,[s,c]}^{(q_\star,r)} \mathbf{K}^{(r)} \left(\boldsymbol{\phi}_{[s,c]},\boldsymbol{\phi}_{[s,c]}\right)^{-1} \boldsymbol{r}_{[s,c_i]}}{\mathfrak{y}_{[s,c]}^{(r)\,\mathsf{T}} \mathbf{K}^{(r)} \left(\boldsymbol{\phi}_{[s,c]},\boldsymbol{\phi}_{[s,c]}\right)^{-1} \boldsymbol{r}_{[s,c_i]}}\,, \qquad (7.3.16)$$

where, as per (7.3.12), $\mathfrak{y}_{[s,c]}^{(r)\,\mathsf{T}}$ is of the form (7.1.7) and $\mathfrak{Y}_{s,[s,c]}^{(q_\star,r)}$ is of the form (7.3.13). We will usually assume that the candidate locations, $\boldsymbol{\phi}_c$, are identical for all sample functions, $(\tilde{\boldsymbol{r}}_{:})_i$. We'll discuss how to select those candidate points later, in Section 8.4.2. An example of the sample log-likelihood functions consequently produced is contained in Figure 8.1.

For each sample likelihood function, then, a (7.3.16) is a weighted linear combination of our predictions $\boldsymbol{q}_{\star,s}$. To then find our "best" estimate for (7.3.1), (7.3.14), we combine $\boldsymbol{a}_l$ in a further weighted linear combination. One way of viewing our approximated prediction for $\boldsymbol{y}_\star$, then, is as a weighted combination of the predictions made under a range of hyperparameter samples. The same is true of our MAP approximation (7.3.12). As such, we can now return to (3.4.8),

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, I\,) \simeq \sum_{i\in s} \rho_i\, p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi_i,\, I\,)\ , \qquad (7.3.17)$$

where the weights $\boldsymbol{\rho}_s$ will be specified as appropriate by one of our approaches to the treatment of $\tilde{\boldsymbol{r}}_{:}$.

7.4 Hyperparameter posterior distribution

We now consider the problem of reporting our posterior beliefs about our hyperparameters, taking once again the Gaussian prior (7.1.2). We can use BQ techniques to estimate the posterior distribution for our hyperparameters [Garnett et al., 2010a],

$$p(\phi_*|\boldsymbol{z}_d,\, I) = \frac{p(\boldsymbol{z}_d|\phi_*,\, I)\ p(\phi_*|I)}{\int p(\boldsymbol{z}_d|\phi,\, I)\ p(\phi|I)\,\mathrm{d}\phi}\ . \qquad (7.4.1)$$

Here we can again take our GP for $\tilde{r}$ (with zero mean and Gaussian covariance (7.1.4)) and use it to perform inference about $\mathring{\varrho}'$

$$\mathring{\varrho}' \triangleq p(\phi_*|\tilde{\boldsymbol{r}}_:,\, \boldsymbol{z}_d,\, I) = \frac{r(\phi_*)\, p(\phi_*|I)}{\int r(\phi)\, p(\phi|I)\, \mathrm{d}\phi}\ .$$

We have

$$\begin{aligned} p(\,\phi_* \mid \boldsymbol{r}_s,\, \boldsymbol{z}_d,\, I\,) &= \iint p(\,\phi_* \mid \varrho'\,)\, p(\,\varrho' \mid \tilde{\boldsymbol{r}}_:,\, I\,)\, p(\,\tilde{\boldsymbol{r}}_: \mid \tilde{\boldsymbol{r}}_s,\, I\,)\, \mathrm{d}\varrho'\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\ &= \iint \varrho'\, \delta(\varrho' - \mathring{\varrho}')\, \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}^{(\tilde{r})}_{:|s}, \mathbf{C}^{(\tilde{r})}_{:|s}\Big)\, \mathrm{d}\varrho'\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\ &= \int \mathring{\varrho}'\, \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}^{(\tilde{r})}_{:|s}, \mathbf{C}^{(\tilde{r})}_{:|s}\Big)\, \mathrm{d}\tilde{\boldsymbol{r}}_: \end{aligned}$$

7.4.1 MAP approximation for log-likelihood function

As before, we might wish to take the MAP approximation for r (7.3.11) to give us

$$p(\,\phi_* \mid \boldsymbol{r}_s,\, \boldsymbol{z}_d,\, I\,) \simeq \frac{m^{(r)}_{*|s}\ p(\phi_*|I)}{\mathfrak{y}^{(r)\,\mathsf{T}}_s\, \mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s}\,, \tag{7.4.2}$$

where $\mathfrak{y}^{(r)}_s$ is of the form expressed in (7.1.7).

7.4.2 Quadrature over log-likelihood functions

Alternatively, if we use our sample log-likelihood functions $(\tilde{\boldsymbol{r}}_:)_l$, and define

$$\begin{gathered} \mathring{\alpha}' \triangleq p(\phi_*|\boldsymbol{r}_s,\, \boldsymbol{z}_d,\, I) \\ \boldsymbol{\varrho}'_l \triangleq \varrho'\big((\tilde{\boldsymbol{r}}_:)_l\big) \triangleq \big[p(\phi_*|\tilde{\boldsymbol{r}}_:,\, \boldsymbol{z}_d,\, I)\,;\, \boldsymbol{r}_: \in (\tilde{\boldsymbol{r}}_:)_l\big]^{\mathsf{T}} \end{gathered}$$

we have

$$p(\,\phi_* \mid \boldsymbol{r}_s,\, \boldsymbol{z}_d,\, \boldsymbol{\varrho}'_l,\, I\,) = m(\mathring{\alpha}' \,|\, \boldsymbol{\varrho}'_l, I) = \mathfrak{y}^{(\varrho')\,\mathsf{T}}_l \mathbf{K}^{(\varrho')}\big((\tilde{\boldsymbol{r}}_:)_l, (\tilde{\boldsymbol{r}}_:)_l\big)^{-1} \boldsymbol{\varrho}'_l \tag{7.4.3}$$

where $\mathfrak{y}^{(\varrho')\,\mathsf{T}}_h$ is of the form (7.1.7) with $\nu^{(\tilde{\boldsymbol{r}}_:)} = \boldsymbol{m}^{(\tilde{r})}_{:|s}$ and $\Lambda^{(\tilde{\boldsymbol{r}}_:)} = \mathbf{C}^{(\tilde{r})}_{:|s}$. (7.4.3) represents our 'best' estimate for (7.4.1). If our sample likelihood functions are again of the form of a conditional GP mean (7.3.15), we have, for $i \in h$

$$\boldsymbol{\varrho}'_i = \varrho'\big((\tilde{\boldsymbol{r}}_:)_i\big) = \frac{m^{(r)}_{*|[s,c]}\ p(\phi_*|I)}{\mathfrak{y}^{(r)\,\mathsf{T}}_{[s,c]}\, \mathbf{K}^{(r)}\left(\boldsymbol{\phi}_{[s,c]}, \boldsymbol{\phi}_{[s,c]}\right)^{-1} \boldsymbol{r}_{[s,c]}}\,,$$

If the posterior distribution for a hyperparameter is to be summarised by a point estimate, we can further employ BQ techniques to determine the posterior mean. This process is described in Appendix D.

7.5 Alternative priors

We now return to what may have appeared a weak link in the reasoning above, our assumption of a Gaussian prior (7.1.2) for ϕ. Clearly this assumption was motivated by our desire for analytic solutions to the integrals that gave us (7.1.5) and (7.1.6). Any prior that can be analytically integrated with respect to some concoction of covariance function is likewise admissable. Examples (for use with a squared exponential covariance) include bounded uniform priors, discrete priors and polynomial kernels. Results corresponding to those in Section 7.3, Section 7.4 and Appendix D for such other priors can be found in Appendix E. Various sums and combinations of such priors are also acceptable, including Gaussian mixture priors.

Of course, even all this flexibility may not prove sufficient – choosing a prior for its analytic properties is an approximation that may not always be justifiable. Choosing a covariance function, representing our beliefs about how f changes with ϕ, purely to match a particular prior forms a similar misuse of probability theory. Take then the very worst case scenario, in which our prior is not at all analytic with respect to the covariance. Here we could use an importance re-weighting trick, re-writing our integrals as per

$$\varphi = \int \left(\frac{f(\phi_*)\, p(\, \phi_* \mid I\,)}{g(\phi_*)} \right) g(\phi_*)\, \mathrm{d}\phi_* \,, \tag{7.5.1}$$

where we treat $f(\phi_*)p(\phi_*|I)\,/g(\phi_*)$ just as we previously treated $f(\phi_*)$, and $g(\phi_*)$ is an appropriate convenient form, such as a Gaussian.

Consider the example in which we take $g(\phi_*) = 1$. This yields $\mathbf{n}_s = h^2\,\mathbf{1}$ and hence a reasonable m_Ψ, but infinite C_Ψ, for any isotropic and stationary covariance. Such covariance functions represent the only sensible choice unless we have exceptional knowledge about how the integrand varies with ϕ. To understand what the GP is

telling us here, note that effectively what we have done is to take a flat, improper prior for ϕ. Hence regardless of the number of ϕ samples we supply it with, the GP can never be at all sure that it has caught all regions where the integrand is significant. We have told it that ϕ space stretches out to infinity in all directions – as far as it knows, the most important lump of integrand mass could be infinitely far removed from the areas it has been told about so far. Given that the correlations the GP assumes hold only between points at a few length scales remove, it has done exactly the right thing in doing its best to report an estimate but warning us that it could be infinitely wrong. The conclusion to draw is that we must employ some restrictive information about the integrand or else be literally infinitely uncertain.

Chapter 8

Sampling for Bayesian Quadrature

8.1 Introduction

We now turn to the choice of sample locations for our quadrature procedures. In particular, we aim to select samples to best evaluate the marginalisation integrals required for prediction (7.3.1) by use of the Bayesian quadrature techniques described in Section 7.3. Note that Bayesian quadrature will effect the correct inference no matter how the samples are obtained. As such, we are liberated to employ any of a number of sampling schemes. We'll outline several such sampling proposals below.

We are particularly interested in sequential applications, which necessarily influences our sampling procedure. For such applications, both the data available to us and the point about which we wish to make predictions are constantly changing. With reference to the predictive problem (7.3.1), $\boldsymbol{z}_d$ and $\boldsymbol{y}_\star$ are constantly changing, and hence so are the numerator and denominator integrands $p(\,\boldsymbol{y}_\star,\, \boldsymbol{z}_d,\, \phi \mid I\,)$ and $p(\,\boldsymbol{y}_d,\, \phi \mid I\,)$. As such, we may not wish to perform a fresh, computationally expensive sampling from these changing integrands every time we wish to update our beliefs. This rules out most traditional sampling schemes.

Also of importance in the context of GP sequential prediction is that the update rules we have described in Section 5.2 hold only for a fixed hyperparameter sample ϕ_i. We can't use computations performed on one ϕ_i in order to update another. Hence, for each new hyperparameter sample we generate we'll have to perform all our expensive GP computation from scratch. This only reinforces the concerns that we

used to motivate Bayesian quadrature in the first place – samples from our integrand are expensive. As such, we must select such samples with care.

8.2 Monte Carlo Sampling

Monte Carlo techniques [Neal, 1993, Loredo, 1999, MacKay, 2002] represent a popular means of sampling in order to approximate integrals. Although widely varied, all attempt to 'randomly sample' from the $p(\,\phi_* \mid I\,)$ in order to estimate the

$$\int f(\phi_*)\, p(\,\phi_* \mid I\,)\; \mathrm{d}\phi_*$$

of (7.1.3) (or sample from (7.3.5) in order to estimate the predictive integrals (7.3.4)). As previously discussed, the estimates typically constructed using such samples suffer from serious flaws. However, we now consider the sampling procedures indepedent from those estimates, with the thought of using samples generated by Monte Carlo techniques for use in Bayesian Quadrature schemes.

The most notable feature of Monte Carlo schemes, the feature that gave rise to their name, is the use of pseudo-random number generators. The schemes aim to generate samples that appear 'randomly drawn' from the appropriate distribution. From a Bayesian perspective, such an approach presents various thorny questions. If a number is random, that simply means we possess some epistemic uncertainty about its value (that is, we are uncertain about the value that will be returned by a random process). How can injecting uncertainty possibly improve our inference procedure? Our aim, after all, is to become more, not less, certain about the value of an integral.

Further, selecting samples is clearly a decision problem (see Section 2.2). We must choose the location of our samples $\boldsymbol{\phi}_s$ in order to approximate our integral as best as possible. The solutions to this problem, as for decision problems in general, are those that minimise our expected loss. The sample sets returned by a Monte Carlo scheme, generated in wilful ignorance, are highly unlikely to represent such solutions. This is particularly true given the complexity and dimensionality of the sampling problem, rendering multiple global minima of the expected loss surface improbable.

Of course, there's no need for us to be at all uncertain about whether our action will be optimal. This uncertainty is an unnecessary consequence of random sampling.

Note also that, in reality, the samples used by Monte Carlo algorithms are produced by algorithmic pseudorandom number generators. There is some non-zero computational cost associated with generating those numbers; we are doing additional work in order to render ourselves more uncertain about the outcome of our algorithm. This problem is accentuated by large numbers of samples traditionally recommended by Monte Carlo practioners, and the practice of simply discarding a not-insignificant number of the initial samples generated (a process known as *burn-in*). The true number of samples to take is the solution to yet another decision problem, but with burn-in, Monte Carlo schemes certainly generate more samples than we actually need. Evaluating our expensive integrand at such samples only aggravates this problem yet further.

The popularity and success of Monte Carlo techniques, can, however be partially understood – they generate samples that have good coverage of the integrand. Our integrand is proportional to the prior $p(\,\phi_* \mid I\,)$; sampling from it ensures that we have many samples where the prior is large, and hence, where our integrand is likely to be large. This is a particular concern for multidimensional integrals, where the problem is complicated by the 'curse of dimensionality' [Duda et al., 2000]. Essentially, the volume of space that could potentially be explored is exponential in its dimension. However, a probability distribution, which must always have a total probability mass of one, will be highly concentrated in this space; ensuring our samples are likewise concentrated is a great boon. Moreover, sampling from the prior ensures a non-zero probability of obtaining samples from any region where the prior is non-zero. This means that we can achieve some measure of both exploitation and exploration of our integrand.

However, balancing exploitation and exploration is entirely possible (and indeed, optimal in a decision theoretic sense) with a deterministic scheme. An example can be found in our GPGO algorithm from Section 6. Using GPGO to optimise an integrand,

although not optimal for quadrature, represents another, deterministic way to give good coverage. Many other deterministic schemes, of various computational costs, could be envisaged along such lines. In order to understand the appeal of Monte Carlo, then, we must turn to a discussion about the use of randomness in general.

For the purposes of thisdiscussion, we consider other popular uses of pseudo-random numbers. Many such applications share the shortcomings outlined above: additional work in order to make us more uncertain about the outcome of our algorithm, unnecessarily; solutions that are unlikely to be of minimal expected loss. Regardless, pseudorandom numbers enjoy a great deal of popularity. Possibly the most ubiquitous example from modern life is the shuffle function on music players. This can be easily understand as providing a counter to boredom. To at least a certain extent, humans find novelty pleasurable. A result, or song, that we are unable to easily predict is a good thing. The pseudo-random generator of the music player, then, is designed such that it is unable to be easily predicted by the human user. Similarly, in cryptography, a randomly generated key is used which is hard for assailants of the cryptographically-protected system to guess. In both examples, a better solution could be found given a sufficiently accurate model of the relevant human agent: with enough information about the human user of a music system, we could determine exactly which song they would find the most pleasurable to hear next. Similarly, for cryptography, with a model of the assailant (marginalised, if necessary, over the many possible assailants), we could determine which key would be the hardest possible to guess. Of course, such analysis is going to be highly computationally demanding. With such costs factored in, approximating what is uncertain for another human agent by something that we, the algorithm designers, are uncertain about, is reasonable.

To go any further, we need to introduce some formality. We characterise a pseudorandom number generator as vector of N numbers, where N is the number of calls we make to it. Of course, for N sufficiently large, allowing for, say, a billion calls a second for a billion years, any practical generator can be so described. The

'randomness' of a particular generator can be characterised in a number of ways, describing its predictability or compressibility. All boil down to the same notion, which we'll generically refer to as structure. Structure is epistemic – an agent that had full access to the seed and algorithm used by a generator (which may not even be considered a great deal of information) would be able to perfectly predict the vector. The randomness of a generator is defined only with respect to a specified agent, which possesses a particular specified set of information. Pseudorandom generators appear to be designed with reference to some sort of typical human agent: their vector is intended to be difficult to compress by a typical human. As an example, the vector of points from $[0, 1]$

$$\boldsymbol{v} = [0.1,\, 0.2,\, \ldots,\, 0.9,\, 1,\, 0.01,\, 0.02,\, \ldots]$$

possesses a structure that a typical human will likely find very predictable, due to our anthropic decimal representation of numbers. If the same vector is passed through an unfamiliar isomorphism, say

$$f : x \to \log\bigl(x + x^2\bigr)$$

the resulting vector[1]

$$f(\boldsymbol{v}) = [-2.20,\, -1.43,\, \ldots,\, 0.54,\, 0.69,\, -4.60,\, -3.89,\, \ldots]$$

might be found harder to predict. This is despite the fact that both vectors can be described by simple rules.

To elaborate upon why such a lack of human structure is useful, consider another common use of random numbers: algorithm testing. Our goal is for a new algorithm to be successful for all possible inputs. As such, the algorithm is tested on our random vector, and the success of the test is taken as an indication of the efficacy of the algorithm in general. Consider an algorithm as the embodiment of a set of information, knowingly or unknowingly (in the case of bugs) encoded by its

[1] While rounded to 2 decimal places here, the hypothetical agent inspecting the vector would have access to as many digits as requested.

programmer. We can model the algorithm as an agent, just as we do humans, and, as the algorithm is coded by humans, it is likely to have a similar information set to one (at least as concerns the algorithm's particular remit). We want the algorithm to find the inputs supplied to it by our generator to be highly unpredictable. The hope is that, as a consequence, the algorithm cannot then inadvertantly exploit special structure in the inputs. It's possible that the algorithm possesses a set of information that will allow it to be successful only for a limited number of inputs, those acknowledged by our algorithm-agent (usually those that were explicitly in the thoughts of the programmer). By giving it our vector, which is hard to predict for a typical human, we are hoping that we rule out such a possibility. We want the structure of the algorithm to be poorly matched by the structure of the vector.

The use of pseudorandom generators to perform sampling for quadrature may be similarly motivated. That is, the goal may be to create a sampling strategy that will be effective for all possible inputs – here, integrands. As an integrand may possess a structure dictated by non-human-like information, it is not entirely unreasonable to use a sampling strategy that will not suffer due to its possession of human-like structure. However, such thinking is short-sighted: our real goal is surely to ensure the best expected coverage of the integrand, where we marginalise over the possible integrands we could receive. Ensuring good coverage of reasonable integrands is readily achieved by deterministic sampling strategies, as above. For a Bayesian algorithm along such lines, the prior information upon which it is built is explicitly acknowledged and tailored to the types of integrand likely to be presented. The pseudo-random alternative is to focus upon pathological cases, as if a malicious agent is responsible for supplying us with the integrand. While an algorithm might be considered an agent with our own goals but with human-like gaps in its knowledge (as for the algorithm testing case above), it is difficult to imagine what is to be gained from treating an integrand as having information akin to a typical human (and so potentially foiled by random selection) but being actively antithetical to our interests. Monte Carlo sampling for quadrature ensures only that we eventually obtain a sample

at every significant point for any integrand, a goal which seems to have been marked as 'success' for this application. Instead, our goal should surely be to optimise the expected performance of a set of samples in estimating an integral.

In summary, the use of pseudo-random generators has some justification in being a low computational-cost method of foiling a typical human's ability to predict. However, such an approach is almost entirely unreasonable in sampling for quadrature applications.

8.3 Initial Grid Sampling

We now consider a very simple sampling scheme [Osborne et al., 2008, 2010b, Garnett et al., 2010a], for use in time-critical sequential applications. As our likelihoods change with every new observation, our first concession is to refrain from sampling over $\boldsymbol{r}_{:}$. Instead, we take the simple MAP approximation for $\boldsymbol{r}_{:}$, giving rise to, for example, (7.3.12). We also take a simple approach to sampling our hyperparameters; sampling them once, off-line, and maintaining those samples throughout. For the purposes of sequential GP inference, this will allow us to use the GP update rules from Section 5.2 for each sample ϕ_i separately, giving us effectively a GP running in parallel for each one. These will then be combined in the Gaussian mixture (7.3.17) to give our final estimates at each time step. The weights $\boldsymbol{\rho}_s$ associated with hyperparameter samples will continue to be revised in light of new observations, rewarding samples that give good fits to the data.

Note that the only computations that are expensive in the number of samples, $|s|$, are the Cholesky decomposition and multiplication of covariance matrices in (7.3.12). These will scale as $O(|s|^3)$. Otherwise, clearly, the evaluations that give $\boldsymbol{q}_{\star,s}$ and $\boldsymbol{r}_s$ for each sample scale only linearly with $|s|$. Fortunately, if we use a fixed set of samples as just proposed, the problematic term $\mathbf{K}^{q_\star}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\mathfrak{Y}_{s,s}^{(q_\star,r)}\mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}$ in (7.3.12) need only be evaluated once, off-line. This is an important boon, rendering it feasible for us to consider a large number of hyperparameter samples to effectively

explore our multi-dimensional integrals.

Yet another substantial saving is found if we are willing to assign independent priors to each hyperparameter. We write $\phi^{(e)}$ for the value of the eth hyperparameter in ϕ. $\phi_i^{(e)}$ is used for the value of the eth hyperparameter in ϕ_i. For each hyperparameter we take an independent prior distribution such that

$$p(\,\phi \mid I\,) \triangleq \prod_e p\big(\,\phi^{(e)} \;\big|\; I\,\big)\;. \tag{8.3.1}$$

We may also be willing to couple this with a covariance structure, over both q and r, of the form (4.2.1),

$$K(\phi_1, \phi_2; \theta) = \prod_e K^{(e)}\Big(\phi_1^{(e)},\, \phi_2^{(e)}; \theta\Big)\,, \tag{8.3.2}$$

so that the covariance function is likewise an independent product of terms over each hyperparameter. We now additionally consider a simple grid of samples, such that $\boldsymbol{\phi}_s$ is the tensor product of a set of unique samples $\boldsymbol{\phi}_u^{(e)}$ over each hyperparameter. Each set of samples $\boldsymbol{\phi}_u^{(e)}$ is deterministically specified given the appropriate prior. For example, we could take uniformly spaced samples centered at the mean of the prior. Alternatively, we could make use of the inverse cumulative distribution function of our prior, taking a set of samples at the quantiles. Note that spacing our samples out in such manners, while giving good coverage of the integrand, also ensures that the conditioning of covariance matrices over $\boldsymbol{\phi}_s$, such as $\mathbf{K}^{q\star}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)$, is not problematically poor.

With such an approach, we have

$$\begin{aligned}\mathbf{K}^{(q_\star)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} &\mathfrak{Y}_{s,s}^{(q_\star,r)} \mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \\ &= \bigotimes_e \mathbf{K}^{(q_\star,e)}(\boldsymbol{\phi}_u^{(e)}, \boldsymbol{\phi}_u^{(e)})^{-1} \mathfrak{Y}_{(s,e),(s,e)}^{(q_\star,r,e)} \mathbf{K}^{(r,e)}(\boldsymbol{\phi}_u^{(e)}, \boldsymbol{\phi}_u^{(e)})^{-1}\,,\end{aligned}$$

so that this problematic term reduces to the Kronecker product of the equivalent term over each individual hyperparameter. We can similarly express as a Kronecker product the $\boldsymbol{\eta}_s^{(r)\mathsf{T}} \mathbf{K}^{(r)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s$ term from (7.3.12) and (7.4.2). This means that we only have to perform the Cholesky factorisation and multiplication of matrices of

size equal to the number of samples for each hyperparameter. This represents one way of evading the 'curse of dimensionality'.

As an example of this initial grid sampling for Bayesian quadrature (GBQ), and how the weights assigned by (7.3.12) are refined in the light of new data, refer back to Figure 6.1.

In limiting ourselves to sampling only once, we have committed to sampling before the first time step, before we have actually seen any data. As such, we are limited to sampling purely from our hyperparameter priors $p(\,\phi \mid I\,)$. This is not unjustifiable – as demonstrated by (7.3.1), these remain as a multiplicative factor in the integrands regardless of how much data is obtained. Hence in ruling out samples where the prior is small, it's reasonable to hope we have only ignored points where the entire integrand would have been similarly insignificant. A more sophisticated scheme is considered in Section 8.4, which allows for the revising of not just the weight assigned to each hyperparameter sample, but also its position.

8.4 Sampling for Bayesian Quadrature

8.4.1 Sampling for simple Bayesian quadrature

Before we can consider a more sophisticated alternative for sampling for the ratio of integrals (7.3.1), we turn to the problem of choosing sample locations $\boldsymbol{\phi}_s$ for the simpler integrals of the form (7.1.1),

$$\mathring{\varphi} \triangleq \int f(\phi_*)\, p(\,\phi_* \mid I\,)\, \mathrm{d}\phi_*\,, \tag{8.4.1}$$

of Section 7.1. As we saw in Section 7.2, only the mean of our belief $p(\,\varphi \mid \boldsymbol{f}_s,\, I\,)$ about probabilistic integrals directly enters our inference.

We now apply the arguments of Section 2.5 to the second-order probability under our consideration. The variance from (7.1.6),

$$C(\mathring{\varphi} \,|\, \boldsymbol{f}_s,\, I) = \mathfrak{C}^f - \mathfrak{y}_s^{(f)\,\mathsf{T}} \mathbf{K}^{(f)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \mathfrak{y}_s^{(f)}\,, \tag{8.4.2}$$

expresses the stability of φ with respect to the information upon which it is conditioned. That is, $C(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I)$ indicates how our estimate for φ would change if we had taken a different sample set ϕ_s. If this variance is minimised, our estimate will be insensitive to the addition of further samples; we will already have sufficient information about the integrand, as desired.

That is, we can use the variance $C(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I)$ as a loss function to induce us to select good locations for the sample set ϕ_s. Explicitly, we select our sample set $\boldsymbol{\phi}_s$ in order to minimise $C(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I)$, following the suggestions of O'Hagan [1991] and Minka [2000]. In that capacity, for (8.4.2), we can ignore $\mathfrak{C}^{(f)}$, which is a constant independent of the choice of sample set $\boldsymbol{\phi}_s$. As such, our goal is the maximisation of

$$\mathfrak{y}_s^{(f)\intercal}\mathbf{K}^{(f)}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\mathfrak{y}_s^{(f)}\,.$$

Note also that for (8.4.2), $h^{(f)}$ represents only a multiplicative constant. As we are interested only in minimising (7.1.6), its exact value is not of great concern. As such, such a multiplicative constant can be safely ignored. We can hence ignore the quadrature-hyperparameter $h^{(f)}$ in considering inference about $\mathring{\varphi}$.

Essentially, minimising $C(\mathring{\varphi}\,|\,\boldsymbol{f}_s,\, I)$ requires finding a well-separated set of samples, ensuring that the domain of the variable of integration is sufficiently covered. The derivatives of the variance with respect to ϕ_s can be determined easily, aiding its minimisation.

8.4.2 Sampling log-likelihood functions

We now return to the problem of selecting sample log-likelihood functions $\tilde{\boldsymbol{r}}_{\boldsymbol{:}}$ in order to evaluate the simple integral (7.3.9) (which is of the form (8.4.1)),

$$\mathring{\alpha} = \int \frac{\int m_{*|s}^{(q_\star)}\, r_*\, p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}{\int r_*\, p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}\,\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{\boldsymbol{:}}; \boldsymbol{m}_{\boldsymbol{:}|s}^{(\tilde{r})}, \mathbf{C}_{\boldsymbol{:}|s}^{(\tilde{r})}\Big)\,\mathrm{d}\tilde{\boldsymbol{r}}_{\boldsymbol{:}}\,. \tag{8.4.3}$$

Our sample functions are chosen with the aim of minimising the variance $C(\mathring{\alpha}\,|\,\boldsymbol{a}_h,\, I)$ (which will be of the form (8.4.2)). As above, then, this entails finding a well-separated sample set. Recall from (7.3.15) that, for analytic convenience, our sample functions

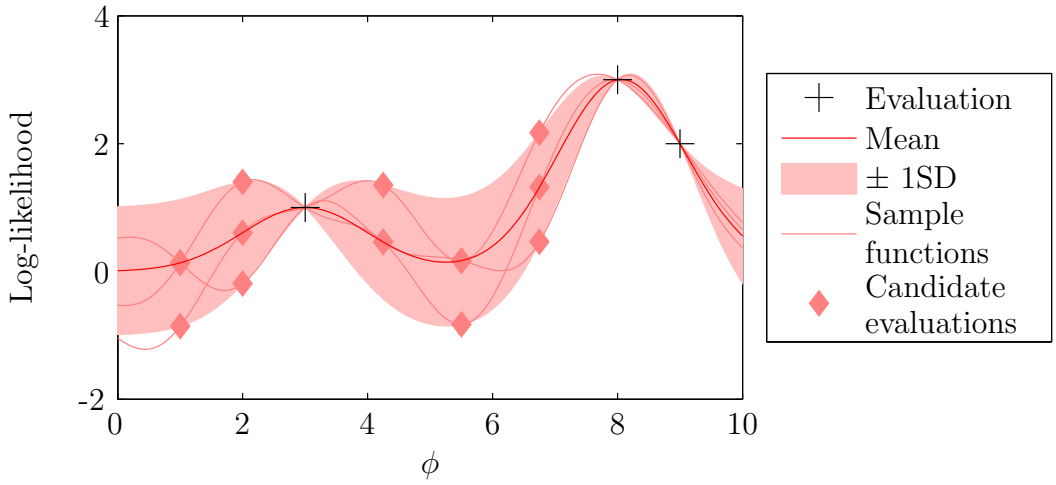


Figure 8.1: A toy example of the selection of five candidate locations, $\boldsymbol{\phi}_c$, and of resulting sample log-likelihood functions.

are taken as GP means conditioned on both our actual likelihood evaluations and the simulated likelihoods at a set of *candidate* points, $\boldsymbol{\phi}_c$. For our candidates $\boldsymbol{\phi}_c$, then, we want points at which we are most uncertain about the likelihood, in order to give us plausible likelihood functions that differ significantly from the mean. Given our smooth GP over the likelihood, the points about which we are most uncertain are typically those furthest from evaluations. We must also ensure, so as to generate even greater separation between sample functions, that such points are well-removed from each other. Such points can be determined through the exploitation of techniques from computational geometry; we select our $\boldsymbol{\phi}_c$ in such a way. Figure 8.1 illustrates the selection of candidate locations.

Given such points, we can now define a covariance function to be used by our GP over the integrand,

$$a(\tilde{\boldsymbol{r}}_{:}) = \frac{\int m_{*|s}^{(q_\star)}\, r_*\, p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}{\int r_*\, p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}\,, \tag{8.4.4}$$

of (8.4.3). It is:

$$\mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_{:})_1, (\tilde{\boldsymbol{r}}_{:})_2\big) \triangleq \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_{c})_1, (\tilde{\boldsymbol{r}}_{c})_2\big) \tag{8.4.5}$$

where

$$\mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_1, (\tilde{\boldsymbol{r}}_c)_2\big) \triangleq \prod_{i \in c} \mathcal{N}\Big((\tilde{r}_i)_1; (\tilde{r}_i)_2, \sigma^2 \mathbf{C}^{(\tilde{r})}_{i|s}\Big) \,.$$

We have reduced a covariance over the whole log-likelihood space to one over the candidate log-likelihoods. Given that the only variability in our sample log-likelihood functions is due to the different candidate log-likelihoods, this is not unreasonable. Here σ, along with $W^{(r)}$, $W^{(q)}$ and $W^{(\tilde{r})}$ (the latter defining $\nu^{(\tilde{\boldsymbol{r}}_:)}$ and $\Lambda^{(\tilde{\boldsymbol{r}}_:)}$) form our set of quadrature hyperparameters. With these, we can determine our estimate (7.3.14) for our ratio of integrals.

We now need to determine the simulated log-likelihoods $\tilde{\boldsymbol{r}}_c$ to take at those candidate locations $\boldsymbol{\phi}_c$. Our goal, of course, is the minimisation of a variance of the form (8.4.2). This is specified by the covariance (8.4.5), expressed over those simulated log-likelihoods $\tilde{\boldsymbol{r}}_c$. As we have now limited the space of all possible likelihood functions $\tilde{\boldsymbol{r}}_:$ (which is infinite-dimensional) to the much smaller space of candidate likelihoods $\tilde{\boldsymbol{r}}_c$, we can envisage performing this minimisation numerically. Specifically, we select $\tilde{\boldsymbol{r}}_c$ in order to minimise the variance $C(\mathring{\alpha} \,|\, \boldsymbol{a}_h, I)$. This minimisation is aided, as mentioned in Section 8.4.1, by the computationally cheap gradient of our variance with respect to the simulated log-likelihoods.

Note, however, that our sample log-likelihood functions $\tilde{\boldsymbol{r}}_:$ are ultimately only of use in allowing us to evaluate a (8.4.4). This term is a ratio of two integrals of likelihood functions $\boldsymbol{r}_:$, the form to which our sample log-likelihood functions must ultimately be transformed. As such, log-likelihoods that are very negative give likelihoods that make very little contribution to a. More specifically, consider the situation in which we have two sets of samples for our simulated log-likelihoods,

$$(\tilde{\boldsymbol{r}}_c)_1 = [1, 2, -10]$$
$$(\tilde{\boldsymbol{r}}_c)_2 = [1, 2, -20] \,.$$

There is unlikely to be any significant difference between a evaluated using $(\tilde{\boldsymbol{r}}_c)_1$ amd a evaluated using $(\tilde{\boldsymbol{r}}_c)_2$; given the large candidate likelihoods from the first

two candidates, the likelihoods given by the third candidate are essentially equally insignificant. Clearly, the magnitudes of the actual evaluated likelihoods $\boldsymbol{r}_s$ is also of significance here – distinguishing between candidate likelihoods that are respectively 10 and 20 orders of magnitude smaller than the highest likelihood evaluated is futile. As such, employing our prior knowledge about the behaviour of the integrand a, we can reduce the search space for the algorithm that is to minimise $C(\mathring{\alpha}\,|\,\boldsymbol{a}_h,\, I)$ to likelihoods of a reasonable dynamic range. In practice, we usually limit ourselves to candidate likelihoods that lie largely between the largest and smallest evaluated likelihoods.

An example of sample log-likelihood functions selected in this way is contained in Figure 8.2a.

8.4.3 Sampling for predictive Bayesian quadrature

Similar techniques can also be used to choose the locations of hyperparameter samples $\boldsymbol{\phi}_s$ for the purposes of evaluating our predictive ratio of integrals (7.3.1). As described in Section 8.4.1, the variance of ϱ represents the uncertainty due to not observing the whole likelihood and prediction surfaces. This forms the loss function driving the decision problem of selecting our hyperparameter samples. Therefore, we choose to minimise the variance of ϱ. We do so around a single $y_\star$, selected as the mean of $p\big(\,\boldsymbol{y}_\star \,\big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, \boldsymbol{z}_d,\, I\,\big)$. Given that this quantity is the most important returned by our prediction algorithm, we want to ensure that the accuracy of our integration is optimised around it. The variance of ϱ is

$$
\begin{aligned}
\Sigma_{\star,s} &\triangleq C\big(\mathring{\varrho}\,\big|\,\tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, I\big) \\
&\triangleq \int \varrho^2\, p\big(\,\varrho \,\big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, I\,\big)\, \mathrm{d}\varrho - \Big(\int \varrho\, p\big(\,\varrho \,\big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, I\,\big)\, \mathrm{d}\varrho\Big)^2 \\
&= \iint m\big(\mathring{\varrho}^2\,\big|\,\boldsymbol{q}_{\star,:},\, \boldsymbol{r}_:,\, I\big)\; p\big(\,\tilde{\boldsymbol{q}}_{\star,:} \,\big|\, \tilde{\boldsymbol{q}}_{\star,s},\, I\,\big)\; p\big(\,\tilde{\boldsymbol{r}}_: \,\big|\, \tilde{\boldsymbol{r}}_s,\, I\,\big)\, \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}\, \mathrm{d}\tilde{\boldsymbol{r}}_: \\
&\qquad - \Big(\iint m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,:},\, \boldsymbol{r}_:,\, I\big)\; p\big(\,\tilde{\boldsymbol{q}}_{\star,:} \,\big|\, \tilde{\boldsymbol{q}}_{\star,s},\, I\,\big)\; p\big(\,\tilde{\boldsymbol{r}}_: \,\big|\, \tilde{\boldsymbol{r}}_s,\, I\,\big)\, \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}\, \mathrm{d}\tilde{\boldsymbol{r}}_:\Big)^2 . \qquad (8.4.6)
\end{aligned}
$$

Of course, finding the true minimum of (8.4.6) over all possible sets of hyperparameter samples is a particularly difficult challenge. We reframe the problem as a sequential decision task. In particular, given an existing sample set, we consider the problem of choosing a single additional *trial* hyperparameter sample ϕ_t; a problem we can resolve by minimising the variance of ϱ. We must actually minimise the *expected* variance of ϱ, as we do not want to have to actually compute the (potentially expensive) likelihood at each trial location in order to determine the relevant variance. The expected variance (also our expected loss) after adding ϕ_t to $\boldsymbol{\phi}_s$ is

$$
\begin{aligned}
&m\Big(\mathring{\Sigma}_{\star,[s,t]} \,\Big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, I\Big) \\
&= \iint p(\tilde{q}_{\star,t}|\tilde{\boldsymbol{q}}_{\star,s}, I) p\big(\tilde{r}_t|\tilde{\boldsymbol{r}}_s, I\big) \\
&\qquad \Big(\iint m\big(\mathring{\varrho}^2 \,\big|\, \boldsymbol{q}_{\star,:}, \boldsymbol{r}_:, I\big)\, p\big(\tilde{\boldsymbol{q}}_{\star,:}|\tilde{\boldsymbol{q}}_{\star,[s,t]}, I\big) p\big(\tilde{\boldsymbol{r}}_:, |\tilde{\boldsymbol{r}}_{[s,t]}, I\big) \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:} \mathrm{d}\tilde{\boldsymbol{r}}_: \\
&\qquad\quad - \Big(\iint m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,:}, \boldsymbol{r}_:, I\big)\, p\big(\tilde{\boldsymbol{q}}_{\star,:}|\tilde{\boldsymbol{q}}_{\star,[s,t]}, I\big) p\big(\tilde{\boldsymbol{r}}_:|\tilde{\boldsymbol{r}}_{[s,t]}, I\big) \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:} \mathrm{d}\tilde{\boldsymbol{r}}_:\Big)^2 \Big) \mathrm{d}\tilde{q}_{\star,t} \mathrm{d}\tilde{r}_t \\
&= \iint m\big(\mathring{\varrho}^2 \,\big|\, \boldsymbol{q}_{\star,:}, \boldsymbol{r}_:, I\big)\, p\big(\tilde{\boldsymbol{q}}_{\star,s}|\tilde{\boldsymbol{q}}_{\star,:}, I\big) p\big(\tilde{\boldsymbol{r}}_:|\tilde{\boldsymbol{r}}_s, I\big) \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:} \mathrm{d}\tilde{\boldsymbol{r}}_: \\
&\qquad - \iint p\big(\tilde{q}_{\star,t}|\tilde{\boldsymbol{q}}_{\star,s}, I\big) p\big(\tilde{r}_t|\tilde{\boldsymbol{r}}_s, I\big) \\
&\qquad\quad \Big(\iint m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,:}, \boldsymbol{r}_:, I\big)\, p\big(\tilde{\boldsymbol{q}}_{\star,:}|\tilde{\boldsymbol{q}}_{\star,[s,t]}, I\big) p\big(\tilde{\boldsymbol{r}}_:|\tilde{\boldsymbol{r}}_{[s,t]}, I\big) \mathrm{d}\tilde{\boldsymbol{q}}_{\star,:} \mathrm{d}\tilde{\boldsymbol{r}}_:\Big)^2 \mathrm{d}\tilde{q}_{\star,t} \mathrm{d}\tilde{r}_t \,.
\end{aligned}
$$

Note that the first term of $m\Big(\mathring{\Sigma}_{\star,[s,t]} \,\Big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, I\Big)$ is independent of ϕ_t and is indeed identical to the first term of $\Sigma_{\star,s}$. As our goal is to compare the relative worth of different selections of ϕ_t, we can translate $\Sigma_{\star,s}$ to remove this additive constant (defining the rescaled variable $\Sigma^{(\mathrm{scld})}_{\star,[s,t]}$) and arrive at

$$
\begin{aligned}
&m\Big(\mathring{\Sigma}^{(\mathrm{scld})}_{\star,[s,t]} \,\Big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, \boldsymbol{f}_{v\times l\times l},\, \boldsymbol{g}_{v\times l\times l},\, I\Big) \\
&\triangleq - \iint \Big(\int m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,[s,t]}, \boldsymbol{r}_:, I\big)\, \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}^{(\tilde{r})}_{:|[s,t]}, \mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\, \mathrm{d}\tilde{\boldsymbol{r}}_:\Big)^2 \\
&\qquad\qquad \mathcal{N}\Big(\tilde{q}_{\star,t}; m^{(\tilde{q}_\star)}_{t|s}, C^{(\tilde{q}_\star)}_{t|s}\Big)\, \mathcal{N}\Big(\tilde{r}_t; m^{(\tilde{r})}_{t|s}, C^{(\tilde{r})}_{t|s}\Big)\, \mathrm{d}\tilde{q}_{\star,t} \mathrm{d}\tilde{r}_t \\
&= - \iiiint m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,[s,t]}, \boldsymbol{r}_:, I\big)\, m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,[s,t]}, \boldsymbol{r}'_:, I\big)\, \mathcal{N}\Big(\tilde{q}_{\star,t}; m^{(\tilde{q}_\star)}_{t|s}, C^{(\tilde{q}_\star)}_{t|s}\Big)\, \mathcal{N}\Big(\tilde{r}_t; m^{(\tilde{r})}_{t|s}, C^{(\tilde{r})}_{t|s}\Big) \\
&\qquad\qquad \mathcal{N}\Big(\tilde{\boldsymbol{r}}_:; \boldsymbol{m}^{(\tilde{r})}_{:|[s,t]}, \mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\, \mathcal{N}\Big(\tilde{\boldsymbol{r}}'_:; \boldsymbol{m}^{(\tilde{r})}_{:|[s,t]}, \mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\, \mathrm{d}\tilde{q}_{\star,t} \mathrm{d}\tilde{r}_t \mathrm{d}\tilde{\boldsymbol{r}}_: \mathrm{d}\tilde{\boldsymbol{r}}'_: \,,
\end{aligned}
$$

where we have approximated

$$\mathcal{N}\Big(\tilde{\boldsymbol{q}}_{\star,:};\boldsymbol{m}^{(\tilde{q}_\star)}_{:|s},\mathbf{C}^{(\tilde{q}_\star)}_{:|s}\Big)\left|\frac{\mathrm{d}\tilde{\boldsymbol{q}}_{\star,:}}{\mathrm{d}\boldsymbol{q}_{\star,:}}\right|\simeq\mathcal{N}\Big(\boldsymbol{q}_{\star,:};\boldsymbol{m}^{(q_\star)}_{:|s},\mathbf{C}^{(q_\star)}_{:|s}\Big) \tag{8.4.7}$$

an assumption identical in purpose and effect to that made in (7.3.10). Referring to (7.3.8), note that

$$\begin{aligned}
m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,[s,t]},\,\boldsymbol{r}_{:},\,I\big) &= \frac{\int m^{(q_\star)}_{*|[s,t]}\,r_*\,p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}{\int r_*\,p\big(\phi_*|I\big)\,\mathrm{d}\phi_*}\\
&= \frac{\int \mathbf{K}^{(q)}(\phi_*,\boldsymbol{\phi}_s)\,r_*\,p(\,\phi_*\mid I\,)\,\mathrm{d}\phi_*}{\int r_*\,p(\,\phi_*\mid I\,)\,\mathrm{d}\phi_*}\mathbf{K}^{(q)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1}\begin{bmatrix}\boldsymbol{q}_{\star,s}\\ q_{\star,t}\end{bmatrix}\\
&\triangleq \boldsymbol{\rho}^{\mathsf{T}}_{\star,s}\,\boldsymbol{q}_{\star,s}+\rho_{\star,t}\,q_{\star,t}\\
m\big(\mathring{\varrho}\,\big|\,\boldsymbol{q}_{\star,[s,t]},\,\boldsymbol{r}'_{:},\,I\big) &\triangleq \boldsymbol{\rho}'^{\mathsf{T}}_{\star,s}\,\boldsymbol{q}_{\star,s}+\rho'_{\star,t}\,q_{\star,t}\,,
\end{aligned}$$

and using the identity

$$\int e^{t\,x}\,\mathcal{N}\big(x;m,\sigma^2\big)\,\mathrm{d}x=e^{\mu\,t+\frac{1}{2}(\sigma\,t)^2}\,,$$

we can write

$$\begin{aligned}
&m\Big(\mathring{\Sigma}^{(\mathrm{scld})}_{\star,[s,t]}\,\Big|\,\tilde{\boldsymbol{q}}_{\star,s},\,\tilde{\boldsymbol{r}}_s,\,I\Big)\\
&=-\iiint\Big(\Big(\boldsymbol{\rho}^{\mathsf{T}}_{\star,s}\,\boldsymbol{q}_{\star,s}+\rho_{\star,t}\,e^{m^{(\tilde{q}_\star)}_{t|s}+\frac{1}{2}C^{(\tilde{q}_\star)}_{t|s}}\Big)\Big(\boldsymbol{\rho}'^{\mathsf{T}}_{\star,s}\,\boldsymbol{q}_{\star,s}+\rho'_{\star,t}\,e^{m^{(\tilde{q}_\star)}_{t|s}+\frac{1}{2}C^{(\tilde{q}_\star)}_{t|s}}\Big)+\rho_{\star,t}\rho'_{\star,t}\chi_t\Big)\\
&\qquad\mathcal{N}\Big(\tilde{r}_t;m^{(\tilde{r})}_{t|s},C^{(\tilde{r})}_{t|s}\Big)\,\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{:};\boldsymbol{m}^{(\tilde{r})}_{:|[s,t]},\mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\,\mathcal{N}\Big(\tilde{\boldsymbol{r}}'_{:};\boldsymbol{m}^{(\tilde{r})}_{:|[s,t]},\mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\,\mathrm{d}\tilde{r}_t\mathrm{d}\tilde{\boldsymbol{r}}_{:}\mathrm{d}\tilde{\boldsymbol{r}}'_{:}\,,
\end{aligned} \tag{8.4.8}$$

where

$$\chi_t\triangleq e^{2m^{(\tilde{q}_\star)}_{t|s}+C^{(\tilde{q}_\star)}_{t|s}}\Big(e^{C^{(\tilde{q}_\star)}_{t|s}}-1\Big)\,. \tag{8.4.9}$$

Here again our integration becomes non-analytic and we are forced to employ a final stage of Bayesian Quadrature. Note that (8.4.8) can be written as the sum of two integrals of the form

$$\begin{aligned}
\beta\triangleq\iiint b\big(\tilde{r}_t,\,\tilde{\boldsymbol{r}}_{:},\,\tilde{\boldsymbol{r}}'_{:}\big)\mathcal{N}\Big(\tilde{r}_t;m^{(\tilde{r})}_{t|s},C^{(\tilde{r})}_{t|s}\Big)\\
\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{:};\boldsymbol{m}^{(\tilde{r})}_{:|[s,t]},\mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\mathcal{N}\Big(\tilde{\boldsymbol{r}}'_{:};\boldsymbol{m}^{(\tilde{r})}_{:|[s,t]},\mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)\mathrm{d}\tilde{r}_t\mathrm{d}\tilde{\boldsymbol{r}}_{:}\mathrm{d}\tilde{\boldsymbol{r}}'_{:}
\end{aligned} \tag{8.4.10}$$

where, for

$$f(\tilde{\boldsymbol{r}}_{:}) \triangleq \boldsymbol{\rho}_{\star,s}^{\mathsf{T}} \, \boldsymbol{q}_{\star,s} + \rho_{\star,t} \, e^{m_{t|s}^{(\tilde{q}_\star)} + \frac{1}{2} C_{t|s}^{(\tilde{q}_\star)}} \tag{8.4.11}$$

$$g(\tilde{\boldsymbol{r}}_{:}) \triangleq \rho_{\star,t} \sqrt{\chi_t} \,, \tag{8.4.12}$$

we have

$$b\big(\tilde{r}_t, \, \tilde{\boldsymbol{r}}_{:}, \, \tilde{\boldsymbol{r}}'_{:}\big) \triangleq f(\tilde{\boldsymbol{r}}_{:}) \, f(\tilde{\boldsymbol{r}}'_{:}) \tag{8.4.13}$$

or

$$b\big(\tilde{r}_t, \, \tilde{\boldsymbol{r}}_{:}, \, \tilde{\boldsymbol{r}}'_{:}\big) \triangleq g(\tilde{\boldsymbol{r}}_{:}) \, g(\tilde{\boldsymbol{r}}'_{:}) \,. \tag{8.4.14}$$

We temporarily consider only the former, (8.4.13), although clearly the results are equally valid for (8.4.14). We place another zero-mean GP over $b\big(\tilde{r}_t, \, \tilde{\boldsymbol{r}}_{:}, \, \tilde{\boldsymbol{r}}'_{:}\big)$, using the covariance

$$\begin{aligned}\mathbf{K}^{(b)}\Big(\big[(\tilde{r}_t)_1, (\tilde{\boldsymbol{r}}_{:})_1, (\tilde{\boldsymbol{r}}'_{:})_1\big], \big[(\tilde{r}_t)_2, (\tilde{\boldsymbol{r}}_{:})_2, (\tilde{\boldsymbol{r}}'_{:})_2\big]\Big) \\ \triangleq \mathbf{K}^{(b)}\big((\tilde{r}_t)_1, (\tilde{r}_t)_2\big)\mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_{:})_1, (\tilde{\boldsymbol{r}}_{:})_2\big)\mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}'_{:})_1, (\tilde{\boldsymbol{r}}'_{:})_2\big) \,.\end{aligned}$$

Here $\mathbf{K}^{(a)}$ is exactly that defined in (8.4.5). This choice is motivated by the reasonable expectation that $f(\tilde{\boldsymbol{r}}_{:})$ will vary with $\tilde{\boldsymbol{r}}_{:}$ in much the same way that $m\big(\mathring{\varrho} \,\big|\, \boldsymbol{q}_{\star,s}, \boldsymbol{r}_{:}, I\big)$ does, given that the former is just the latter with a single additional trial hyperparameter sample ($q_{\star,t} = \exp(m_{t|s}^{(\tilde{q}_\star)} + \frac{1}{2} C_{t|s}^{(\tilde{q}_\star)})$) included. A more flexible choice could be made at the cost of introducing additional quadrature hyperparameters. $\mathbf{K}^{(b)}$ is again of the Gaussian form (7.1.4), with $W^{(b)}$ representing another quadrature hyperparameter. The final quadrature hyperparameter to include in θ is $W^{(\tilde{q})}$, required to determine $m_{t|s}^{(\tilde{q}_\star)}$ and $C_{t|s}^{(\tilde{r})}$.

We now take a set of samples for $\tilde{r}_t$, $(\tilde{r}_t)_v$, which can be determined by minimisation of the variance of the β integral (8.4.10), as discussed in Section 8.4.1. Explicitly, focussing on the integral over $\tilde{r}_t$, we simply select the samples $(\tilde{r}_t)_v$ as per

$$(\tilde{r}_t)_v = \underset{(\tilde{r}_t)_v}{\operatorname{argmax}} \, \mathfrak{y}_v^{(b)\mathsf{T}} \mathbf{K}^{(b)}((\tilde{r}_t)_v, (\tilde{r}_t)_v)^{-1} \mathfrak{y}_v^{(b)} \tag{8.4.15}$$

We use $(\tilde{r}_t)_v$ to generate the grid of samples $(\tilde{r}_t)_v \times (\tilde{\boldsymbol{r}}_c)_l \times (\tilde{\boldsymbol{r}}'_c)_l$. That is, for each different value of the trial likelihood, we take the set of likelihood functions that pass both through it and a grid of candidate likelihoods. Note that the prior $\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{:}; \boldsymbol{m}^{(\tilde{r})}_{:|[s,t]}, \mathbf{C}^{(\tilde{r})}_{:|[s,t]}\Big)$ for log-likelihood functions $\tilde{\boldsymbol{r}}_{:}$ that do not pass through $\tilde{r}_t$ is zero. The locations of the candidate likelihoods are those chosen as described in Section 8.4.2, where we must ensure that our candidate locations $\boldsymbol{\phi}_c$ are well separated from both the evaluated likelihood locations $\boldsymbol{\phi}_s$ and the new trial location ϕ_t.

This grid will allow us highly convenient analytic properties, as per

$$
\begin{aligned}
m\Big(\mathring{\beta}\Big|\, \boldsymbol{b}_{v\times l\times l},\, I\Big) = \Bigg(\Big(\int \mathbf{K}^{(b)}\big((\tilde{r}_t)_v, (\tilde{r}_t)_*\big)\mathcal{N}\Big((\tilde{r}_t)_*; m^{(\tilde{r})}_{t|s}, C^{(\tilde{r})}_{t|s}\Big)\,\mathrm{d}(\tilde{r}_t)_*\Big) \\
\otimes \Big(\int \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_l, (\tilde{\boldsymbol{r}}_c)_*\big)\mathcal{N}\Big((\tilde{\boldsymbol{r}}_c)_*; \boldsymbol{m}^{(\tilde{r})}_{c|[s,t]}, \mathbf{C}^{(\tilde{r})}_{c|[s,t]}\Big)\,\mathrm{d}(\tilde{\boldsymbol{r}}_c)_*\Big) \\
\otimes \Big(\int \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}'_c)_l, (\tilde{\boldsymbol{r}}'_c)_*\big)\mathcal{N}\Big((\tilde{\boldsymbol{r}}'_c)_*; \boldsymbol{m}^{(\tilde{r})}_{c|[s,t]}, \mathbf{C}^{(\tilde{r})}_{c|[s,t]}\Big)\,\mathrm{d}(\tilde{\boldsymbol{r}}'_c)_*\Big)\Bigg)^{\mathsf{T}} \\
\Big(\mathbf{K}^{(b)}\big((\tilde{r}_t)_v, (\tilde{r}_t)_v\big)^{-1} \otimes \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_l, (\tilde{\boldsymbol{r}}_c)_l\big)^{-1} \otimes \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}'_c)_l, (\tilde{\boldsymbol{r}}'_c)_l\big)^{-1}\Big) \\
\Big[f\big((\tilde{r}_t)_i, (\boldsymbol{m}^{(\tilde{r})}_{:|[c,s,t]})_{l,i}\big) \otimes f\big((\tilde{r}_t)_i, (\boldsymbol{m}^{(\tilde{r})}_{:|[c,s,t]})_{l,i}\big);\ i \in v\Big]\,.
\end{aligned}
$$

Here the use of $\mathbf{K}^{(a)}$ has reduced the integrals over $(\tilde{\boldsymbol{r}}_{:})_*$ and $(\tilde{\boldsymbol{r}}'_{:})_*$ to integrals over $(\tilde{\boldsymbol{r}}_c)_*$ and $(\tilde{\boldsymbol{r}}'_c)_*$. Note that the value of the likelihood at the trial location has a negligible influence on the value of the likelihood at the candidate locations. Recall that these are selected such that they are far removed from other points, including the trial location. We can then approximate

$$
\mathcal{N}\Big(\tilde{\boldsymbol{r}}_c; \boldsymbol{m}^{(\tilde{r})}_{c|[s,t]}, \mathbf{C}^{(\tilde{r})}_{c|[s,t]}\Big) \simeq \mathcal{N}\Big(\tilde{\boldsymbol{r}}_c; \boldsymbol{m}^{(\tilde{r})}_{c|s}, \mathbf{C}^{(\tilde{r})}_{c|s}\Big)\ ,
$$

which gives us

$$
\begin{aligned}
m\Big(\mathring{\beta}\Big|\, \boldsymbol{b}_{v\times l\times l},\, I\Big) = \Big(\mathfrak{y}^{(b)\,\mathsf{T}}_v \mathbf{K}^{(b)}\big((\tilde{r}_t)_v, (\tilde{r}_t)_v\big)^{-1}\Big) \\
\Big[\Big(\mathfrak{y}^{(a)\,\mathsf{T}}_l \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_l, (\tilde{\boldsymbol{r}}_c)_l\big)^{-1} f\big((\tilde{r}_t)_i, (\boldsymbol{m}^{(\tilde{r})}_{:|[c,s,t]})_{l,i}\big)\Big)^2;\ i \in v\Big]\,,
\end{aligned}
$$

where $\mathfrak{y}^{(b)\,\mathsf{T}}_v$ is of the form (7.1.7) with $\nu^{(\tilde{r}_t)} = \boldsymbol{m}^{(\tilde{r})}_{t|s}$ and $\Lambda^{(\tilde{r}_t)} = \mathbf{C}^{(\tilde{r})}_{t|s}$. Using this result, and as the mean of a sum is the sum of the means, we can return to (8.4.8), the scaled

Table 8.1: Quantities required to compute the expected variance (8.4.16).

Quantity	Name	Definition
q	Prediction	(7.3.2)
r	Likelihood	(7.3.3)
$\tilde{q}$	log-Prediction	(7.3.6)
$\tilde{r}$	log-Likelihood	(7.3.7)
$(\tilde{\boldsymbol{r}}_c)_l$	Sample candidate log-likelihoods	Section 8.4.2
$(\tilde{r}_t)_v$	Sample log-likelihoods at ϕ_t	(8.4.15)
f		(8.4.11)
g		(8.4.12)
$\boldsymbol{\rho}$	Weights over samples $\boldsymbol{q}$	(7.3.16)
χ_t		(8.4.9)
$\boldsymbol{m}$	Posterior GP mean	(3.3.2)
$\mathfrak{y}$		(7.1.7)
$\mathbf{K}^{(a)}$	Covariance a	(8.4.5)
$\mathbf{K}^{(b)}$	Covariance b	(7.1.4)

expected variance

$$
\begin{aligned}
&m\Big(\mathring{\Sigma}^{(\mathrm{scld})}_{\star,[s,t]} \,\Big|\, \tilde{\boldsymbol{q}}_{\star,s},\, \tilde{\boldsymbol{r}}_s,\, \boldsymbol{f}_{v\times l\times l},\, \boldsymbol{g}_{v\times l\times l},\, I\Big) \\
&\quad = \Big(\mathfrak{y}_v^{(b)\intercal}\, \mathbf{K}^{(b)}\big((\tilde{r}_t)_v, (\tilde{r}_t)_v\big)^{-1}\Big)\Big[\Big(\mathfrak{y}_l^{(a)\intercal}\, \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_l, (\tilde{\boldsymbol{r}}_c)_l\big)^{-1} f\big((\tilde{r}_t)_i, (\boldsymbol{m}^{(\tilde{r})}_{:|[c,s,t]})_{l,i}\big)\Big)^2 \\
&\qquad\qquad + \Big(\mathfrak{y}_l^{(a)\intercal}\, \mathbf{K}^{(a)}\big((\tilde{\boldsymbol{r}}_c)_l, (\tilde{\boldsymbol{r}}_c)_l\big)^{-1} g\big((\tilde{r}_t)_i, (\boldsymbol{m}^{(\tilde{r})}_{:|[c,s,t]})_{l,i}\big)\Big)^2;\ i \in v\Big]. \quad (8.4.16)
\end{aligned}
$$

This is the the expected (scaled) variance of our estimate of the integrals 7.3.1 associated with selecting the additional hyperparameter sample ϕ_t, given that we already have hyperparameter samples at $\boldsymbol{\phi}_s$. Our goal is to minimise this expected loss (8.4.16) to determine the optimal location for ϕ_t. We summarise in Table 8.1 the definitions of quantities required to compute (8.4.16).

8.4.4 Algorithm

We now return to consideration of how to effect our sampling scheme in the context of sequential inference. Here we are faced with the challenge of maintaining an effective set of hyperparameter samples. Given an initial set of such samples (which might be sampled at the quantiles of the prior $p(\,\phi \mid I\,)$), we minimise (8.4.16) in order to

determine the optimal location for a new hyperparameter sample, ϕ_t. That determined, we can then use the same expression to calculate the variance expected after dropping each sample in turn, including the variance of the *no-change* condition in which ϕ_t is itself dropped. The lowest of these expected variances specifies our new hyperparameter set. Figure 8.2 illustrates an example of the consequent action of the SBQ algorithm. If time permits, we can also iterate this *add-one–drop-one* procedure multiple times for each increment of the data, $\boldsymbol{z}_d$. We refer to our procedure as SBQ (sampling for Bayesian quadrature). Algorithm 3 represents our proposal for sequential prediction, although of course very similar schemes are readily devised for other applications, such as sequential global optimisation.

As our sample set only changes in the addition of a few new hyperparameter samples with each increment, GP sequential prediction remains feasible. If desired, we could also maintain a grid of samples as per Section 8.3. This would require that we instead consider adding and removing elements to the sets of samples $\boldsymbol{\phi}_u^{(e)}$ over each hyperparameter $\phi^{(e)}$. If we have two hyperparameters, we would be adding and removing rows and columns to the grid of samples plotted on a Cartesian plane.

Recall that we are motivated by problems for which evaluating the prediction and likelihood functions at a given hyperparameter set is computationally expensive. In such a context, each new sample must be carefully selected so as to both explore and exploit those functions (e.g. to determine points of high likelihood). Additionally, however, this selection must be considerate of each sample's contribution to our sample set. SBQ tackles these problems in a principled, Bayesian manner.

Its clear theoretical foundations are in stark contrast to those of competing Monte Carlo methods. Our quadrature procedure makes full use of all samples, while ignoring any irrelevant information. Further, our selection of our samples requires no poorly motivated pseudo-random sampling, instead we simply select the action that minimises our expected loss. Finally, we require no 'burn-in' period; our samples are optimally selected from the outset.

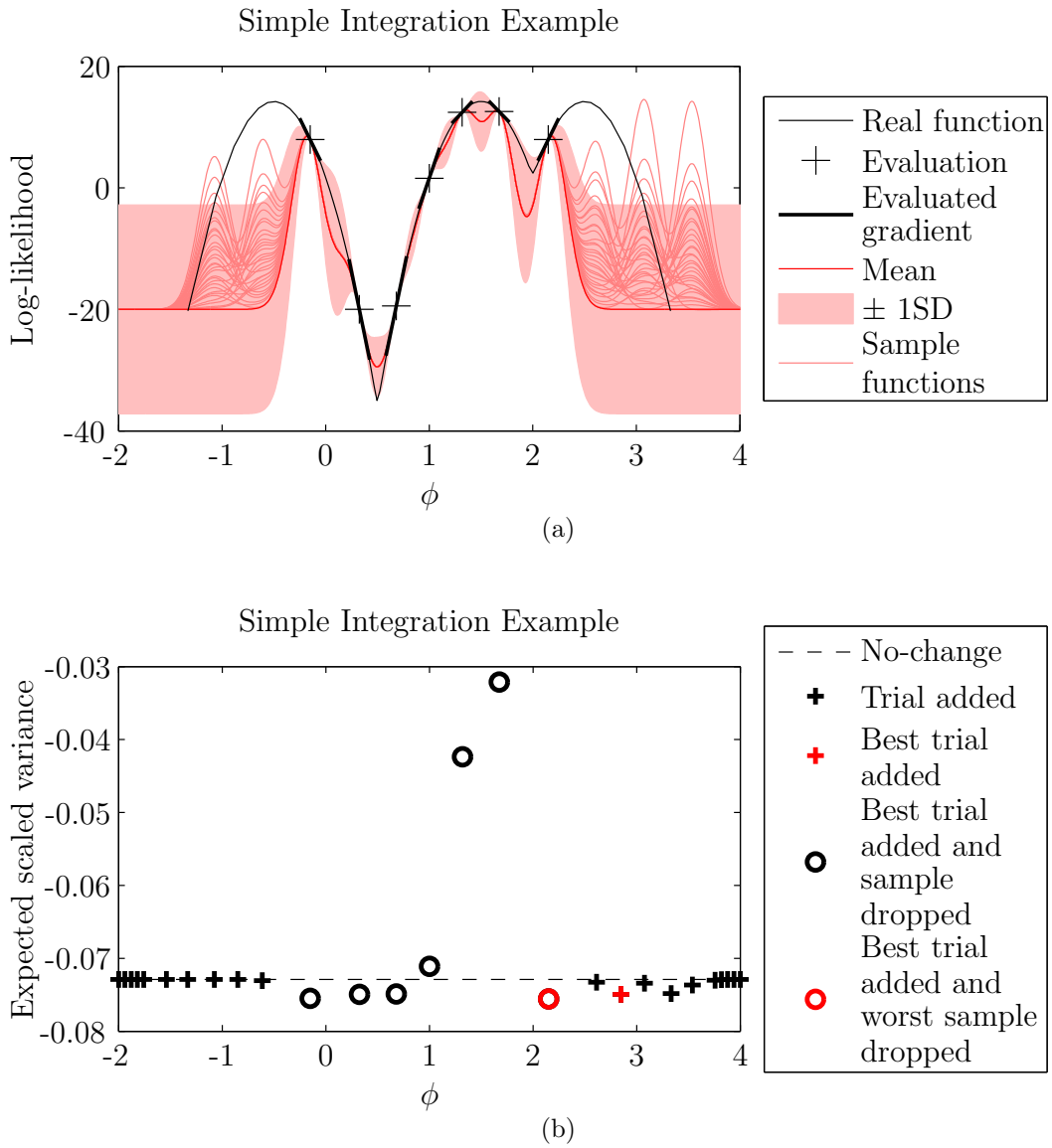


Figure 8.2: An example of the action of the SBQ algorithm on one of the simple Gaussian integration problems from Section 9.6. The prior over ϕ was a Gaussian with mean 1 and standard deviation 1, and, for simplicity, the prediction function $q(\phi)$ was taken as identical to the likelihood function $r(\phi)$. (a) The GP distribution for the log-likelihood surface, along with sample functions formed as GP means conditioned on the log-likelihood at five candidate locations ($\boldsymbol{\phi}_c = [-1.08, -0.61, 2.61, 3.08, 3.54]$). As this is the first iteration of the algorithm, the sequential maximum likelihood procedure we use to specify the quadrature-hyperparameters θ has not yet provided an accurate estimate for the input scale over this surface. (b) The expected scaled uncertainty (8.4.16) after adding a trial hyperparameter ϕ_t at various locations, along with the expected scaled uncertainty after then additionally dropping one of the existing hyperparameter samples $\boldsymbol{\phi}_s$. Note that SBQ drops the existing $\phi = 2.15$ sample in favour of one closer to an unexplored region. The existing sample with highest likelihood is deemed least worthy of being dropped.

8.4.5 Implementation details

Our fundamental assumption is that our likelihood surface is unlikely to change significantly with an increment of $\boldsymbol{z}_d$; that is, the hyperparameter samples from the last increment are a reasonable guide for the best locations of samples for this increment. However, only the likelihoods for the current increment are employed in our analysis. In principle, there is nothing preventing us from placing a covariance over the increment, such that we could exploit the information from older likelihoods. This would, however, introduce some numerical conditioning issues. For example, we would expect our evaluation of the likelihood for a particular sample ϕ_s at both the current increment and that increment immediately preceding it to be highly correlated. Covariance matrices over such evaluations are consequently prone to poor conditioning. As such, we use only evaluations from the current increment in our procedure.

In a similar vein, note that we incorporate into r_s not just evaluations of the likelihood surface, but also of its derivatives. This requires only a trivial modification of the covariance function (see Section 4.7) used for Bayesian quadrature, and has previously been performed by O'Hagan [1992]. Gradient evaluations supply information about the local neighbourhood of a hyperparameter set; thus giving invaluable guidance as to the selection of samples for the next increment. Clearly this relies upon our assumption that the likelihood is smooth over increments. However, such an assumption is invalidated for the prediction surface $q(y_\star, \phi)$, which is liable to undergo dramatic changes, as the point about which we are making predictions, $x_\star$, changes completely from one increment to the next. Therefore, we do not include derivative evaluations of this surface. While certainly possible, doing so would be computationally expensive and unlikely to improve the sample selection process.

We also do not sample over the possible values of the derivatives at the trial location. While we are hence slightly underestimating how informative the new sample will be, the additional variables we would be required to numerically sample over present too great a computational burden.

Finally, we employ various maximum-likelihood schemes for all our quadrature hyperparameters. Several of these estimates can be determined incrementally using sequential gradient ascent techniques. Our candidate points $\boldsymbol{\phi}_c$ are selected as the vertices of the Voronoi diagram of the set of samples $\boldsymbol{\phi}_s$. We limit their locations to the box defined by the hyperparameter prior mean plus or minus three standard deviations.

At those candidate points, we then take simulated log-likelihoods are described in Section 8.4.2. Note that the optimal designs for combinations of candidate log-likelihoods can be determined in advance, and stored for all future runs of the SBQ algorithm. We determine the optimal design for each possible number of candidates (we never take more than 10), expressed in terms of the numbers of standard deviations above or below the mean at each location. We artifically limit each candidate log-likelihood to the range between the mean and the mean plus three standard deviations, as per the discussion in Section 8.4.2. The standard deviations and means here can then be readily substituted in by those of the GP conditioned on the actual evaluated log-likelihoods, as required.

Our goal is to find the sample position that is most likely to minimise the expected uncertainty (8.4.16). For the purposes of this minimisation, we are fortunately not completely ignorant. We expect that the uncertainty will be lower both if we have a widely dispersed set of samples (so that the domain is well explored) and if we have samples at locations where the integrands are significant (so that we can be more confident of having captured most of the integral's mass). To meet the first criterion, it seems reasonable to try adding those points farthest away from all existing hyper-samples. Fortunately, of course, such points have already been determined by reference to the Voronoi diagram, as described above. Of course, if we take a trial at the location ϕ_t, we then do not take a candidate at that location. To address the second criterion, we try adding points near to existing high likelihood hypersamples. More precisely, given that we already have the likelihood gradient at those points, we try points removed by a single step of gradient ascent from those hypersamples.

We then perform local minimisation of the uncertainty around the trial hypersamples comprised of those two different types. If desired, more sophisticated methods of optimising the expected loss surface could be used, such as GPGO, at the cost of a greater computational expense.

Algorithm 3 SBQ-TRACK($\mathrm{Q}(\cdot,\cdot,\cdot)$,$\mathrm{R}(\cdot,\cdot)$, $\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \delta, \boldsymbol{\phi}_s, \boldsymbol{\theta}_{qs}$)

// Returns the sequential lookahead predictions made with SBQ.
// $\mathrm{Q}(\cdot,\cdot,\cdot)$: Prediction function, takes predictant locations,
// hyperparameter and data as arguments.
// $\mathrm{R}(\cdot,\cdot)$: Likelihood function, takes hyperparameter and data
// as arguments.
// $\{(\boldsymbol{x}_t, z_t)\}_{t=1}^{T}$: Initial data.
// δ: Lookahead.
// $\boldsymbol{\phi}_s \triangleq \{\phi_j\}_{j=1}^{J}$: Hyperparameter samples (HSs).
// $\boldsymbol{\theta}_{qs} \triangleq \{\theta_k\}_{k=1}^{K}$: Quadrature hyperparameter samples (QHSs).

for $t = 1$ **to** $T - \delta$ **do**
  for $j = 1$ **to** J **do**
    $r_j \leftarrow \mathrm{R}(\phi_j, \{(\boldsymbol{x}_{t'}, z_{t'})\}_{t'=1}^{t})$ // Calculate HS likelihood.
  end for
  for $k = 1$ **to** K **do**
    QHS-likelihoods$[k] \leftarrow$ CALCULATE-QHS-LIKELIHOOD$(\theta_k, (\boldsymbol{\phi}_s, \boldsymbol{r}_s))$ // Equal to $\mathcal{N}\big(\tilde{\boldsymbol{r}}_s; 0, K^{(\tilde{r})}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s, \theta_k)\big)$, evaluating the fit for $\tilde{r}$ surface under QHS θ_k (using BQ GP model for r).
  end for
  $m \leftarrow \mathrm{argmax}_k\{$*QHS-likelihoods*$[k]\}$
  $\{\boldsymbol{\phi}_c\} \leftarrow$ DETERMINE-CANDIDATES$(\boldsymbol{\phi}_s, \theta_m)$ // Using Voronoi diagram, find candidates that are well-removed from samples.
  $\boldsymbol{\rho} \leftarrow$ DETERMINE-HS-WEIGHTS$((\boldsymbol{\phi}_s, \boldsymbol{r}_s), \boldsymbol{\phi}_c, \theta_m)$ // Using (7.3.16).
  $\boldsymbol{\rho}' \leftarrow$ DETERMINE-LIKELIHOOD-FUNCTION-WEIGHTS$((\boldsymbol{\phi}_s, \boldsymbol{r}_s), \boldsymbol{\phi}_c, \theta_m)$ // Using (7.3.14).
  for $j = 1$ **to** J **do**
    $q_{t+\delta,j} \leftarrow \mathrm{Q}(\boldsymbol{x}_{t+\delta}, \phi_j, \{(\boldsymbol{x}_{t'}, z_{t'})\}_{t'=1}^{t})$ // Calculate HS predictions; the predictant locations $\boldsymbol{x}_\star$ are $\boldsymbol{x}_{t+\delta}$ in this context.
  end for
  prediction[t] $\leftarrow$ COMBINE-PREDICTIONS$(\boldsymbol{q}_{t+\delta,s}, \boldsymbol{\rho}, \boldsymbol{\rho}')$ // Combined in a weighted mixture, as per (7.3.14) and (7.3.16).
  Return: *prediction[t]*
  $\boldsymbol{\phi}_s \leftarrow$ MANAGE-HSS$(\boldsymbol{x}_{t+\delta}, (\boldsymbol{\phi}_s, \boldsymbol{r}_s), \boldsymbol{\phi}_c, \theta_m)$ // Add and drop HSs by minimising (8.4.16), while ensuring that our covariance matrices over samples remain well-conditioned.
  $\boldsymbol{\theta}_{qs} \leftarrow$ MANAGE-QHSS$(\boldsymbol{\theta}_{qs},$*QHS-likelihoods*$)$ // Perform gradient ascent in QHS-likelihood space to move QHSs.
end for

Chapter 9

Empirical Evaluation

9.1 Dataset Navigator

We now present the results of a number of different empirical tests of the methods proposed above. These methods can be combined and used in different ways, and, as such, a large number of tests are possible. Table 9.1 outlines the nature of the tests actually undertaken. The efficacy of the SBQ algorithm from Section 8.4 could be illustrated in a vast number of contexts, but for conciseness we chose only to perform tests for the two most generic problem types, prediction and optimisation using traditional covariance functions. Other possible types of test (GPGO using changepoint covariances, prediction and active data selection (ADS) with a covariance function over sets) were not undertaken due to an absence of suitable data or problem motivation.

All illustrations of predictive performance contained in this chapter use the

Table 9.1: The section numbers for results of various types.

Gaussian Process Type		Problem Type		
Covariance Function	Sampling	Prediction	ADS	GPGO
Standard	GBQ	9.2	9.4.1-9.4.2	9.5.1
	SBQ	9.6.1		9.6.2
Changepoint	GBQ	9.3	9.4.3-9.4.4	
Sets	GBQ			9.5.4

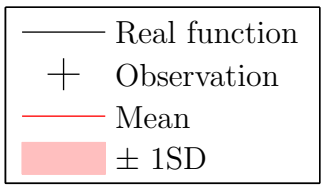


Figure 9.1: The legend used for all figures illustrating prediction in this chapter.

common legend displayed in Figure 9.1. The meaning of the different terms varies slightly with context, and will be defined as required. Where appropriate, predictive quality will be evaluated using the popular root mean squared error (RMSE), defined as

$$\text{RMSE}(\boldsymbol{m}, \boldsymbol{y}) = \sqrt{\frac{1}{N}\sum_{i}^{N}(m_i - y_i)^2}\,.$$

Here $\boldsymbol{m}$ is a vector of N estimates for the vector of variables $\boldsymbol{y}$. For probabilistic methods, the estimates we use are the means for $\boldsymbol{y}$ conditioned on some set of observations $\boldsymbol{z}$ (as dependent on the application). Note that for such probabilistic methods, the most natural performance metric is the evidence [MacKay, 2002], $p(\,\boldsymbol{z} \mid I\,)$ (also known as the *marginal likelihood*). This expresses how well our models and assumptions I actually explain the data $\boldsymbol{z}$, and, unlike the RMSE, reflects the quality of our predictions beyond simply the accuracy of their point estimates. That is, it explicitly captures the predictive uncertainty. If we assign equal prior probabilities to all possible sets of models and assumptions, the evidence, proportional to the posterior probability $p(\,\boldsymbol{z} \mid I\,)$, gives a means of ranking all those sets of models and assumptions. However, we choose the RMSE to allow us to compare both probabilistic and non-probabilistic methods.

All priors for the various problems considered were formed in the same manner. We again write $\phi^{(e)}$ for the value of the eth hyperparameter in ϕ. $\phi_i^{(e)}$ is used for the value of the eth hyperparameter in ϕ_i. For each hyperparameter we take an

independent prior distribution, as per (8.3.1) such that

$$p(\,\phi \mid I\,) \triangleq \prod_e p\big(\,\phi^{(e)} \;\big|\; I\,\big)\;.$$

We also take covariance over q and r that are likewise products of independent terms over each hyperparameter, as per (8.3.2). These choices will allow us, where desired, to benefit from the convenient analytic properties outlined in Section 8.3. For any real hyperparameter $\phi^{(e)}$, we take a Gaussian prior

$$p\big(\,\phi^{(e)} \;\big|\; I\,\big) = \mathcal{N}\big(\phi^{(e)}; \nu_e, \lambda_e^2\big)\;; \tag{9.1.1}$$

if our hyperparameter is restricted to the positive reals, we instead assign a Gaussian distribution to its logarithm. For a hyperparameter $\phi^{(e)}$ known only to lie between two bounds l_e and u_e, we take the uniform distribution over that region,

$$p\big(\,\phi^{(e)} \;\big|\; I\,\big) = \frac{\sqcap(\phi^{(e)}; l_e, u_e)}{\Delta_e}\,, \tag{9.1.2}$$

where $\Delta_e \triangleq u_e - l_e$ and $\sqcap(\theta; l, u)$ is used to denote the rectangular function

$$\sqcap\,(\phi^{(e)}; l_e, u_e) \triangleq \begin{cases} 1 & (l_e < \phi^{(e)} < u_e) \\ 0 & (\text{otherwise}) \end{cases}\;. \tag{9.1.3}$$

Occasionally we may also want to consider a discrete hyperparameter $\phi^{(e)}$. In this case we will take the uniform prior

$$P\big(\,\phi^{(e)} \;\big|\; I\,\big) = \frac{1}{\Delta_e}\,, \tag{9.1.4}$$

where Δ_e is here defined as the number of discrete values the hyperparameter can take.

The parameters for these priors were chosen as appropriate to the problem. If not explicitly mentioned otherwise, we used the simple GBQ algorithm to sample our hyperparameters, as per Section 8.3. Using the marginalisation techniques presented in Chapter 7, we can determine the weights associated with different hyperparameter samples, as per (7.3.17). It is commonly the case that after a significant volume of data, the majority of the weight will be concentrated at a single trial hyperparameter

sample – the posterior distribution for the hyperparameter $p(\phi \mid \boldsymbol{z}_d, I)$ will be close to a delta function at that value. In this case, we will say, informally, that we have *learned* the hyperparameter to have that single value. Where this appears to be the case, performing a local optimisation of the likelihood $p(\boldsymbol{z}_d \mid \phi, I)$ can help to find the point of highest posterior density.

9.2 Online Prediction

In this section we will test the use of the efficient, online GP framework of Chapters 3-5, using GBQ (Section 8.3) for our management of hyperparameters. Specifically, the depicted predictions about time t are determined using all data obtained prior to $t-\epsilon$, where ϵ is the lookahead, specified for the relevant problem. We will compare the use of a multi-output GP formalism built around the multi-output covariance (4.2.3) against independent GPs such that the readings from each sensor are modelled separately (i.e. correlations between sensors are ignored). We also perform a comparison against a Kalman Filter [Jazwinski, 1970, Lee and Roberts, 2008], in particular, a simple dynamic auto-regressive model. We also test against the naïve algorithm that simply predicts that the variable at the next time step will be equal to that most recently observed at that sensor.

The empirical evaluation is performed in the context of a network of weather sensors located on the south coast of England[1] [Rogers et al., 2008, Osborne et al., 2008, 2010b]. This network consists of four sensor stations (named Bramblemet, Sotonmet, Cambermet and Chimet), each of which measures a range of environmental variables (including wind speed and direction, air temperature, sea temperature, and tide height) and makes up-to-date sensor measurements available through separate web pages (see `http://www.bramblemet.co.uk`). The Bramblemet station itself is depicted in Figure 9.2.

[1]The network is maintained by the Bramblemet/Chimet Support Group and funded by organisations including the Royal National Lifeboat Institution, Solent Cruising and Racing Association and Associated British Ports.

Figure 9.2: The Bramblemet weather station.

Weather data is an attractive test space, as it presents a variety of challenging and relevant online prediction problems. Of course, weather variables are generally well-predicted by complicated physical models built upon historical data, giving rise to published weather forecasts and tide tables. However, these predictions usually do not account for unusual, local conditions, which we can detect using our sensors. This information can then be exploited by our framework to produce accurate local forecasts. Moreover, our predictions are made using a full probabilistic model, giving a measure of uncertainty in our predictions, valuable information to users of the weather sensor network.

We will present results for a variety different weather variables: air pressure, tide height, air temperature and wind speed. Note that we have not considered inference over multiple weather variables simultaneously e.g. considering the correlations that might exist between air pressure and wind speed. There is, however, no theoretical obstacle that would prevent us doing so – these would simply represent additional outputs, over which a covariance function could be formed. This would be expected to further improve the performance of our predictions.

The Bramblemet network possesses characteristics that have motivated many of our methods described above. In particular, our four sensor stations are not separated

by more than a few tens of kilometres and hence can be expected to experience reasonably similar weather conditions. As such, their readings are likely to display a strong degree of correlation. However, there may be a slight difference in the times at which each station experiences, say, a particular weather front, as it moves along the southern coast of England. Hence a sensor's readings may exhibit some latency relative to another's.

The network is also subject to various sensor faults and network outages. A new sensor reading is recorded every minute and stored, while every five minutes a reading is transmitted and eventually uploaded to the world wide web. The former, fine-grained readings form our 'real function', and the latter, transmitted readings form our observations. The transmission is prone to temporary failures, giving rise to missing observations. However, after the event, we are able to use the real readings downloaded from the sensors directly to evaluate the accuracy of predictions made using only the transmitted observations. Note also that all acquired readings are a rounded version of the true data. In principle, such observations should be treated as censored observations as discussed in Section 4.6.

9.2.1 Air pressures

Air pressure was chosen as a test set due to the comparatively severe rounding (to the nearest Pascal) it is subject to. Figure 9.3 demonstrates 20-minute-lookahead prediction over the rounded air pressure observations from the Bramblemet sensor alone. We used two GP models (with a Matérn covariance of the form (3.2.5)). One treated the rounded observations as censored using the approach of Section 4.6 and using numerical integration [Genz, 1992] to approximate (4.6.5), the other assumed that the observations were corrupted by simple Gaussian noise of unknown variance. There was no significant improvement noted using the more accurate, censored noise model (for example, RMSEs on a five-day dataset were identical to five significant figures). However, the censored noise model presents a far more challenging computational load, as we must numerically integrate over a large number of correlated,

Table 9.2: RMSEs for five-day air pressure dataset.

Algorithm	RMSE (Pa)
Naïve	3.6068
Kalman filter	3.2900
Non-censored GP	0.3851
Censored GP	0.3851

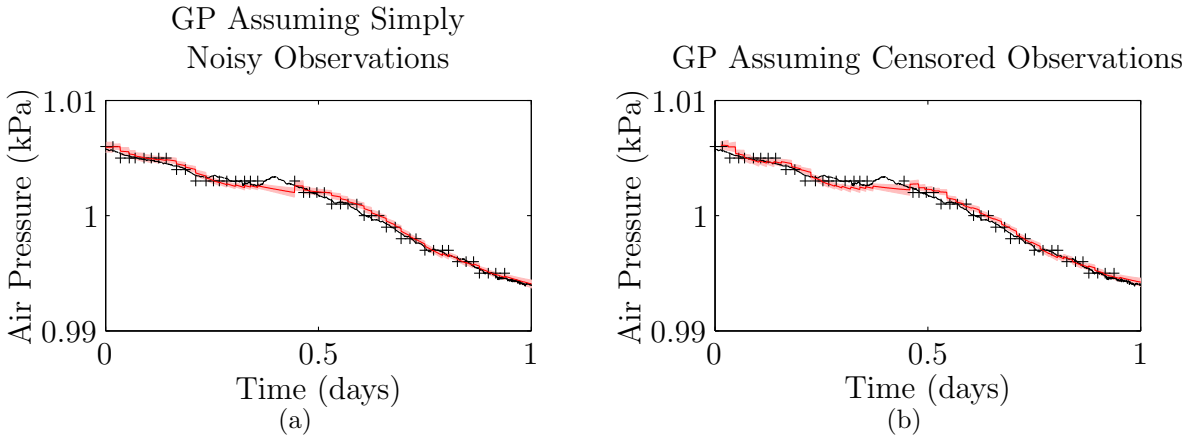


Figure 9.3: There is little perceptible difference between the predictions made for air pressure assuming (a) that the observations were corrupted by simple Gaussian noise and those made assuming (b) that the observations were rounded to the nearest Pascal.

censored observations. As such, for all other testing performed, we considered only Gaussian observation models.

Our GP approach does provide a dramatic improvement over Kalman Filter prediction. The RMSEs for the various algorithms are listed in Table 9.2. Both the GP and Kalman Filter have an order of 16; that is, they store only up to the 16 most recent observations.

9.2.2 Tide heights

With its clear periodic behaviour, a tide height dataset allows us to demonstrate the power of periodic covariances, as discussed in Chapter 4. Further, the dataset depicted in Figure 9.4 contains an interesting period in which extreme weather conditions (a

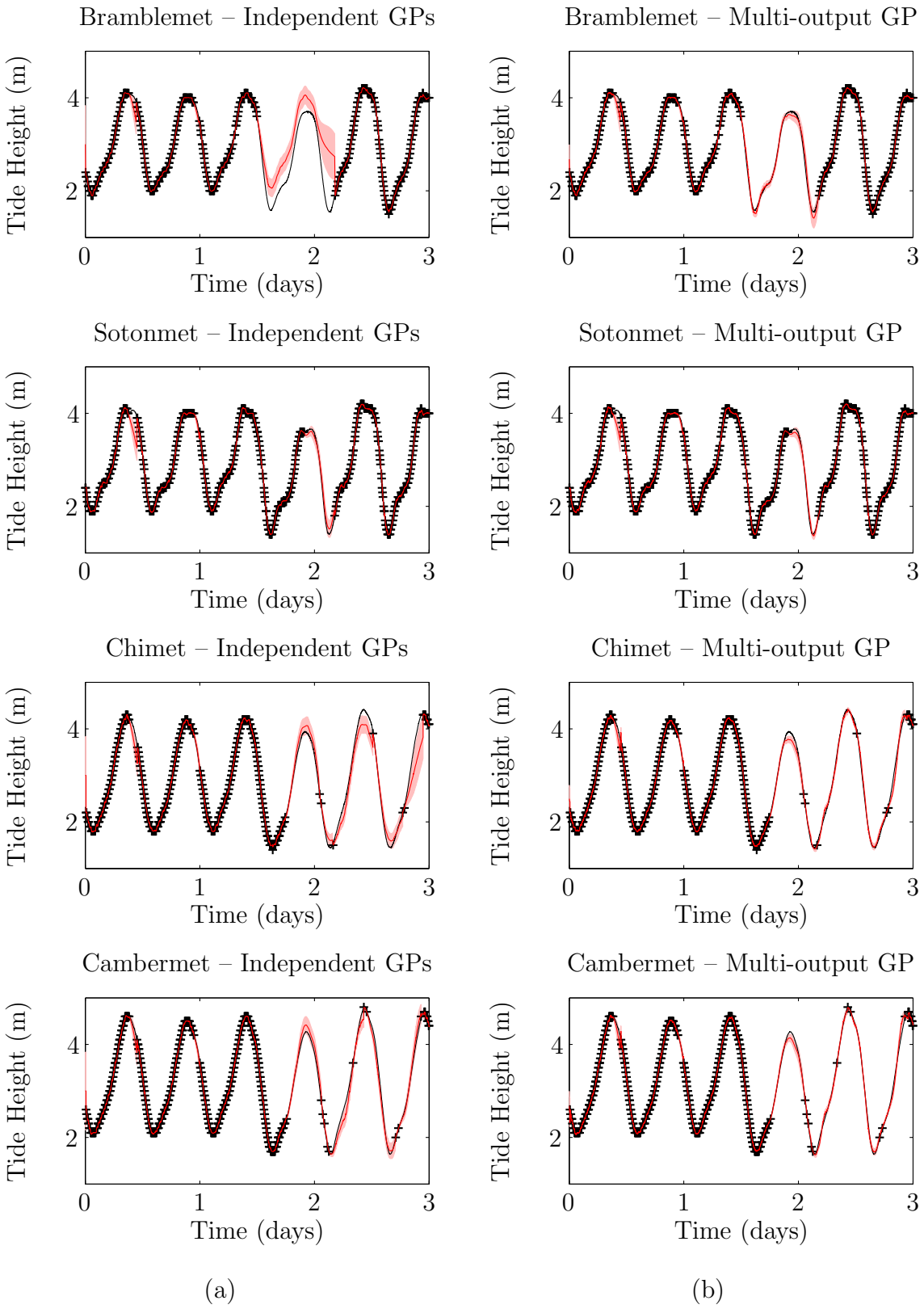


Figure 9.4: Prediction and regression of tide height data for (a) independent and (b) multi-output Gaussian processes.

Northerly gale) cause both an unexpectedly low tide and a failure of the wireless connection between sensors and the shore. Hence our algorithm must cope with periods of missing data.

Figure 9.4 illustrates the efficacy of zero-lookahead GP prediction in this scenario. In order to manage the four outputs of our tide function (one for each sensor), we take the approach embodied in (4.2.3). That is, we rewrite so that we have a single output and inputs t, time, and l, a sensor label. As such, our covariance function is of the form

$$K\big([t_1,\, l_1],\, [t_2,\, l_2]\big) = W(l_1,\, l_2)\,\Big(K^{(\text{per-Mtn})}\big(t_1 - \Delta_{l_1}, t_2 - \Delta_{l_2}; h_P = 1,\, w_P, \nu = \tfrac{5}{2}\big) + K^{(\text{Mtn})}\big(t_1 - \Delta_{l_1}, t_2 - \Delta_{l_2}; h_D,\, w_D, \nu = \tfrac{5}{2}\big)\Big), \quad (9.2.1)$$

where W is a covariance matrix over sensors formed using the spherical decomposition (4.2.4). We used two different GP models: one in which the parameters of the spherical decomposition were inferred from the data; and one for which the matrix S was taken as an identity matrix. In the former case, we consequently have a multi-output formalism, in the latter, independent GPs for each sensor.

Note that our covariance over time is the sum of a periodic term (using (4.3.2)) and a *disturbance* term. Both are of the Matérn form (3.2.4) with $\nu = \frac{5}{2}$ and an appropriate modification made to allow for latency between sensors, as per (4.2.6). This form is a consequence of our expectation that the tides would be well modelled by the superposition of a simple periodic signal and an occasional disturbance signal due to exceptional conditions. Of course, for a better fit over the course of, say, a year, it would be possible to additionally incorporate longer-term drifts and periods.

The period w_P of the first term was unsurprisingly learnt as being about half a day, whereas for the disturbance term the time scale w_D was found to be about two and a half hours. Note that this latter result is concordant with our expectations for the time scales of the weather events we intend our disturbance term to model.

Our algorithm learned that all four sensors were very strongly correlated, with $S^\mathsf{T}S$ (where S is from (4.2.5)) containing elements all very close to one. W addition-

ally gives an appropriate length scale for each sensor. Over this data set, the Chimet sensor was found to have a length scale of 1.4m, with the remainder possessing scales of close to 1m. Note that h_P and h_D in (9.2.1) are dimensionless ratios, serving as weightings between our various covariance terms. Hence we can set $h_D = 1$ without loss of generality, upon which we learn that h_P is around 0.2. Hence weather events were observed to have induced changes in tide height on the order of 20%.

We also make allowances for the prospect of relative latency amongst the sensors by incorporating delay variables, represented by the vector Δ. We found that the tide signals at the Cambermet station were delayed by about 10 minutes relative to the others. This makes physical sense – the Bramblemet and Sotonmet stations are quite exposed to the ocean, while water has to run further through reasonably narrow channels before it reaches the Cambermet station.

Only the correlations amongst sensors and their individual delays were sampled over, all other hyperparameters for our covariance were learned by the GP prior to the depicted data set. Such hyperparameters were then specified with single samples at the learned values.

Note the performance of our multi-output GP formalism when the Bramblemet sensor drops out at $t = 1.45$ days. In this case, the independent GP quite reasonably predicts that the tide will repeat the same periodic signal it has observed in the past. However, the GP can achieve better results if it is allowed to benefit from the knowledge of the other sensors' readings during this interval of missing data. Thus, in the case of the multi-output GP, by $t = 1.45$ days, the GP has successfully determined that the sensors are all very strongly correlated. Hence, when it sees an unexpected low tide in the Chimet sensor data (caused by the strong Northerly wind), these correlations lead it to infer a similarly low tide in the Bramblemet reading. Hence, the multi-output GP produces significantly more accurate predictions during the missing data interval, with associated smaller error bars. Exactly the same effect is seen in the later predictions of the Chimet tide height, where the multi-output GP predictions use observations from the other sensors to better predict the high tide

Table 9.3: RMSEs for five-day tide height dataset.

Algorithm	RMSE (m)
Naïve	7.5×10^{-1}
Kalman filter	3.9×10^{5}
Independent GPs	2.1×10^{-2}
Multi-output GP	9.2×10^{-3}

height at $t = 2.45$ days.

Note also that there are brief intervals of missing data for all sensors just after both of the first two peak tides. During the second interval, the GP's predictions for the tide are notably better than for the first – the greater quantity of data it has observed by the time of the second interval allow it to produce more accurate predictions. With time, the GP is able to build successively better models for the series.

The RMSEs for our various algorithms over this dataset can be found in Table 9.3.

9.2.3 Air temperatures

We next tested on air temperature data, chosen as it exhibits a very different structure to the tide height data.

Figure 9.5 demonstrates our algorithm's 5 minute lookahead predictive performance, where 5 minutes was also the usual (although not exclusive) time increment from one datum to the next. Note that the Cambermet station does not record air temperature, and is hence discluded from consideration here. Taking the covariance function used for tide height data (9.2.1) as a model, we assumed the covariance

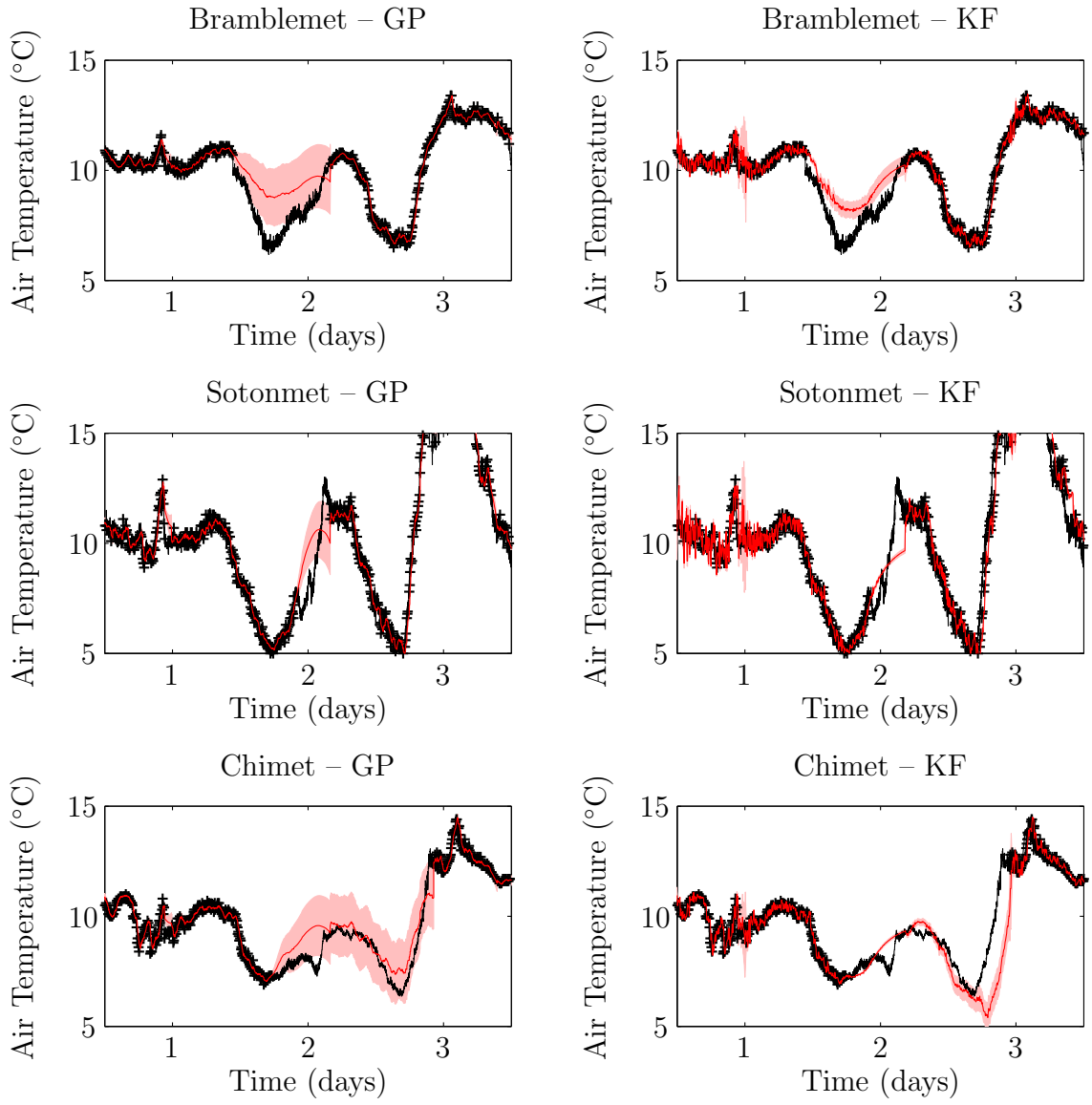


Figure 9.5: 5 minute lookahead prediction of air temperature data for (a) a multi-output Gaussian process and (b) a Kalman filter.

Table 9.4: RMSEs for five-day air temperature dataset.

Algorithm	RMSE (°C)
Naïve	3.82
Kalman filter	0.92
Multi-output GP	0.74

function

$$K\big([t_1,\, l_1],\, [t_2,\, l_2]\big) = W(l_1,\, l_2) \Big(K^{(\text{per-Mtn})}\big(t_1 - \Delta_{l_1}, t_2 - \Delta_{l_2}; h_P = 1,\, w_P, \nu = \tfrac{5}{2}\big) + K^{(\text{Mtn})}\big(t_1 - \Delta_{l_1}, t_2 - \Delta_{l_2}; h_T,\, w_T, \nu = \tfrac{5}{2}\big)\Big) + \delta(l_1 - l_2)\, K^{(\text{Mtn})}\big(t_1, t_2; h_D,\, w_D, \nu = \tfrac{3}{2}\big)\,, \quad (9.2.2)$$

Our covariance over time consisted of three terms. The first was periodic, whose period w_P was learned as 1 day, describing nightly dips in temperature. The second term describes longer-term trends in temperature. Its time scale w_T was learned as 3 days, with a relative magnitude h_T just over twice that of the periodic term, h_P. From this we can induce that the periodic term represents only a relatively minor component of this model. Finally, the third term represents higher frequency disturbances, with a time scale learned as 4 hours. Note that this final term is not correlated amongst sensors as were the other two: these disturbances appear to be local to each sensor. The individual output scales for each sensor given by W were learned as 2°C, 6°C and 2.6°C respectively. These give the scales of the periodic fluctuations and longer-term drifts in the air temperature at each sensor. For comparison, the scale of the disturbances h_D were learned as 1°C. As for the tide heights example, only the sensor correlations and delays were sampled over for the depicted predictions. The RMSEs for this dataset are listed in Table 9.4.

9.2.4 Running time

As described earlier, a key requirement of our algorithm is computational efficiency, in order that it can be used to represent multiple correlated sensors, and hence, used for real-time information processing. Here we consider the computation times involved in producing the results presented in the previous sections [Osborne et al., 2008, 2010b]. To this end, table 9.5 tabulates the computation times required in order to update the algorithm as a new observation is received. This computation time includes the cost of updating the weights of equation (7.3.12) and the Cholesky factor of $\mathbf{V}$ (as described in Section 5.2). Once these calculations have been performed, making predictions at any point in time is extremely fast (it is simply a matter of adding another element to $\boldsymbol{y}_\star$).

We denote the the number of stored data points as N and the number of hyperparameter samples as h. Note that we expect the cost of computation to grow as $O(N^2)$ in the number of stored data points. Note, of course, that our proposed algorithm can automatically determine the quantity of data to store in order to achieve the desired level of accuracy, as per Section 5.3. In the problems we have studied, a few hundred points to a couple of thousand were normally sufficient, although of course this will depend critically on the nature of the variables under consideration. Note also that the cost of computing the $\mathbf{K}^{(q_\star)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1}\mathfrak{Y}_{s,s}^{(q_\star,r)}\mathbf{K}^{(r)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1}$ term will grow in the cube of the number of samples in each hyperparameter. However, as previously mentioned, we consider only a fixed set of samples in each hyperparameter, and hence this term need only be computed once, off-line. In this case, our on-line costs are limited by the multiplication of that term by the likelihoods $\boldsymbol{r}_s$ to give the weights of equation (7.3.12), and this only grows as $O(h^2)$. Furthermore, note that this cost is independent of how the h samples are distributed amongst the hyperparameters. The cost of actually computing the likelihoods and predictions associated with each hyperparameter sample clearly scales linearly in h. Note also that this operation lends itself to parallelisation – the likelihood and prediction for each

Table 9.5: Required computation time (seconds) per update, over N the number of stored data points and h the number of hyperparameter samples. Experiments performed using MATLAB on a 3.00GHz processor with 2GB of RAM.

	N		
h	10	100	500
1	< 0.01	< 0.01	0.04
10	0.02	0.02	0.20
100	0.14	0.22	2.28
1000	1.42	2.22	29.73

sample can be computed completely independently of every other sample.

The results in Table 9.5 indicate that real-time information processing is clearly feasible for the problem sizes that we have considered. In general, limiting the number of hyperparameter samples is of critical importance to achieving practical computation. As such, we should exploit any and all prior information that we possess about the system to limit the volume of hyperparameter space that our GP is required to explore online. For example, an informative prior expressing that the tidal period is likely to be around half a day will greatly reduce the number of samples required for this hyperparameter. Similarly, an offline analysis of any available training data will return sharply peaked posteriors over our hyperparameters that will further restrict the required volume to be searched over on-line. For example, we represent the tidal period hyperparameter with only a single sample on-line, so certain does training data make us of its value. Finally, a simpler and less flexible covariance model, with fewer hyperparameters, could be chosen if computational limitations become particularly severe. Note again that the use of the completely general spherical parameterisation requires a correlation hyperparameter for each pair of variables, an approach which is clearly only feasible for moderate numbers of variables. While a more complicated model will return better predictions, a simple one or two hyperparameter covariance may supply accuracy sufficient for our needs.

9.3 Prediction in the Presence of Changepoints

We now present results [Garnett et al., 2009, 2010a] emerging from the use of our changepoint covariances, from Sections 4.4 and 4.5. In comparison to other approaches to changepoint problems, the key feature of our approach is the treatment of the location and characteristics of changepoints as covariance hyperparameters. As such, for the purposes of prediction, we marginalise them using our GBQ approach (Section 8.3), effectively averaging over models corresponding to a range of changepoints compatible with the data. If desired, the inferred nature of those changepoints can also be directly monitored via our Bayesian quadrature estimates for the posterior distribution for hyperparameters (Section 7.4). We will demonstrate the efficacy of our approach in the context of both retrospective regression and online prediction examples.

9.3.1 Toy example

As an expository example, we consider a function that undergoes a sudden change in both input scale and output scale. The function $y(x)$ is displayed in Figure 9.6; it undergoes a sudden change in input scale (becoming smaller) and output scale (becoming larger) at the point $x = 0.5$. We consider the problem of performing one-step lookahead prediction on $y(x)$ using GP models with a moving window of size 25.

The uppermost plot in Figure 9.6 shows the performance of a standard GP prediction model with the squared exponential covariance (3.2.3), using hyperparameters learned from data before the changepoint. The standard GP prediction model has clear problems coping with the changepoint; after the changepoint it makes predictions that are very certain (that is, have small predictive variance) that are nonetheless very inaccurate.

The central plot shows the performance of a GP prediction model using the changepoint covariance function $K^{(\mathrm{G})}$ (4.4.8). The predictions were calculated via

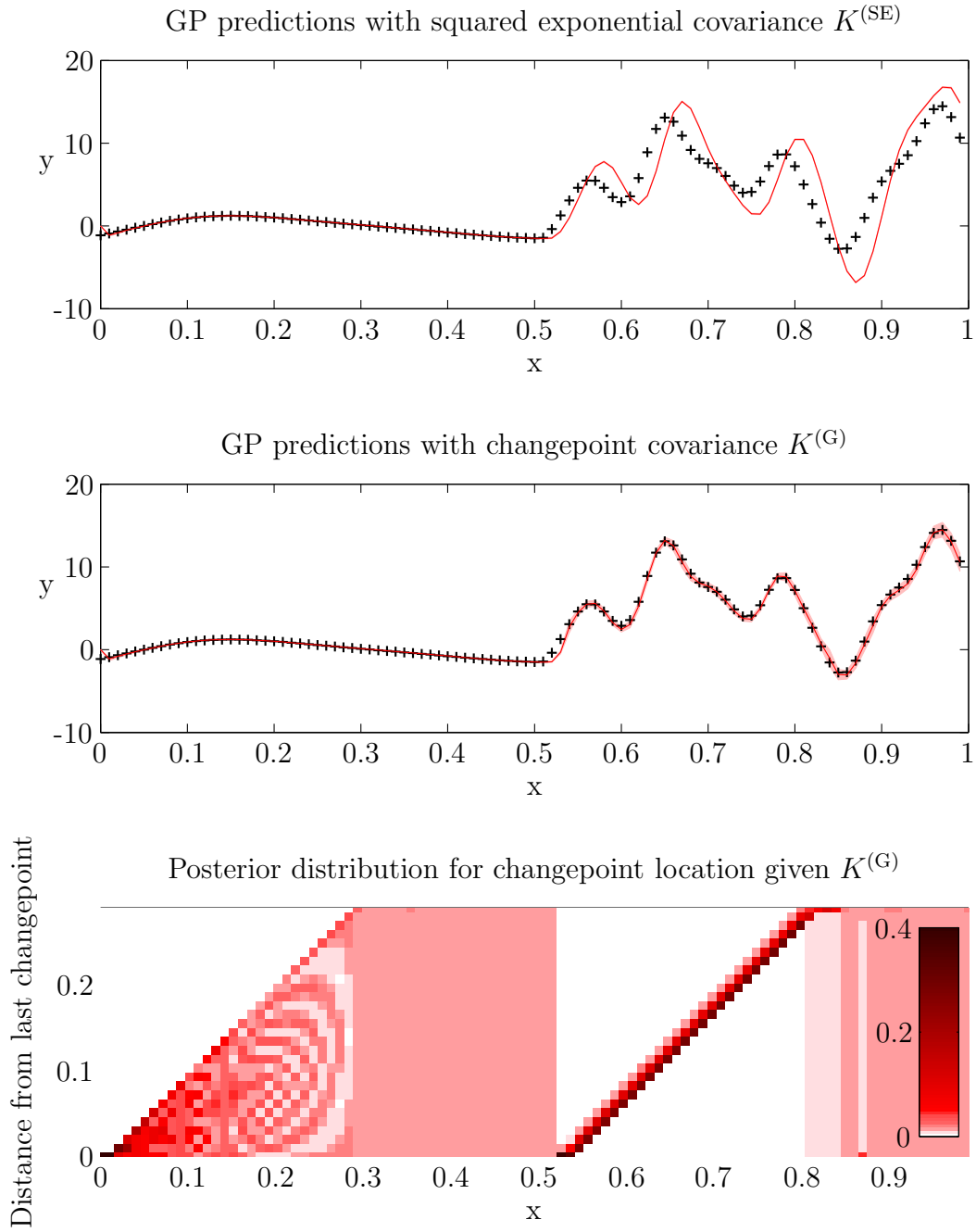


Figure 9.6: Prediction over a function that undergoes a change in both input scale and output scale using covariance $K^{(G)}$.

BQ hyperparameter marginalisation using GBQ; three samples each were chosen for the hyperparameters $\{h_a, h_b, w_a, w_b\}$, and 25 samples were chosen for the location of the changepoint. Our model easily copes with the changed parameters of the process, continuing to make accurate predictions immediately after the changepoint. Furthermore, by marginalising the various hyperparameters associated with our model, the uncertainty associated with our predictions is conveyed honestly. The standard deviation becomes roughly an order of magnitude larger after the changepoint due to the similar increase in the output scale.

The lowest plot shows the posterior distribution of the distance to the last changepoint corresponding to the predictions made by the changepoint GP predictor. Each vertical "slice" of the figure at a particular point shows the posterior probability distribution of the distance to the most recent changepoint at that point. The changepoint at $x = 0.5$ is clearly seen in the posterior distribution.

9.3.2 Nile data

We next consider a canonical changepoint dataset, the minimum water levels of the Nile river during the period AD 622–1284 [Whitcher et al., 2002]. Several authors have found evidence supporting a change in input scale for this data around the year AD 722 [Ray and Tsay, 2002]. The conjectured reason for this changepoint is the construction in AD 715 of a new device (a "nilometer") on the island of Roda, which affected the nature and accuracy of the measurements.

We performed one-step lookahead prediction on this dataset using the input scale changepoint covariance $K^{(\mathrm{D})}$ (4.4.4), and a moving window of size 150. Eleven samples each were used for the hyperparameters w_a and w_b, the input scales before and after a putative changepoint, and 150 samples were used for the location of the changepoint x_c.

The results can be seen in Figure 9.7. The upper plot shows our predictions for the dataset, including the mean and ± 1 standard deviation error bars. The lower plot shows the posterior distribution of the number of years since the last changepoint. A

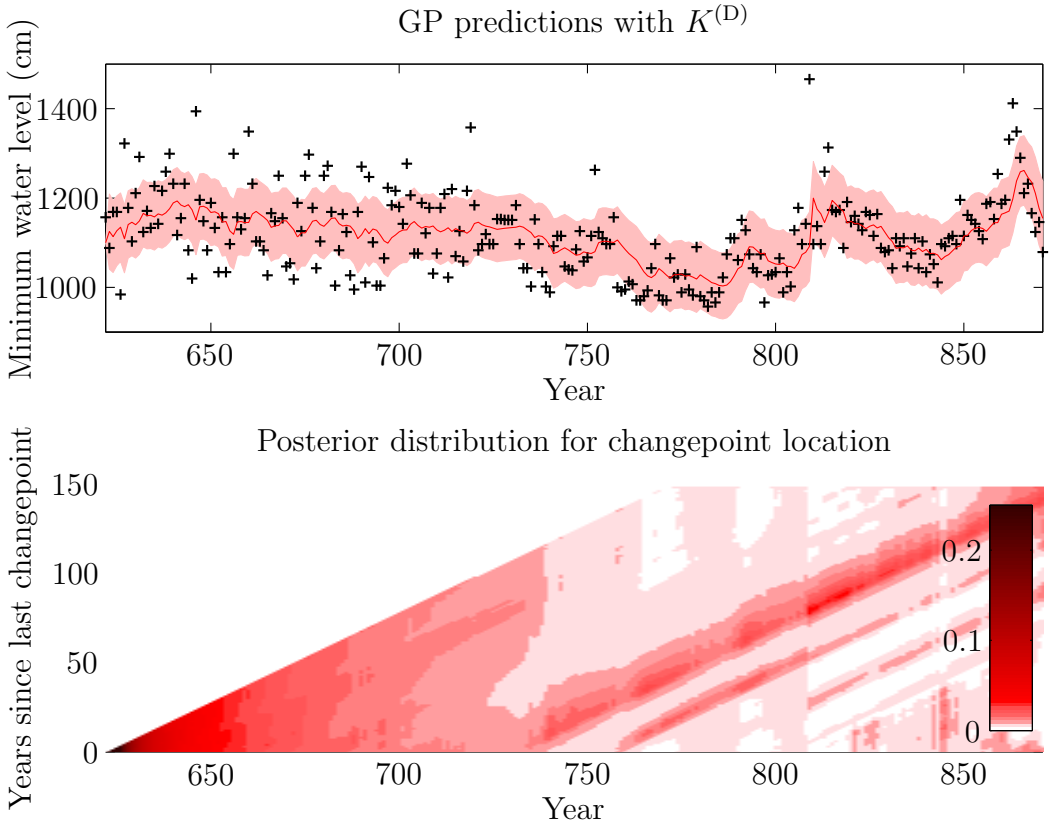


Figure 9.7: Prediction for the Nile dataset using input scale changepoint covariance $K^{(\mathrm{D})}$, and the corresponding posterior distribution for time since changepoint.

changepoint around AD 720–722 is clearly visible and agrees with previous results.

9.3.3 Well-log data

Also commonly considered in the context of changepoint detection is the *well-log* dataset, comprising 4050 measurements of nuclear magnetic response made during the drilling of a well [Ruanaidh et al., 1994]. The changes here correspond to the transitions between different strata of rock.

We performed prediction on this dataset using a simple diagonal covariance that assumed all measurements were independent and identically distributed (IID). The noise variance for this covariance (alternatively put, its output scale) was learned off-line; it was assumed known *a priori*. We then took a mean function that was constant for each rock stratum; that is, the mean undergoes changes at changepoints (and only at changepoints). Given the length of the dataset, and that regions of data before and after a changepoint are independent, we performed predictions for a point by considering a window of data centered on that point. Essentially, we performed sequential prediction for predictants mid-way through the window of past data available at any point. In each window (comprising 50 observations), we allowed for a single changepoint. Hence our model was required to marginalise over three hyperparameters, the mean before the changepoint, the mean after the changepoint and the location of that changepoint. For these hyperparameters, we took 13, 13 and 40 samples, respectively.

We compared our results against those produced by a variational Bayesian hidden Markov model with a mixture of Gaussians emission probability [Ji et al., 2006, Lee, 2009], which we refer to as the VBHMM. The resulting predictions for both methods are depicted in Figure 9.8 and RMSEs (along with log-evidences) are listed in Table 9.6. Our GP's performance was significantly better than the VBHMM alternative, largely due to the predictions made in the regions just prior to $x = 1600$ and just after $x = 2400$.

Table 9.6: RMSEs and log-evidences for Well-log dataset.

Algorithm	RMSE	$\log p(\boldsymbol{z}_d \mid I)$
VBHMM	3.52×10^3	-1.51×10^5
Changepoint GP	2.51×10^3	-1.02×10^4

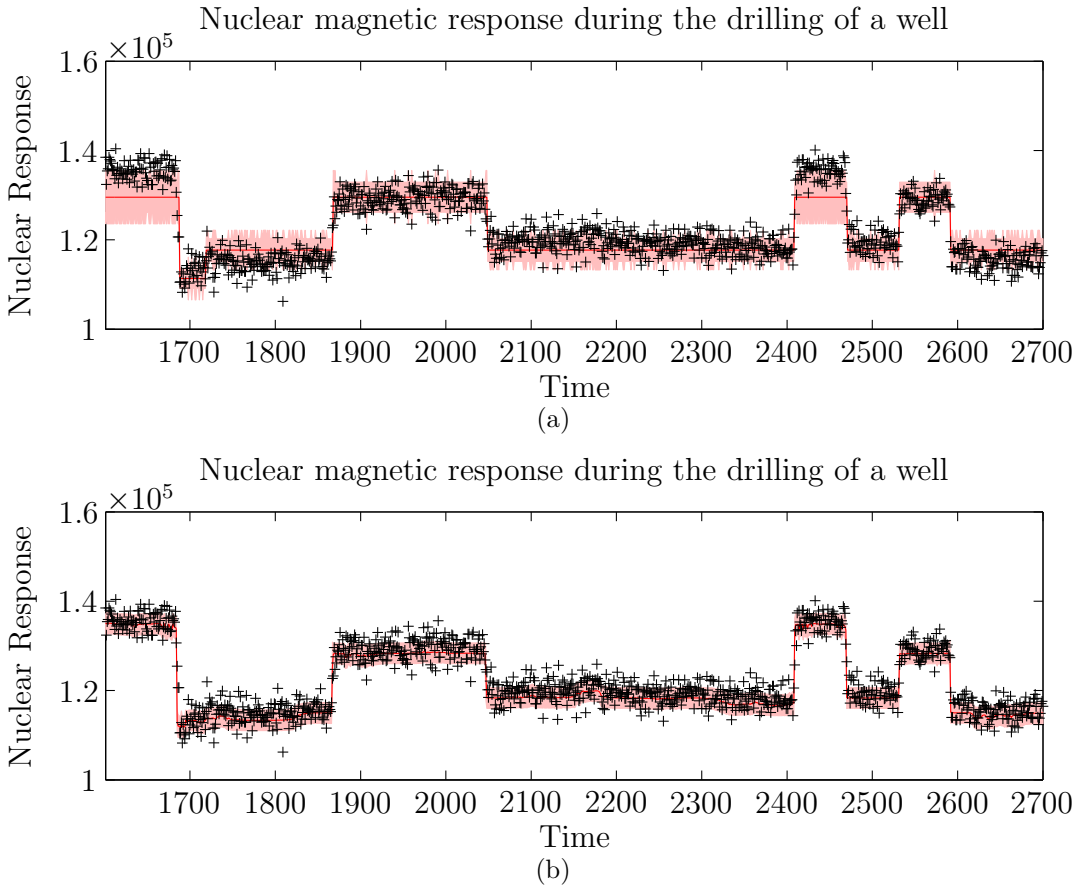


Figure 9.8: Retrospective predictions for the well-log data using (a) hidden Markov model and (b) a GP with drastic change covariance function $K^{(\mathrm{A})}$.

9.3.4 1972-1975 Dow-Jones industrial average

A final canonical changepoint dataset is the series of daily returns of the Dow-Jones industrial average between the 3rd of July, 1972 and the 30th of June, 1975 [Adams and MacKay, 2007]. This period included a number of newsworthy events that had significant macroeconomic influence, as reflected in the Dow-Jones returns.

We performed sequential prediction on this data using a GP with a diagonal covariance that assumed all measurements were IID. However, the variance of these observations was assumed to undergo changes, and as such we used a covariance $K^{(\mathrm{D})}$ that incorporated such changes in output scale. The window used was 350 observations long, and was assumed to contain no more than a single changepoint. As such, we had three hyperparameters to marginalise: the variance before the changepoint, the variance after the changepoint and, finally, the location of that changepoint. For these hyperparameters, we took 50, 17 and 17 samples, respectively.

Our results are plotted in Figure 9.9. Our model clearly identifies the important changepoints that likely correspond to the commencement of the OPEC embargo on the 19th of October, 1973, and the resignation of Richard Nixon as President of the U.S.A. on the 9th of August, 1974. A weaker changepoint is identified early in 1973, which [Adams and MacKay, 2007] speculate is due to the beginning of the Watergate scandal.

9.3.5 EEG data with epileptic event

We now consider electroencephalography (EEG) data from an epileptic subject [Roberts, 2000]. Prediction here is performed with the aim of ultimately building models for EEG activity strong enough to forecast seizure events [Faul et al., 2007]. The particular dataset plotted in Figure 9.10 depicts a single EEG channel recorded at 64Hz with 12-bit resolution. It depicts a single epileptic event of the classic "spike and wave" type.

We used the covariance $K^{(\mathrm{B})}$ (4.4.2) to model our data, accommodating the

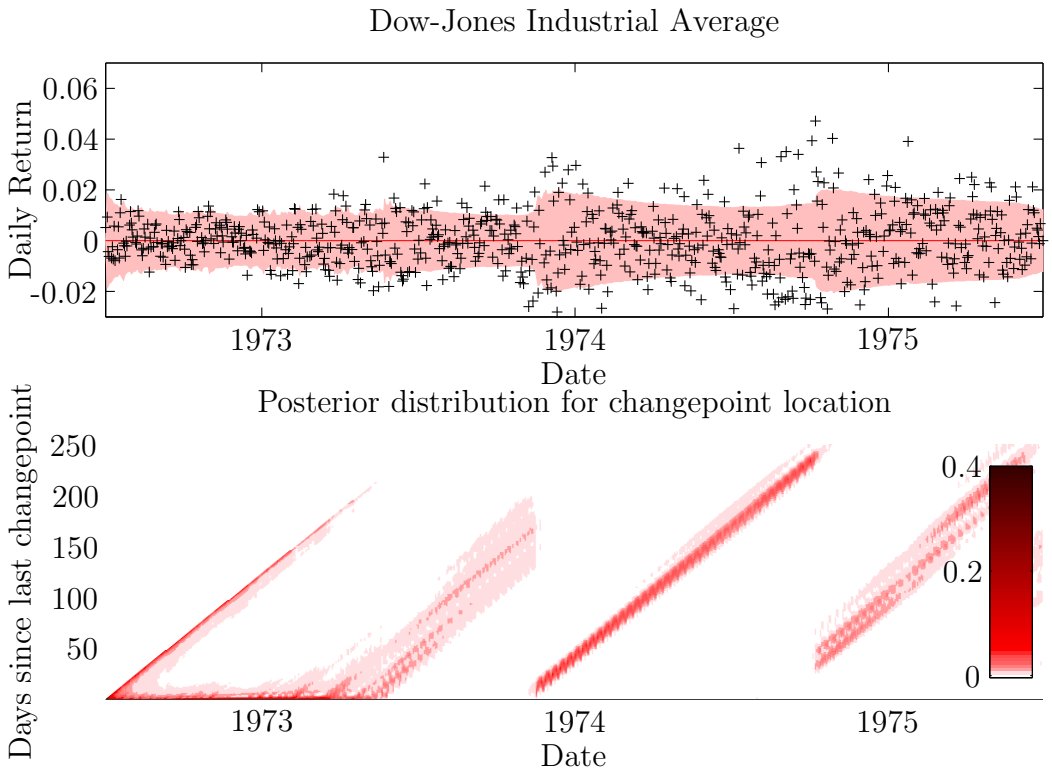


Figure 9.9: Online predictions and posterior for the location of changepoint for the Dow-Jones industrial average data using covariance $K^{(\mathrm{D})}$.

smooth transition of the data between drastically different regimes. We took $K^{(\mathrm{a})}$ as a simple squared exponential (3.2.3) and $K^{(\mathrm{b})}$ as a periodic covariance (4.3.3) multiplied by another squared exponential. $K^{(\mathrm{b})}$ is intended to model EEG data during the course of seizure, $K^{(\mathrm{a})}$, data from other regions. We assume that we have sufficient exemplars of EEG data unaffected by seizure to learn the hyperparameters for $K^{(\mathrm{a})}$ offline. We further assumed that the input scale of the non-periodic squared exponential within $K^{(\mathrm{b})}$ was identical to that for $K^{(\mathrm{a})}$, representing a constant long-term smoothness for both seizure and non-seizure periods. The hyperparameters we were required to marginalise, then, were the period T, amplitude h and roughness w of (4.3.3) for $K^{(\mathrm{b})}$, along with the location of the changepoint and its type (either periodic to non-periodic, or non-periodic to periodic). For these hyperparameters, we took respectively 7, 7, 5, 50 and 2 samples.

This model was used to perform effective retrospective prediction over the dataset, as depicted in Figure 9.10. As can be seen, our posterior distribution for the location of the changepoint correctly locates the onset of seizure.

9.3.6 Stuck sensor

To illustrate our approach to sensor fault detection, we return to the Sotonmet station from Section 9.2. In particular, we performed on-line prediction over tide height data in which readings from the sensor became stuck at an incorrect value. As such, we used the change in observation model taken from Section 4.5.2. The covariance for the underlying plant process was taken to be the sum of a periodic and a non-periodic component, as per (9.2.1), the hyperparameters for which can be learned off-line. As such, we need to marginalise only the hyperparameter corresponding to the location of a changepoint in the window, and over the type of that change point (ie. either not-stuck to stuck, or stuck to not-stuck). Clearly, our belief about the stuck value can be heuristically determined for any appropriate region – it is a delta distribution at the constant observed value. We employed a window size of 350 data points, and, correspondingly, 350 samples for the location of the changepoint. Results are plotted

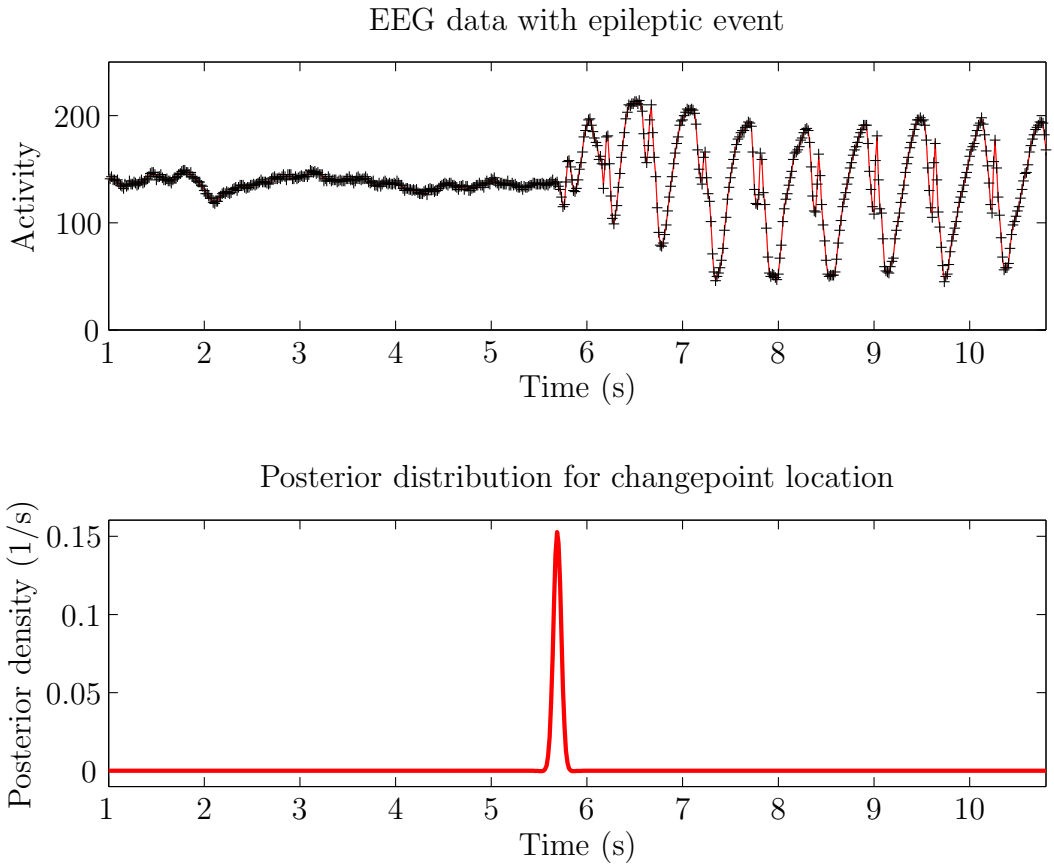


Figure 9.10: Retrospective predictions and posterior for the location of changepoint for the EEG data with epileptic event. The covariance $K^{(\mathrm{B})}$ was employed within our GP framework.

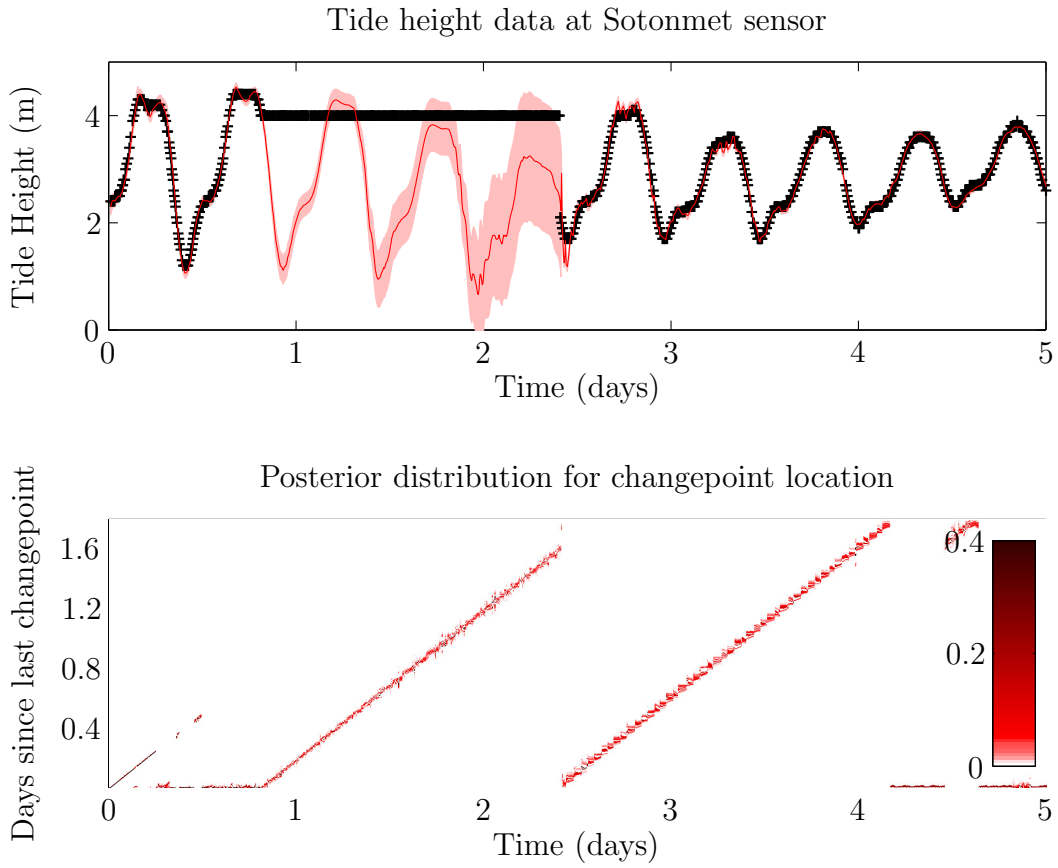


Figure 9.11: Online predictions and posterior for the location of changepoint for the tide height data. The fault was modelled as a change in observation likelihood of the form described in Section 4.5.2.

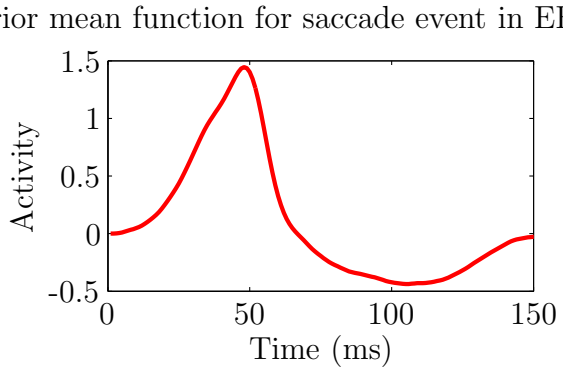


Figure 9.12: Eye movement activity during a saccade event.

in Figure 9.11. Our model correctly identified the beginning and end of the fault. By then performing fault removal via (4.5.2), the model is able to perform effective prediction for the plant (tide) process throughout the faulty region.

9.3.7 EEG data with saccade event

EEG is a useful, non-invasive measure of brain activity. However, the EEG signal is often corrupted by eye movement artifacts referred to as saccades; it is necessary to remove the artifact from the EEG signal. Figure 9.12 shows typical activity during a saccade. We treat such saccade events as faults, in this context, we use our methods to perform prediction and effect fault removal [Reece et al., 2009]. This problem was treated as a blind source separation problem in [Roberts et al., 1999] and an independent component analysis solution was proposed which identified the separate artifact free EEG signal.

We employ the model proposed in Section 4.5.3. We take a squared exponential covariance for the underlying EEG plant process, and another squared exponential covariance for the saccade drift fault contribution. We consider two different prior mean functions $\boldsymbol{c}(\boldsymbol{x};\phi)$ for use for the fault process. The first assumes that no further information about the shape of the EOG signal is known and, as such, the fault prior mean $\boldsymbol{c}(\boldsymbol{x};\phi)$ is zero throughout. For the second variation of the drift model, a prior mean $\boldsymbol{c}(\boldsymbol{x};\phi)$ is learnt from samples of EEG activity during saccades, taking the shape depicted in Figure 9.12. In this case, the drift covariance function models the residual

between the prior drift mean and the current signal.

The presence of abundant uncorrupted EEG signal data allowed the length and height scale hyperparameters for the model to be learned offline. We assumed the input scale for the saccade was the same as for the EEG signal itself. As such, we were required to marginalise three hyperparameters: the output scale h of the drift covariance; and the artifact start time and duration. For the zero mean fault model we took 13, 13 and 150 samples, respectively, for those hyperparameters. For the non-zero mean model we took 5, 7 and 75 samples, respectively. The non-zero mean model also requires a vertical scaling factor for the prior mean shape (Figure 9.12) and, for this hyperparameter, we took 9 samples.

For the artifact start time hyperparameter, we took a uniform prior over the extent of the dataset. As usual, if no artifact was detected, the posterior mass for the start time would be concentrated at the end of the dataset. We cannot be very certain *a priori* as to the duration of a saccade, which will be dependent upon many factors (notably, the size of the saccade). However, a reasonable prior [Jürgens et al., 1981] might place upon the logarithm of the saccade duration a Gaussian with a mean of log(110ms) and a standard deviation of 0.6 (meaning that saccade durations of 60ms and 200ms are both a single SD from the mean). This was the prior taken for the artifact duration hyperparameter.

Figures 9.13 and 9.14 show the result of performing retrospective prediction over our EEG data. Figure 9.13 shows the 1 standard error confidence interval for the artifact free EEG signal and the saccade artifact obtained using our algorithm with a zero prior mean for the fault model. The figure also shows the retrospective posterior distribution over the artifact start time. Although our approach is able to determine when an artifact occurs, its start-time is hard to determine as, at the artifact onset, the observed signal is very similar to the pure plant signal. However, the approach successfully removes the saccade artifact from the EEG signal. We can also use the methods of Section 7.4 to produce the full posterior for the EEG signal over the saccade event, as plotted in Figure 9.15a. Note that we can distinguish two

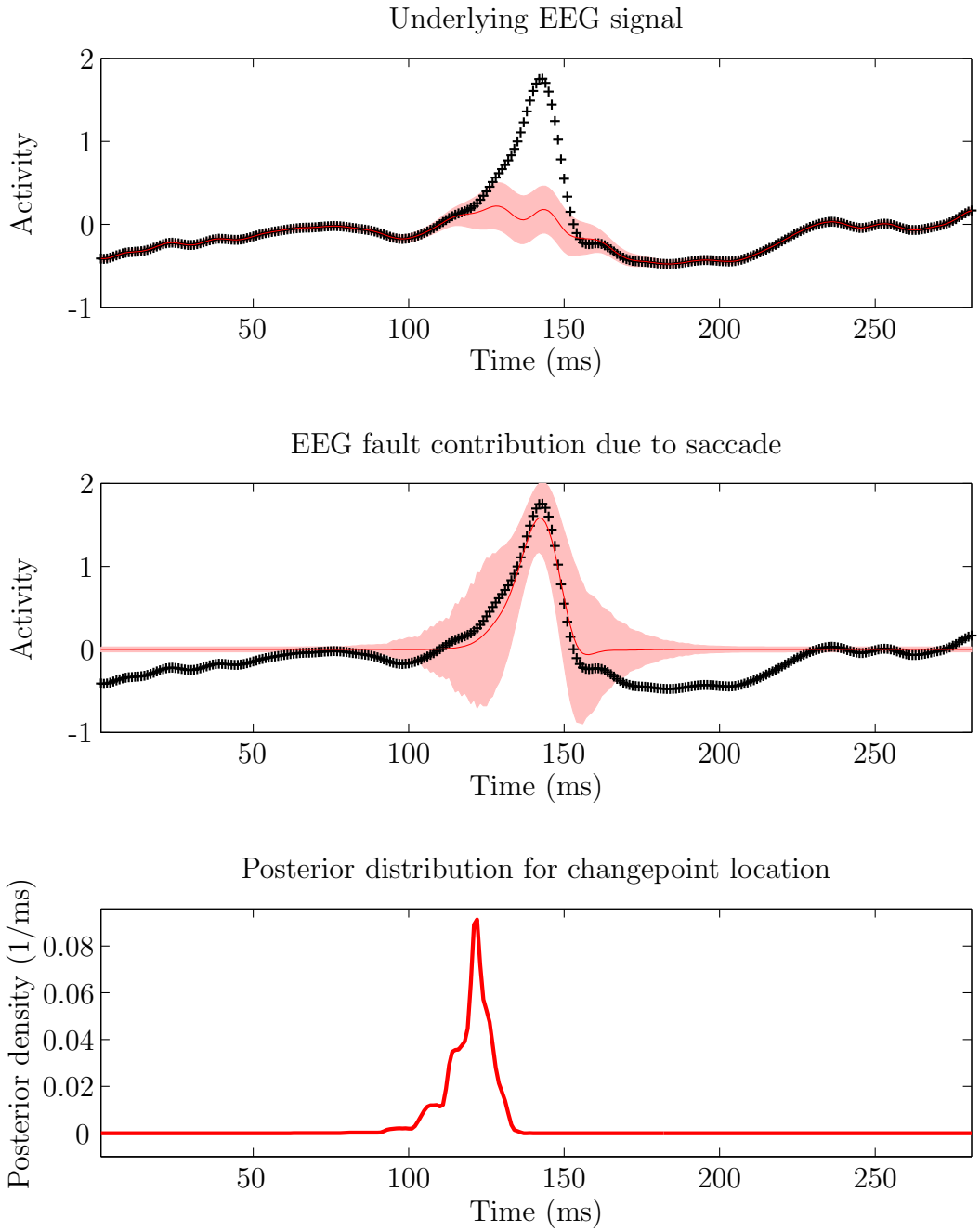


Figure 9.13: Retrospective predictions and posterior for the location of changepoint for the EEG data with saccade. Predictions are made both for the plant process (the underlying EEG signal) using (4.5.2), as well as for the fault contribution due to saccade, using (4.5.3). The GP assumes a zero prior mean during the saccade.

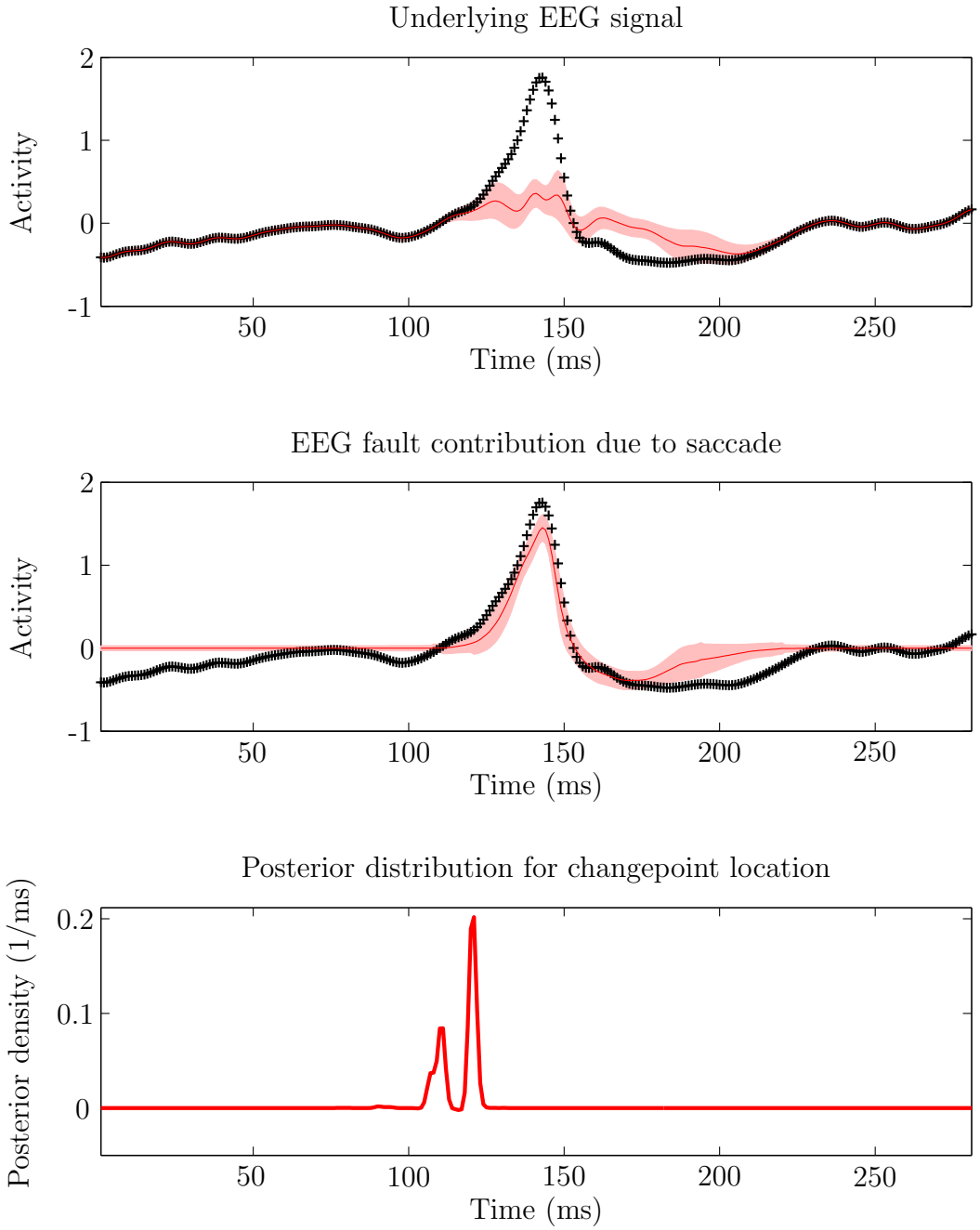


Figure 9.14: Retrospective predictions and posterior for the location of changepoint for the EEG data with saccade. Predictions are made both for the plant process (the underlying EEG signal) using (4.5.2), as well as for the fault contribution due to saccade, using (4.5.3). The GP assumes a prior mean during the saccade of the common form (Figure 9.12) for activity during such an event.

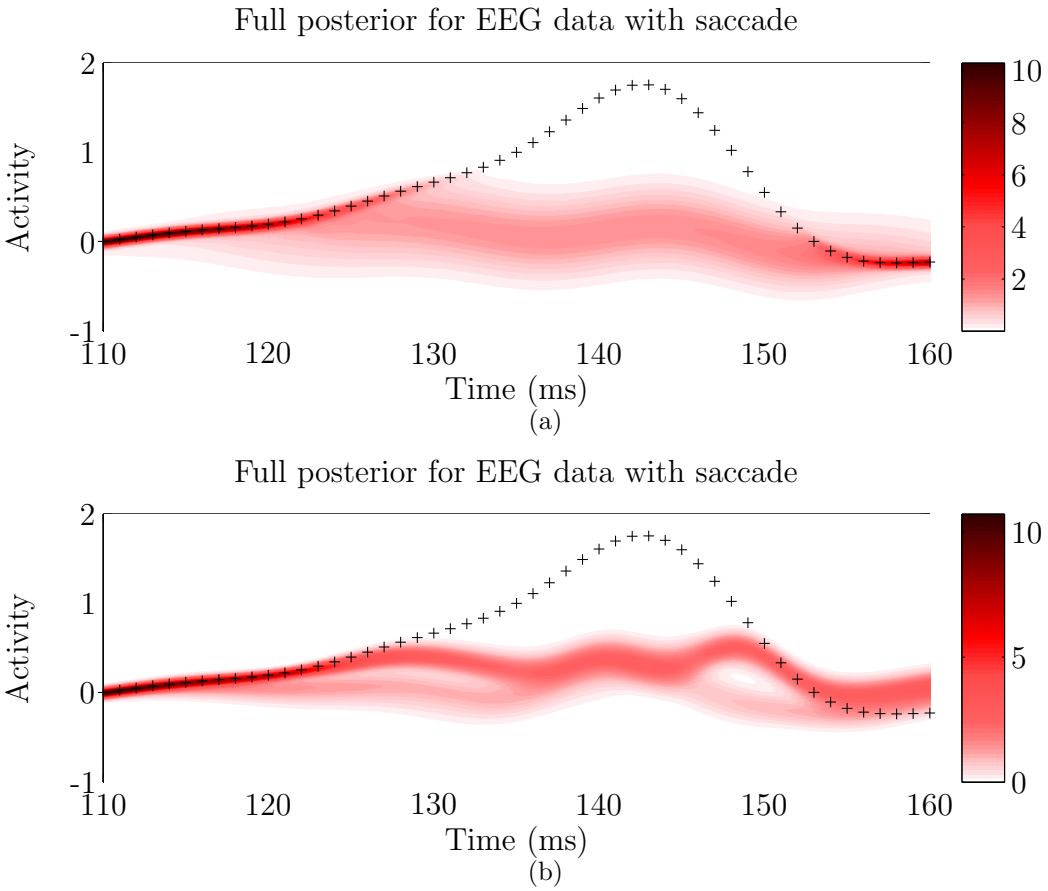


Figure 9.15: Full retrospective posteriors for the EEG data with saccade plant process, using (4.5.2). A GP was taken that assumes (a) a zero prior mean and (b) a prior mean during the saccade of the common form (Figure 9.12) for activity during such an event.

models: the model that simply follows the EEG signal; and the model that assumes that a saccade artifact may have occurred. The former is characterised by a tight distribution around the observations, the latter is much more uncertain due to its assumption of a fault. Note that the first model gradually loses probability mass to the second until the first becomes completely implausible.

Figure 9.14 shows the recovered signals obtained using the non-zero mean saccade artifact model. In this case our approach more accurately identifies the start and finish times of the artifact and also accurately separates the plant and faulty signals. It is interesting to note that this approach results in a bimodal distribution over the artifact start-time. The most likely start times are identified by our algorithm to occur at the kinks in the data at t=115 ms and t=120 ms. This results in a bimodal estimate of the true and faulty signals. These bimodal signal estimates and the distribution over the artifact start times are shown in Figure 9.15b.

9.4 Active Data Selection

We now apply our methods for active data selection to several datasets [Osborne et al., 2010a]. Specifically, we perform GP prediction for a variable, and use those predictions in order to perform active data selection as per Section 5.4. Although we do not have active control over the sensors referred to below, all datasets were sufficiently dense that we can select from the pre-recorded observations without being significantly constrained by the available times of observation. The full list of all available observations additionally serves as our 'real function' (in the sense of the legend of Figure 9.1) in order to verify the accuracy of our predictions, except, of course, where the observations are faulty. Finally, we produce a posterior over changepoint/fault locations as per Section 7.4, as required.

With readings taken by each sensor every minute, we simply round the observation times requested by the GP down to the nearest minute.

Table 9.7: Active data selection over a tide-heights dataset.

	Observations taken from station			
GP type	Bramblemet	Sotonmet	Chimet	Cambermet
Independent	74	75	74	74
Multi-output	49	52	82	9

9.4.1 Tide Heights

We firstly perform ADS over the four tide sensors discussed in Section 9.2.2. Results from the active selection of observations from these sensors are displayed in Figure 9.16. We tested, as before, both a model which took independent GPs for the readings from each sensor, and a model which treated the tide readings using a multi-output GP approach. The cost of observation, C, was taken as 0.2m for each sensor, and a lookahead of zero was used.

Consider first the case shown in Figure 9.16(a), in which separate independent GPs are used to represent each sensor. Note that a large number of observations are taken initially as the dynamics of the sensor readings are learnt, followed by a low but constant rate of observation. In contrast, for the multi-output case shown in Figure 9.16(b), the GP is allowed to explicitly represent correlations and delays between the sensors. This data set is notable for the slight delay of the tide heights at the Cambermet station. Note that after an initial learning phase as the dynamics, correlations, and delays are inferred, the GP chooses to sample only very rarely from the delayed Cambermet station. Despite the reduced number of observations from the Cambermet station being made, the resulting predictions remain remarkably accurate. The numbers of samples drawn from each sensor are listed in Table 9.7. This demonstrates the significant reduction in sampling frequency resulting from the proper treatment of the correlations amongst our sensors.

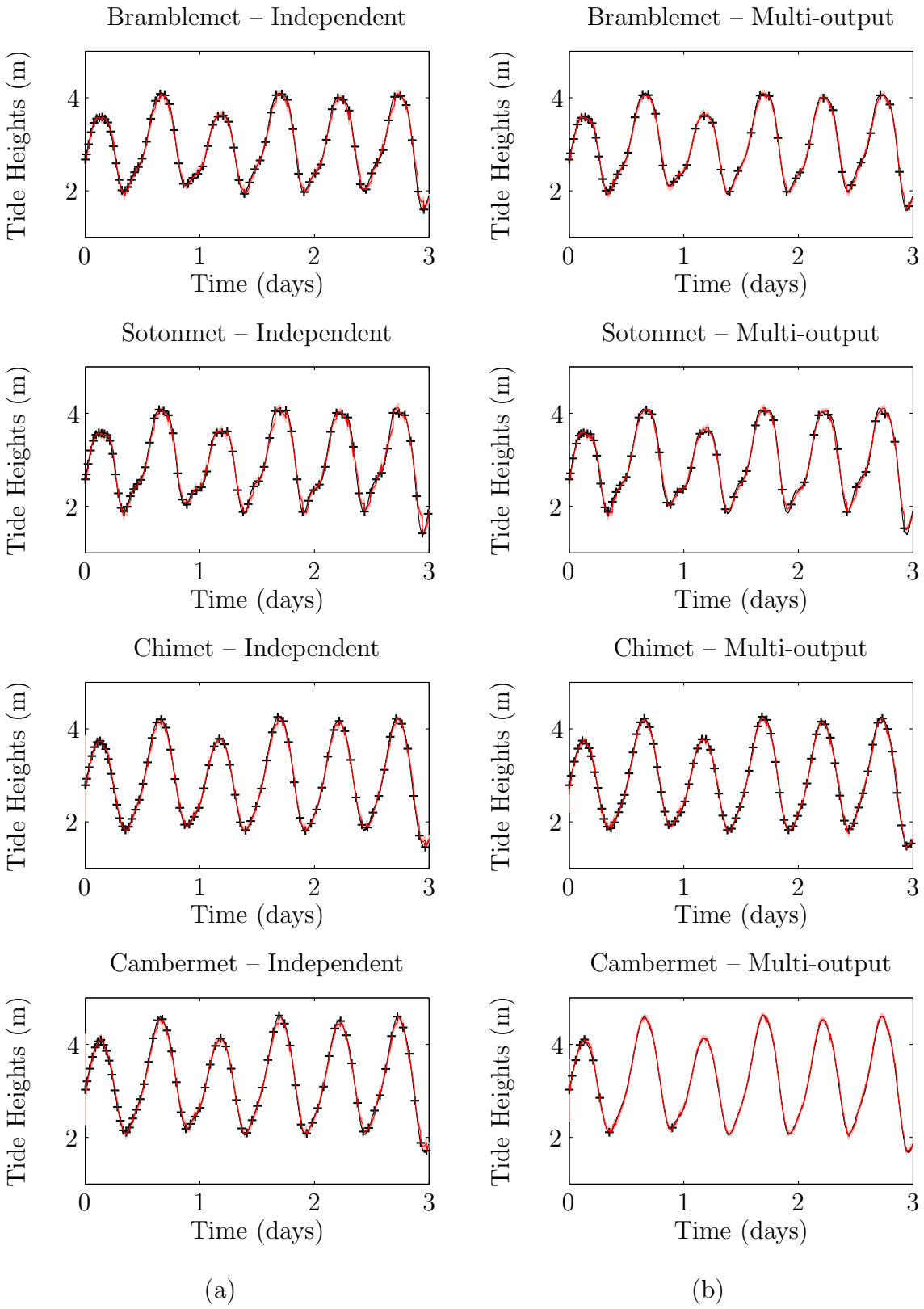


Figure 9.16: Comparison of active sampling of tide data using (a) independent and (b) multi-output Gaussian processes.

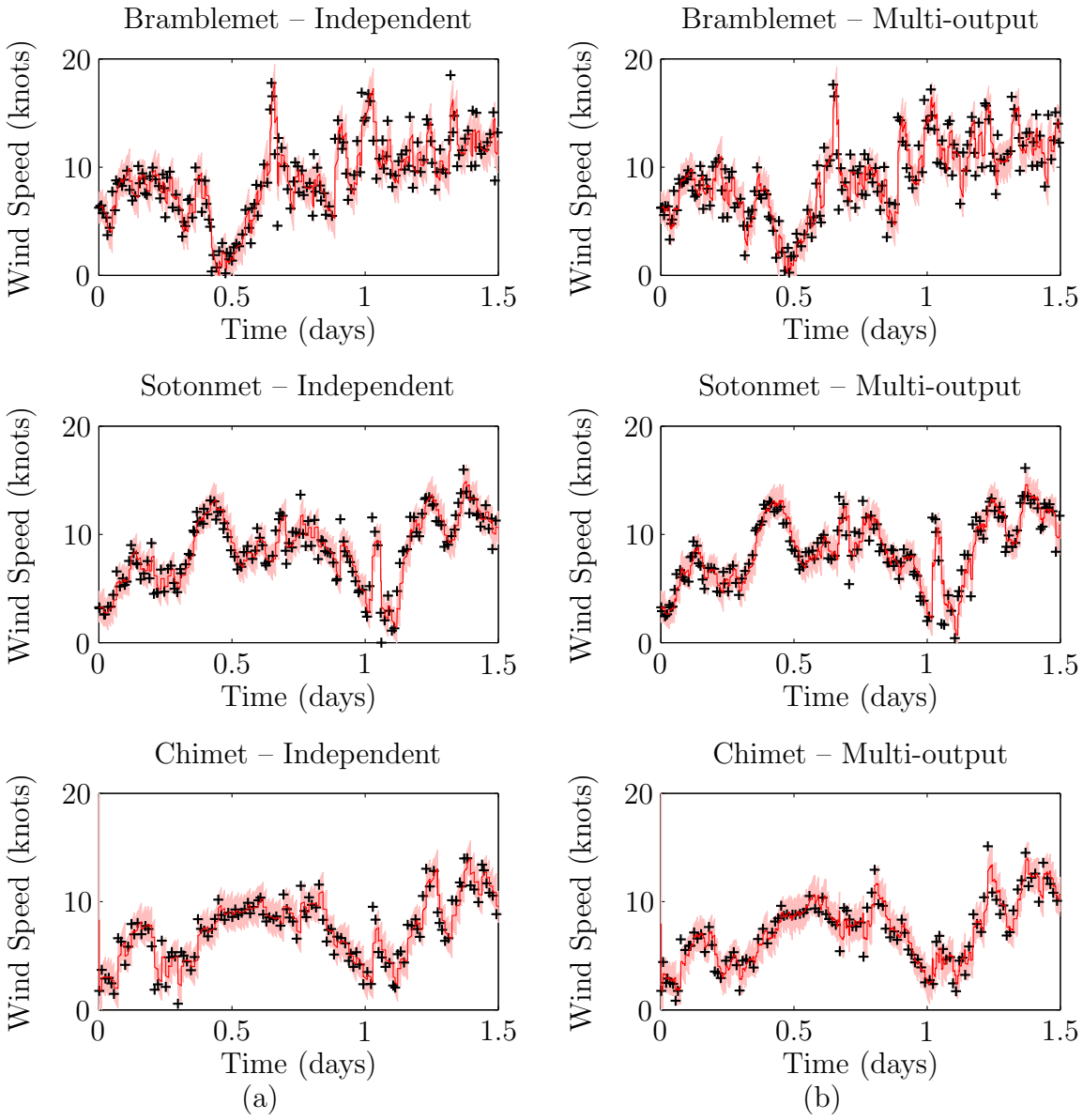


Figure 9.17: Comparison of active sampling of wind speed using (a) independent and (b) multi-output Gaussian processes.

Table 9.8: Active data selection over a wind-speed dataset.

	Observations taken from station		
GP type	Bramblemet	Sotonmet	Chimet
Independent	170	160	124
Multi-output	176	161	114

9.4.2 Wind Speeds

We now turn to a wind speed dataset, drawn from the Bramblemet network discussed in Section 9.2. By analogy to the covariance function over tides (9.2.1), we used the following covariance function

$$K\big([t_1,\, l_1],\, [t_2,\, l_2]\big) = W(l_1,\, l_2)\, K^{(\text{per-Mtn})}\big(t_1 - \Delta_{l_1}, t_2 - \Delta_{l_2}; h_P = 1,\, w_P, \nu = \tfrac{1}{2}\big) + \\ \delta(l_1 - l_2)\, K^{(\text{Mtn})}\big(t_1, t_2; h_D,\, w_D, \nu = \tfrac{3}{2}\big)\,. \quad (9.4.1)$$

An observation cost C of 3 knots was taken for each station, along with a zero lookahead. Our results are plotted in Figure 9.17.

Table 9.8 lists the numbers of observations selected from each sensor station. Due to the very weak correlations that exist between sensors for this data, there is only a very small reduction in the number of observations taken when the GP is permitted its full multi-output form. Note also that the different dynamics of the wind speed at each station mean that very different numbers of samples are drawn from each: the more volatile readings at the Bramblemet station must be sampled more often than for the more sedate wind speeds at the Chimet station. This is unlike the results over tide data (Table 9.7), where for the independent case almost identical numbers of samples are drawn from the four very similar streams of data.

9.4.3 Stuck Tide Heights

Figure 9.18 demonstrates data selection over a dataset (see Section 9.3.6) featuring a faulty period in which the observations returned by a tide height sensor become stuck at a reading of 4m. As such, we used a model that allowed for changes in the observation likelihood of this type. The covariance function used for non-faulty behaviour was that used for tide heights above, with its hyperparameters (such as tidal period and amplitude) learned off-line from the large corpus of such data. Hence we were required to marginalise only the hyperparameters that directly specify the

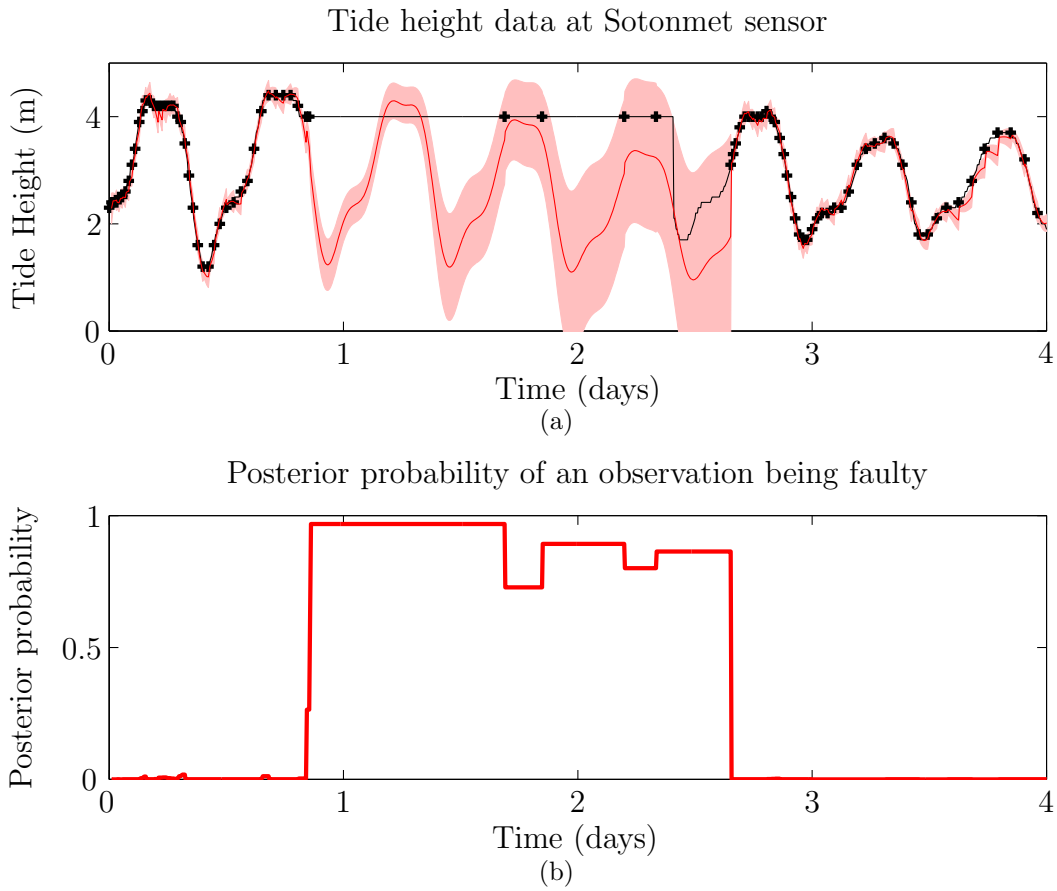


Figure 9.18: Active data selection over intermittently faulty tide height data from the Sotonmet station – (a) the selected observations and consequent predictions (with lookahead $\epsilon = 5$ mins), (b) the posterior probability that an observation taken at a time $t + \epsilon$ would be faulty, given all observations taken up to and including time t.

fault[2]. This necessitated the stipulation of a prior over the usual length of such faults. We took a prior over the natural logarithm of this length with a mean corresponding to a fault length of 12 hours and a standard deviation of 0.5. A grid of hyperparameter samples was taken with 11 samples in the length of the fault, and 100 samples in its location. The cost of an observation was taken to be 0.075m and the lookahead was 5 minutes.

Our algorithm correctly detects the stuck fault beginning at around $t = 0.8$ days, reducing its sampling frequency as a result. More observations are selected as it begins to suspect that the fault may have ended. Accurate predictions, with realistic uncertainties, are made throughout.

9.4.4 Wannengrat weather sensor network

Our second dataset was drawn from the Wannengrat Alpine Observatory being built above and around the town of Davos as part of Swiss Experiment[3]. This comprises a wireless sensor network deployed with the aim of understanding the complex metereological processes occurring on the snowpack. The changepoints that exist within this dataset, along with the limited battery life of the remote sensors, make it a natural fit for our methods. In particular, we perform active data sampling over a dataset comprising ambient temperature measurements featuring a dramatic changepoint.

We use a covariance of the form (4.2.1). We express the covariance over sensors $K^{(l)}$ as a function of the (isotropic) spatial distance between the known sensor locations. We used the Matérn covariance function (3.2.5) for this purpose. The covariance over time $K^{(t)}$ was another Matérn covariance of the form (3.2.5), modified to allow for changepoints in output scale, using $K^{(\mathrm{E})}$ from Figure 4.4c. Our grid of hypersamples had 50 samples in the location of the changepoint, 7 samples in both the output scale before and after a putative changepoint, 5 samples in the input scale and 3 samples in the noise SD σ. The cost of observation was taken to be 0.5°C.

[2]Clearly, our belief about the stuck value can be heuristically determined for any appropriate region—it is a delta distribution at the constant observed value.

[3]See `www.swiss-experiment.ch/index.php/Wannengrat:Home` for more details.

Although slightly higher than is realistic given the actual batteries used, this cost did allow us to produce visually intelligible plots.

Figure 9.19 demonstrates active data selection over this dataset. Two initial samples are taken from each sensor at near-identical times, serving to educate the algorithm about the correlations that exist among the sensors. Subsequently, these correlations are exploited by taking a more dispersed pattern of observations; it is wasteful to take observations from two sensors simultaneously given the knowledge that their readings are strongly correlated. It can be seen that there is a dramatic increase in sampling frequency coincident with the volatile fluctuations in temperature that begin at about $t = 0.7$ days. Subsequently, when the fluctuations dissipate at about $t = 0.9$ days, the rate of sampling drops again.

9.4.5 EEG data with saccade event

We now test our ADS approach upon the EEG data corrupted by a saccade fault, from Section 9.3.7. We used a prior upon the logarithm of the saccade duration that was a Gaussian with a mean of log(110ms) and a standard deviation of 0.6. Using a grid of hyperparameter samples, we took 100 samples in the location of the fault, 9 in its length and 5 for the scale of variation of the drift. The cost of observation used was 0.1.

Fig. 9.20 displays active data selection and prediction over our EEG data, demonstrating the identification of the fault. Sampling is reduced during the fault, when observations become more (but not completely) uninformative.

9.5 Global Optimisation

We have compared the results of our Gaussian Process Global Optimisation (GPGO) method from Chapter 6, with several proposed alternatives on an extensive set of standard test functions for global optimisation [Osborne et al., 2009].

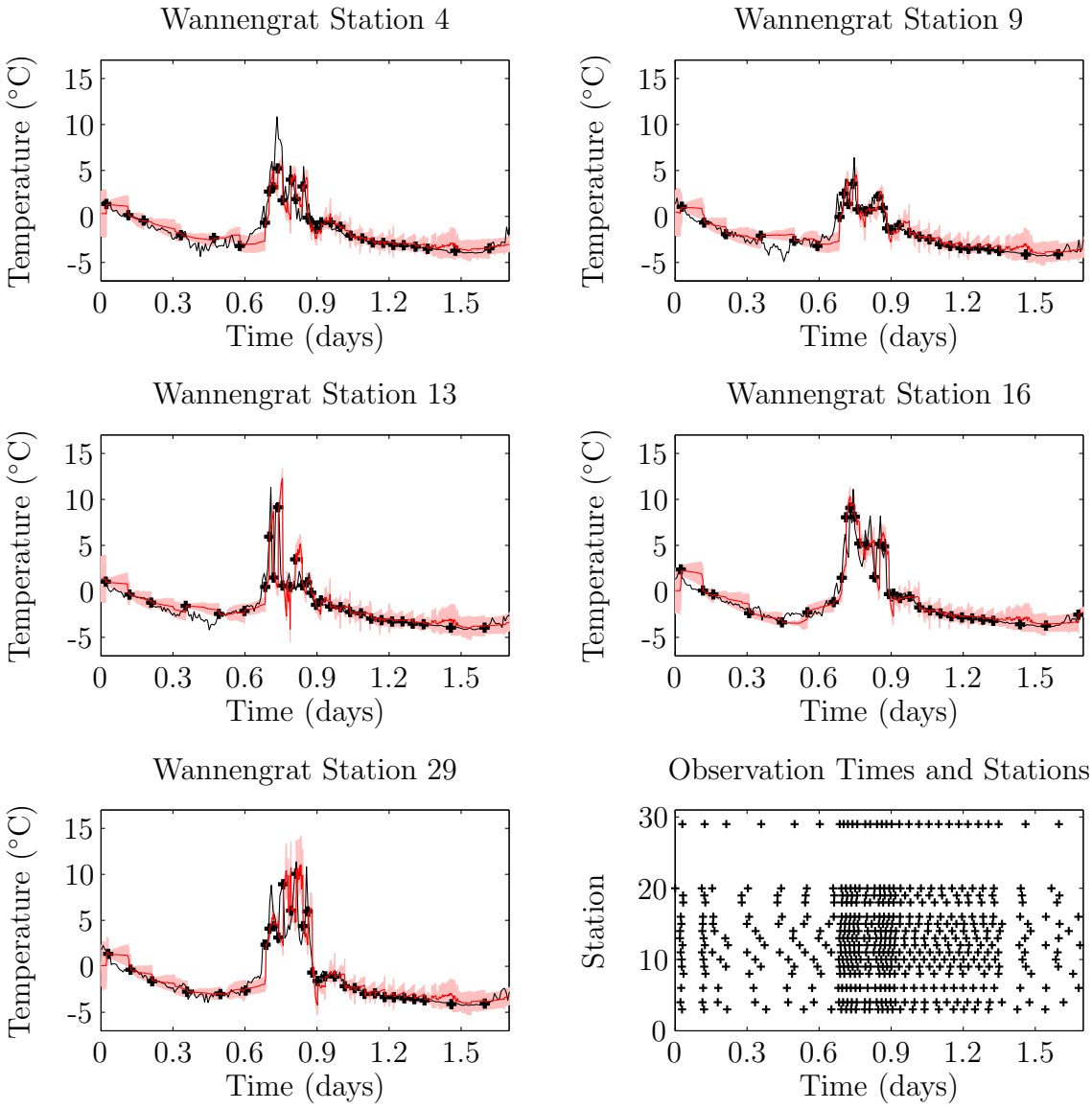


Figure 9.19: Active data selection of ambient temperatures at 16 Wannengrat sensor stations. Displayed predictions were made with zero lookahead, $\epsilon = 0$.

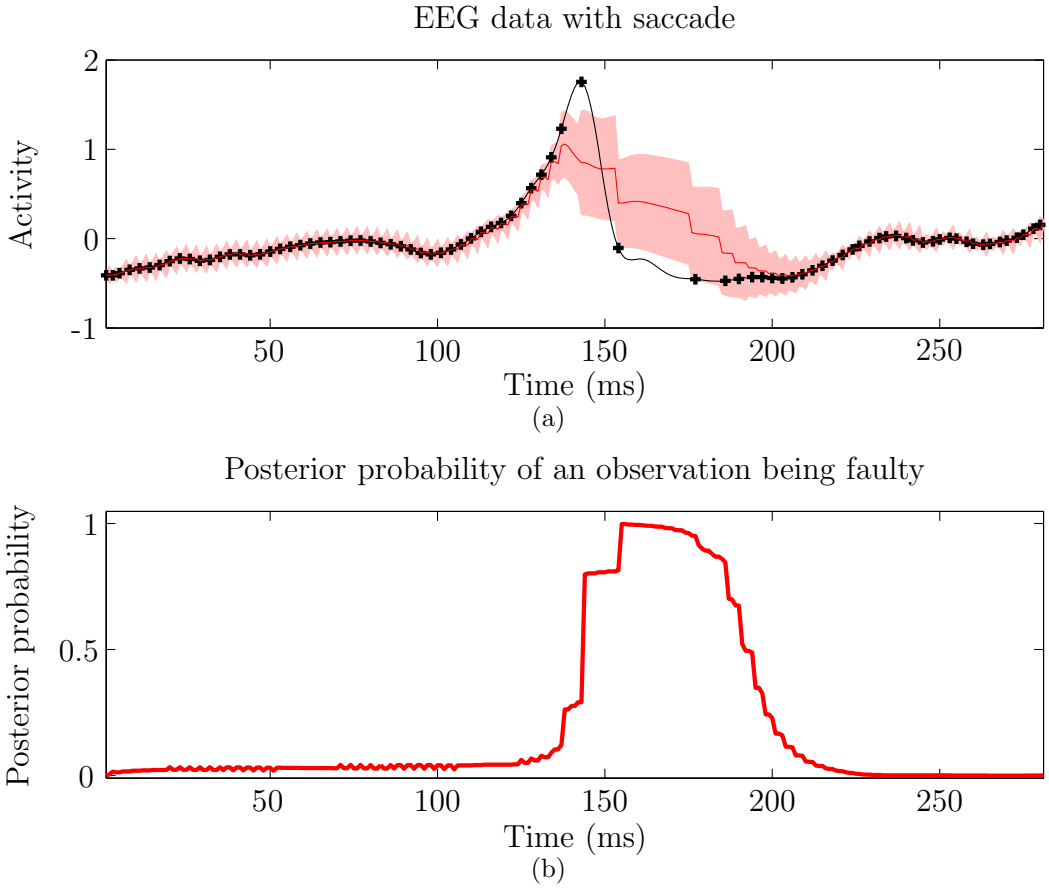


Figure 9.20: Active data selection over EEG data containing a saccade fault – (a) the selected observations and consequent tracking predictions (with zero lookahead), (b) the posterior probability that an observation taken at a time t would be faulty, given all observations taken up to and including time t.

9.5.1 Testing methodology

To compare the various optimisation methods, we chose a diverse list of 14 standard test functions (see Table 9.9) for global optimisation methods [Molga and Smutnicki, 1995]. Each standard function additionally has a commonly used "standard" box-shaped test region containing the global minimum[4]. For noiseless testing, to reduce the possibility of an algorithm 'stumbling' upon an optimal value rather than systematically finding it, for each problem we created ten associated subproblems by randomly[5] translating the test region while guaranteeing that all minima remain within the box (except for multi-step lookahead problems, for which time restricted us to testing two each). This allows us to compare the average performance of each method on each problem. These translated subproblems also increase the difficulty of the test functions in several cases, because several of the functions are ill-behaved outside their standard test domain. A similar tactic was used to reduce sample bias during noisy testing; for each test problem and noise level, three different realisations of noise were used, and the performance for each method was determined from its aggregate performance.

Additionally, we tested the noiseless optimisation routines on two- and three-dimensional functions generated from the GKLS random-function generator [Gaviano et al., 2003]. The default parameters were used, except the number of local minima was increased from ten to twenty; the first ten generated functions for each set of parameters formed the test set.

Because the objective functions we are considering are assumed to be very expensive to evaluate, we limit the number of allowed function evaluations. For noiseless problems, we use ten-times the dimension of the problem; for noisy problems, we use double this number. This restriction also applied to any initial function evaluations

[4]Note that our algorithm is not inherently limited to box-bounded problems; we can impose any arbitrary constraint by setting the expected loss to $+\infty$ over the region outside the desirable domain. The problem of constrained optimisation is then passed onto the routine minimising our expected loss.

[5]See Section 8.2 for a discussion concerning the validity of using pseudo-random generators for the purposes of algorithm testing.

required to establish hyperparameters, if prescribed by the procedure under consideration. For very expensive functions, limiting the number of function evaluations is a natural stopping criterion.

We use the 'gap' measure of performance to compare the methods [Huang et al., 2006]

$$G \triangleq \frac{y\big(x_{\text{first}}\big) - y\big(x_{\text{best}}\big)}{y\big(x_{\text{first}}\big) - y_{\text{opt}}}\,, \tag{9.5.1}$$

where y is the objective function, y_{opt} is the global optimum, and x_{first} and x_{best} are the first point evaluated and best point found at termination, respectively. For every method tested, the initial point x_{first} was chosen to be the medial point of the test region.

9.5.2 Implementation details

The implementations used for the efficient global optimisation (EGO) [Jones et al., 1998], the radial basis function (RBF) [Gutmann, 2001], and the dividing rectangles (DIRECT) [Jones et al., 1993] optimisation algorithms were those in the TOMLAB/CGO toolbox for MATLAB, version 6 [Holmström, 1999]. The default settings were used for each, except that the number of function evaluations used in the initial samples (a fit to which permits the methods to set their hyperparameters) was reduced by 25%. Otherwise, the methods would have used all of their available function evaluations just to build their model. Thereby we improve the performance of these algorithms by allowing them to intelligently – rather than systematically – choose where to place at least a portion of their function evaluations. The MATLAB implementations used for the noisy optimisation routines Implicit Filtering (IF) [Kelley, 2008], version 0.7, and Stable Noisy Optimisation by Branch and Fit (SNOBFIT) [Huyer and Neumaier, 2008], version 2.1, are freely available from the authors' web pages.

The optimisation of the expected loss surface was performed using parallel subspace trust-region, interior-reflection Newton methods. Problematically for simple optimisation procedures, the expected loss function is usually multimodal. Of course,

even a local minimum of the expected loss is likely to represent a reasonably informative observation. That observation made, the local minimum will be eroded and we will be compelled to explore elsewhere. Indeed, our results proved largely insensitive to changes in the method used for this optimisation.

For all testing, we used a constant prior mean μ. The covariance function used for modelling the objective function was the sum of two isotropic squared exponential covariance functions:

$$K(x_1, x_2) \triangleq h_A^2 \exp\left(-r_A^2/2\right) + h_B^2 \exp\left(-r_B^2/2\right),$$

where, if d indexes the dimensions of the input space, $r_A^2 \triangleq \sum_d (x_{1,d} - x_{2,d})^2 / w_A^2$ gives a non-periodic term (as per (3.2.3)) and $r_B^2 \triangleq \sum_d (\sin \pi (x_{1,d} - x_{2,d})/w_B)^2$ a periodic term (as per (4.3.3)). This is intended to allow us to model functions that are the superposition of a non-periodic and a periodic component. Fortunately, of course, there exist a wide variety of other covariance functions should the problem call for them. Note also that we need not place the GP on the function directly – clearly, a function known to be strictly positive might benefit from a GP over its logarithm. Other transformations might always be considered to improve our performance for a particular objective function. For functions known to have periodic components, the full covariance function was used; otherwise, only the non-periodic component was used by setting $h_B = 0$. We tested both the myopic one-step lookahead and the two-step lookahead versions of our algorithm, the latter only applied to the most demanding problems. For noisy optimisation, we limited ourselves to the myopic one-step lookahead policy (6.6.1) with the non-periodic covariance. Such restrictions were made to accelerate testing (allowing more functions to be included in the test).

The input scales w_A and w_B and the output scales h_A and h_B were marginalised using GBQ as described in Section 8.3. Suppose the test region is specified by the box $\prod_i [a_i, b_i]$, and let $m \triangleq \max_i (b_i - a_i)$ be the largest dimension of this box. The samples for these hyperparameters were chosen to be uniform in the log-space; specifically, the samples for $\log(w_A)$ and $\log(w_B)$ were chosen to be equally spaced in the interval

Table 9.9: Objective functions used for testing the optimization algorithms. When noted, the dimension d is equal to that indicated by the function name.

Abbreviation	Full Name	Test Region
A2/5	Ackley 2/5	$[-32.8, 32.8]^d$
Br	Branin	$[-5, 10] \times [0, 15]$
C6	Camelback 6	$[-5, 5]^2$
GK2/3	GKLS 2/3	$[-1, 1]^d$
G–P	Goldstein–Price	$[-5, 5]^2$
G2/5	Griewank 2/5	$[-600, 600]^d$
H3/6	Hartman 3/6	$[0, 1]^d$
R	Rastrigrin	$[-5.12, 5.12]^2$
Sh5/7/10	Shekel 5/7/10	$[0, 10]^4$
Shu	Shubert	$[-10, 10]^2$

$[\frac{m}{10} - \frac{3}{2}, \frac{m}{10} + \frac{3}{2}]$, and the samples for $\log h_A$ and $\log h_B$ were chosen to be equally spaced in the interval $[-2, 14]$. Five samples were used for each input scale parameter and nine samples were used for each output scale parameter.

For noisy optimisation, the noise standard deviation σ was marginalised in a similar manner. The samples for σ were chosen to be uniform in the log-space in the interval $[\ln(0.01), 0]$. Five samples for the noise parameter were used. The threshold ϵ used was equal to 1 in all noisy tests.

9.5.3 Discussion

The results of testing are shown in Tables 9.10 and 9.11. Our method performed (or tied with) the best out of the four algorithms tested for thirteen of the sixteen noiseless problems. Additionally, the grand mean performance of our method (even for the simplest model and least informative prior) surpasses the other methods by 16%. Clearly GPGO provides an improvement over other state-of-the-art solutions for global optimisation of expensive functions.

Our performance can be partially explained by our proper handling of hyperparameters. By utilising sequential Bayesian quadrature methods we can intelligently

Table 9.10: Mean measure of performance G for various global optimization techniques on noiseless objective functions. Numbers highlighted in bold are the highest for the relevant problem. The GPGO method is further specified by the use of 1- or 2-step lookahead (1/2) and the use of a non-periodic or periodic covariance function (NP/P).

	Algorithm					
Problem	EGO	RBF	DIRECT	GPGO–1NP	GPGO–1P	GPGO–2NP
A2	0.347	0.703	0.675	0.606	0.612	**0.781**
A5	0.192	**0.381**	0.295	0.089	0.161	—
Br	0.943	0.960	0.958	**0.980**	—	—
C6	0.962	0.962	0.940	0.890	—	**0.967**
GK2	0.571	0.567	0.538	**0.643**	—	—
GK3	0.519	0.207	0.368	**0.532**	—	—
G–P	0.783	0.815	**0.989**	0.804	—	**0.989**
G2	0.979	**1.000**	0.981	**1.000**	**1.000**	—
G5	**1.000**	0.998	0.908	0.925	0.957	—
H3	0.970	0.867	0.868	**0.980**	—	—
H6	0.837	0.701	0.689	**0.999**	—	—
R	0.652	0.647	0.776	0.675	**0.933**	—
Sh5	0.218	0.092	0.090	**0.485**	—	—
Sh7	0.159	0.102	0.099	**0.650**	—	—
Sh10	0.135	0.100	0.100	**0.591**	—	—
Shu	**0.492**	0.383	0.396	0.437	0.348	0.348
mean	0.610	0.593	0.604	**0.705**	—	—

Table 9.11: Mean measure of performance G for various noisy global optimization techniques on noisy objective functions. Objective function outputs were corrupted with Gaussian noise with the indicated standard deviation. Numbers highlighted in bold are the highest for the relevant problem.

	$\sigma = 0.1$			$\sigma = 0.2$			$\sigma = 0.5$		
Problem	SNOBFIT	IF	GPGO	SNOBFIT	IF	GPGO	SNOBFIT	IF	GPGO
A2	0.738	0.051	**0.768**	0.713	0.061	**0.808**	0.792	0.055	**0.902**
A5	**0.499**	0.053	0.293	**0.577**	0.051	0.369	**0.482**	0.043	0.269
Br	0.980	**0.999**	0.996	0.983	**0.999**	0.997	0.979	**0.998**	0.993
C6	0.997	0.994	**0.998**	**0.999**	0.996	0.998	0.995	0.994	**0.997**
G–P	0.998	0.997	**1.000**	0.998	0.997	**1.000**	0.998	0.997	**1.000**
G2	**1.000**	0.320	**1.000**	**1.000**	0.320	**1.000**	**1.000**	0.320	**1.000**
G5	**1.000**	0.195	**1.000**	**1.000**	0.195	**1.000**	**1.000**	0.195	**1.000**
H3	0.956	0.954	**0.977**	0.972	0.921	**0.984**	0.881	0.563	**0.945**
H6	0.797	0.733	**0.936**	**0.864**	0.537	0.615	0.602	0.000	**0.864**
R	0.000	**0.988**	0.000	0.000	**0.988**	0.000	0.030	**0.632**	0.000
Sh5	0.066	0.001	**0.134**	0.128	0.002	**0.309**	0.001	0.002	**0.221**
Sh7	0.118	0.038	**0.158**	0.115	0.034	**0.278**	**0.052**	0.015	0.047
Sh10	0.133	0.284	**0.288**	0.130	0.100	**0.351**	0.032	0.002	**0.082**
Shu	0.667	0.417	**0.669**	0.488	0.452	**0.919**	**0.610**	0.419	0.590
mean	0.638	0.501	**0.658**	0.640	0.474	**0.696**	0.596	0.373	**0.636**

select where to place function evaluations from the very beginning, learning more both about the model hyperparameters and the objective function with each new observation. This is in contrast with the naïve MLE or MAP approaches to hyperparameter management used by competing methods. Additionally, an improvement in performance can be seen when using a more informative prior covariance for periodic functions, illustrating the power of incorporating all available prior knowledge. We finally see an improvement when relaxing the strict myopic assumption. Combining our best results for each problem (using a periodic covariance or multi-step lookahead, as appropriate), our grand mean performance increases to 0.755, a 24% improvement over the closest tested competitor.

Our ability to exploit prior information is also effective in optimising functions corrupted by noise. As can be seen, our algorithm significantly outperformed tested competitors, exceeding or equalling their performance in twenty-nine of the forty-two problems. This is even more remarkable given the myopic approximation (6.6.1) used for testing. The use of a more sophisticated multi-step lookahead algorithm can be expected to even further improve our performance.

9.5.4 Sensor set selection

We now turn to a demonstration of the use of GPGO for set selection, as proposed in Section 6.8. Consider the problem of selecting the optimal locations for measurements to be made of some dynamic spatial process. This problem, studied under the name of *spatial sampling design* [Olea, 1984, Zhu and Stein, 2006] is encountered in a wide range of applications, including soil science, petroleum geology, and oceanography.

We test our algorithm on the UK Meteorological Office MIDAS land surface stations data [British Atmospheric Data Centre, 2009]. In particular, we chose a dispersed set of 50 sensors (whose locations are plotted in Figure 9.21) which recorded meteorological measurements for every day between 1959 and 2009, inclusive. We focus on the network's readings of maximum daily temperature. Our goal is to select the optimal subset of 5 sensors for making global predictions about this dynamic field.

Equipped with an appropriate distance between point sets for the task at hand, we describe our algorithm for optimising the objective function over subsets of sensor locations. In this case, the domain X whose power set we wish to consider is the chosen set of 50 sensors. The function f to be optimised is a measure of the effectiveness of a selection of sensors. The particular measure of performance used in our experiments was the root mean squared error (RMSE) of the predictions made by using observations from the chosen sensors to predict the maximum temperature at every sensor location in the UK.

Specifically, by varying the sensor set, we aim to minimise the objective function $\mathrm{RMSE}([S, t_0]; \Delta)$, where the inputs are an initial time t_0 and a set of sensors S. Here, RMSE returns the root mean squared error of the predictions made about a relevant field at all available locations at the (discretised) set of test times from t_0 to $t_0 + \Delta$.

For the MIDAS data, we began a new trial at the beginning of each period of $\Delta = 28$ days. We select a set of 5 sensors, and then perform zero-step lookahead prediction for the temperatures at all 50 sensors at each of the subsequent 28 days. As a means of testing the effectiveness of our algorithm, we force our algorithm to choose its best subset every five years, and evaluate the performance of predictions made over the entire subsequent year.

Our predictions were made using a GP sequentially trained on the readings at all test times. By this, we mean that the GP possesses observations from S at all available times in the range t_0 to $t_\star$, inclusive, when making predictions about that test time $t_\star$. That is, we perform zero-step lookahead. Prediction about future times (that is, the use of a non-zero lookahead) is, of course, possible if an application calls for such an objective function. If necessary, the prediction algorithm could also be granted a "burn-in" period so that the RMSE is not dominated by poor predictions made when very little data is available.

For the GP used to perform temperature predictions, we used a simple covariance that was a product of a term over time and a term over space. Both covariances were taken to be rational quadratics (3.2.6), the former using the simple

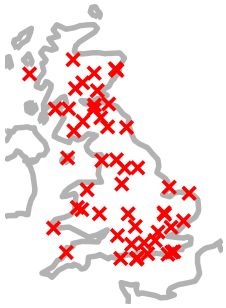


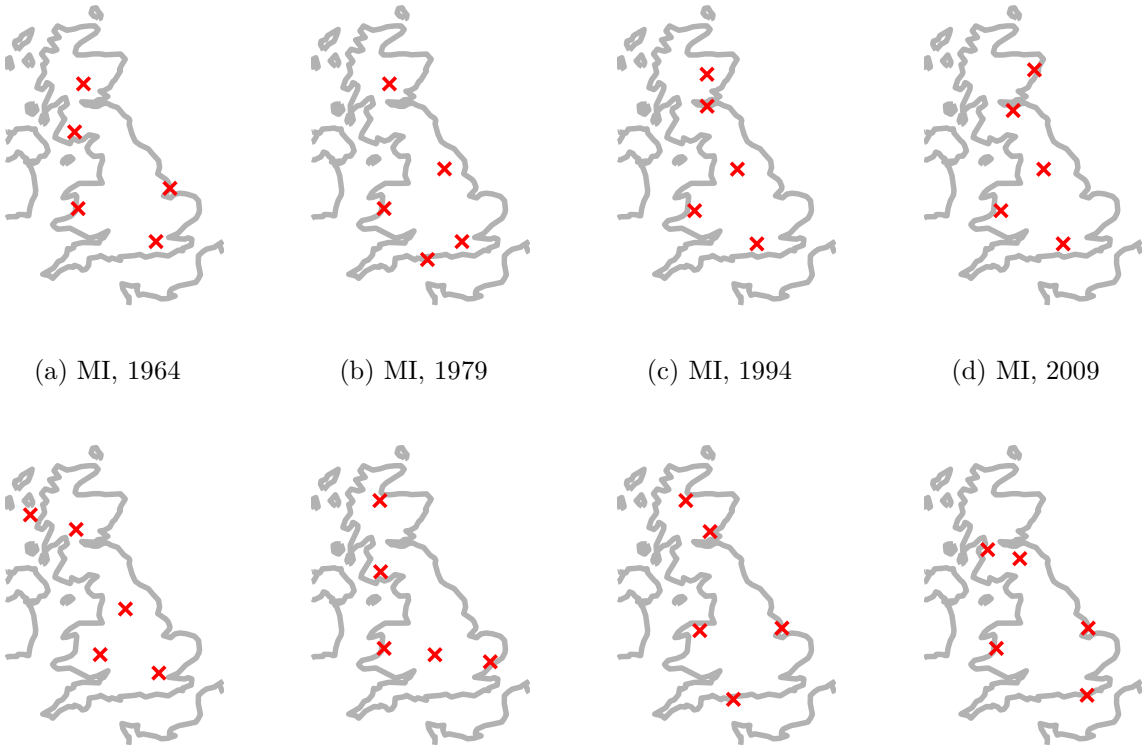
Figure 9.21: For our experiments over the MIDAS data, the locations of all 50 sensors.

distance $d^{(\text{simple})}$ from (3.2.2), and the latter using the great-circle distance between points on the surface of the earth. The hyperparameters of this GP were sequentially marginalised using GBQ. The emphasis of our experiments was not to make the most accurate predictions possible (for which we would recommend more sophisticated non-stationary covariances), but rather to perform sensor selection given any arbitrary black-box objective function.

The covariance function used for GPGO was taken as a product of a term over time and a term over sets. Both were taken as squared exponentials (3.2.3), the the former using the simple distance $d^{(\text{simple})}$ from (3.2.2). For the latter, we used our weighted EMD, $d^{(\text{EMD})}$ from (4.8.1) and (4.8.2). The hyperparameters of these covariances were sequentially marginalised using GBQ. We applied search heuristics as suggested in Section 6.8, minimising the expected loss at approximately 40 000 points for each selection. This includes all subsets that differ in only a single point from the current subset, along with many other, randomly chosen, subsets included for exploration.

We concentrate here on the selection of sets of a pre-selected, fixed size (5 sensors). We search over sets of that size when selecting the set at which our next evaluation of the objective function will be made.

To provide a comparison for our algorithm, we firstly tested against a simple random-selection algorithm. For this, subsets were randomly chosen for each trial



(a) MI, 1964 (b) MI, 1979 (c) MI, 1994 (d) MI, 2009

(e) GPGO, 1964 (f) GPGO, 1979 (g) GPGO, 1994 (h) GPGO, 2009

Figure 9.22: For our experiments over the MIDAS data, the locations of sensors selected at the years indicated by (a)–(d), MI and (e)–(h), GPGO.

period of 28 days. For each test year, we selected the subset that yielded the best RMSE performance in the preceding five years.

We also implemented the method outlined in Krause et al. [2008], henceforth referred to as the mutual information (MI) algorithm. Using it, sensors were greedily selected (up to the allowed number of 5 sensors) to maximise the mutual information between the subsequently evaluated locations in space-time (that is, observations at the positions of the selected sensors, once a day for 28 days) and the locations of all of the other 50 sensors for each of the forthcoming 28 days. This mutual information was determined using the same GP used to provide predictions. Specifically, the MI method's selection was made according to the posterior covariance of this GP (in particular, its highest weighted hyperparameter sample).

The results of our algorithm are shown in Table 9.5.4. Overall, our algorithm was able to provide better subsets for the purposes of maximising sensor network performance over time, providing sets with the best predictive accuracy for all bar one of the ten selected test years. As can be seen, our approach also yielded an overall RMSE that was significantly better than either tested alternative.

The subsets selected by both MI and our algorithm for a selected number of test years are displayed in Figure 9.22. All selected sets seem intuitively to provide good overall coverage of the UK. In that there is any deviation from this general spread of sensors, note that our algorithm selects a greater number on the west coast of the UK. It is possible that sensors in this region were determined to be useful for predicting temperatures via their good performance on the prediction task (weather predominantly moves from west-to-east in the UK). This nuance was automatically captured by our algorithm.

Our algorithm was able to explore the subset space reasonably well and locate reasonable optima, despite few observations (about 13 per year for 50 years from a total number of $\binom{50}{5} = 2\,118\,760$ possibilities). Additionally, our algorithm was able to adapt to information obtained by observing the performance of various subsets through time. Finally, by using a covariance defined on both space and time, our

Table 9.12: The root mean squared error (in degrees Celsius) of the GP prediction algorithm using subsets selected by our subset optimisation routine and the MI algorithm for tracking the weather in the UK for selected years.

	Method		
Year	Random	MI	GPGO
1964	2.9416	3.0182	**2.4408**
1969	3.0295	3.1553	**2.7702**
1974	2.6945	2.8208	**2.4310**
1979	2.8359	2.8238	**2.8143**
1984	2.9156	3.0690	**2.6249**
1989	2.9538	3.0845	**2.6438**
1994	3.0370	2.9886	**2.8252**
1999	3.0223	3.5081	**2.8561**
2004	2.9548	3.0695	**2.6886**
2009	3.0111	**2.8051**	2.8981
mean	2.9396	2.8256	**2.6993**

algorithm can cope with changes to the performance of various subsets over time (for example, if a sensor becomes faulty), allowing for us to react to *concept drift* in the prediction problem.

9.5.5 Running time

We now consider the computational costs incurred in performing GPGO. Suppose that we have chosen h hyperparameter samples. Incorporating the Nth objective function evaluation into our optimisation GP requires making rank-one updates to h Cholesky factorisations of size $(N-1)^2$, and each training step thus runs in $O(hN^2)$ time. Evaluating the expected loss at k points takes time $O(khN^2)$.

In practice, a somewhat inefficient implementation of our algorithm took approximately 120 seconds to train the GPGO model (with 100 observations) and find the point of minimum expected loss (of 40 000 evaluations). 1 000 hyperparameter samples were used, and the context was the set selection experiment described in Section 9.5.4. The machine was an Apple Mac Pro workstation with two 2.26 GHz

quad-core Intel Xeon "Gainestown" processors (model E5520) and 16 GB of RAM, running the Linux kernel (version 2.6.31).

Of course, it should be noted that our methods require significantly more computation than their competitors. If we use a total of h hyperparameter samples, one-step lookahead GPGO is roughly h times slower than the slowest competitor, EGO, which uses only one for its online operation. Two-step lookahead requires us to perform an additional sampling and minimisation stage for each evaluation of the expected loss, meaning that it will be roughly several thousand times slower than one-step lookahead. Nonetheless, the efficacy of our methods justifies their use for applications for which the cost of evaluating the objective function is very high. For applications such as sensor network optimisation, where we might have to wait a month to evaluate the performance of a particular arrangement, the evaluation cost of the objective function completely swamps any costs due to the optimisation procedure.

9.6 Sampling for Bayesian Quadrature

We now compare our SBQ proposal from Section 8.4 against two competitors. First, we used a simple gradient-ascent scheme to determine a maximum-likelihood hyperparameter sample (ML). The likelihood is then essentially approximated as a delta function at that sample. Second, we considered hybrid (Hamiltonian) Monte Carlo [Duane et al., 1987, Neal, 1993] (HMC), which, similar to our algorithm, exploits information about the gradient of the likelihood surface. HMC uses samples generated from $p\big(\phi|\boldsymbol{z}_d, I\big)$ directly to evaluate (7.3.1), as per (7.3.4) and Section 8.2. We tested these methods in three contexts.

As a toy example, we first approximated a simple ratio of (stationary) integrals of the form (3.4.1). Both the likelihood r and prediction q surfaces were chosen to be sums of (up to three) Gaussians of varying parameters; permitting us to analytically determine the correct value for the ratio. We allowed 100 evaluations of the q and r functions, with only 7 permitted to evaluate the integrals, as befits the con-

text considered in which evaluating a large number of samples is computationally onerous. With so few samples, exploring hyperparameter space was a challenge of some difficulty, with Gaussians often not located and significant error introduced as a consequence. Averaging over 10 experiments, SBQ's estimates were found to differ from the real value by only 48% of its value, compared to 113% for ML and 121% for HMC. In this deliberately difficult challenge, our method clearly outperformed the competitors. We now turn to the context by which SBQ is truly motivated, sampling for sequential problems.

9.6.1 Online Prediction

We next performed one-step lookahead prediction over various time-series prediction problems. Our prediction model was a Gaussian process, for which we must marginalise the (initially unknown) hyperparameters of its covariance.

First, we tested on an artificially generated periodic function $y(x)$, marginalising over the period (known in reality to be 1). More precisely, we supplied each algorithm with 80 initial observations of $y(x)$ (up to $x = 5$) and five initial hyperparameter samples at challenging locations on the resulting likelihood surface, as illustrated in Figure 9.23. Our algorithms then had to adjust the locations and weights (in the sense of (7.3.17)) associated with those five samples in order to best perform one-step lookahead prediction about the function. The prediction effected by the algorithm fitted with SBQ is depicted in Figure 9.24 and the sample locations and weights in Figure 9.25. Despite the multimodal likelihood surface, the SBQ algorithm quickly determines the correct period, placing a very highly weighted sample close to $\log T = 0$ shortly before a single period is completed. Subsequent to that, other samples are used to explore the likelihood space. Accurate prediction for y is achieved as a consequence.

We also tested on three datasets drawn from the Bramblemet weather sensor network discussed in Section 9.2. In particular, we tested on two single-sensor wind-speed datasets, using the covariance (9.4.1), for which we were required to marginalise

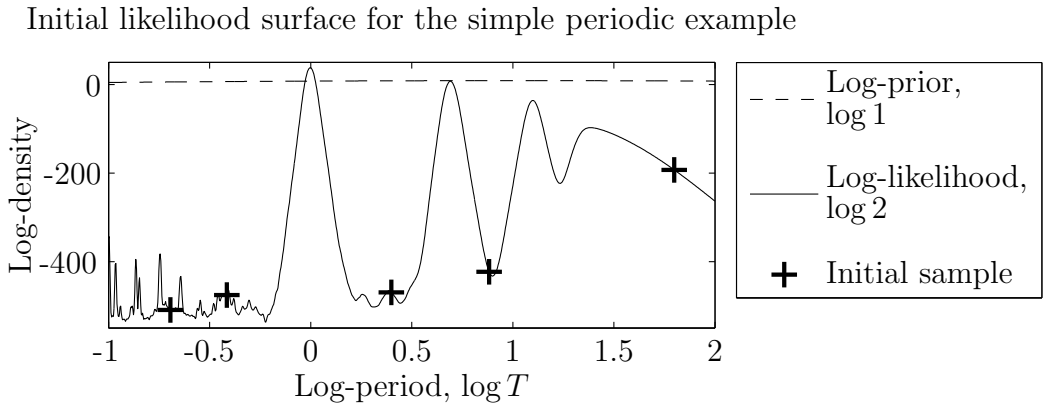


Figure 9.23: The log-likelihood $\log p\big(\boldsymbol{z}_d|\phi, I\big)$ after 80 observations $\boldsymbol{z}_d$ (along with the log-prior, $\log p\big(\phi|I\big)$) from the simple periodic function as a function of the log-period hyperparameter $\log(T) \in \phi$. Note the highly multimodal nature of the surface, even in the regions between harmonics of the real period of $1 = \exp 0$. The initial hyperparameter samples supplied to the various algorithms were chosen to lie at particularly problematic positions.

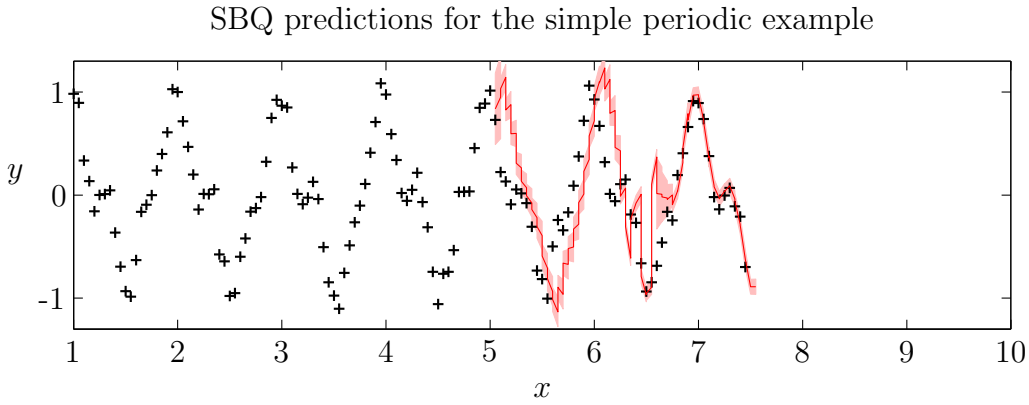


Figure 9.24: The predictions made using SBQ on the simple periodic function.

over four input- and output-scale hyperparameters. We also tested over an air temperature dataset, with covariance (9.2.2), for which we were required to marginalise over the six hyperparameters specifying the strength of the correlations among three sensors. All other hyperparameters were learned offline. Results are tabulated in Table 9.13.

9.6.2 Global optimisation

We also used SBQ to perform optimisation using GPGO, as per Chapter 6. The Gaussian process model used as a surrogate for this optimisation typically has several

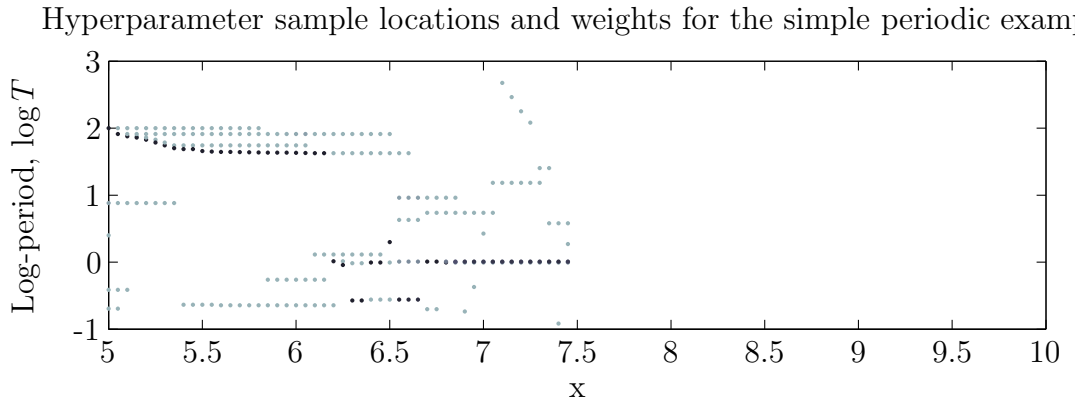


Figure 9.25: The locations and weights (the darker, the more highly weighted) of hyperparameter samples selected by SBQ for prediction of the simple periodic function. Note the real period, $1 = \exp 0$, is found rapidly, but exploration continues.

Table 9.13: RMSE (normalised by the dynamic range of the data) for four tracking datasets for four optimisation datasets using three different hyperparameter management techniques: sequential maximum likelihood (ML), hybrid Monte Carlo (HMC), and sampling for Bayesian quadrature (SBQ). Best result for each in bold.

	Method		
Problem	ML	HMC	SBQ
Periodic	0.167	0.207	**0.032**
Wind-speed 1	**0.110**	0.291	0.118
Wind-speed 2	0.075	0.272	**0.066**
Air Temp	0.043	**0.043**	0.043

Table 9.14: Mean gap measure for four optimisation datasets using three different hyperparameter management techniques: sequential maximum likelihood (ML), hybrid Monte Carlo (HMC), and sampling for Bayesian quadrature (SBQ). Best result for each in bold.

	Method		
(dim)/(# minima)	ML	HMC	SBQ
2/10	0.565	**0.658**	0.510
2/20	0.576	0.548	**0.711**
3/10	0.392	0.399	**0.406**
3/20	0.396	0.442	**0.478**

hyperparameters that need to be marginalised. Nonetheless our methods continue to apply. Here the test set comprised ten objective functions in each of four different sets: in each pair of $\{2, 3\}$-dimensions and with $\{10, 20\}$ local minima. The objective functions were generated randomly with the GKLS algorithm [Gaviano et al., 2003]. The measure of performance was the average value of the gap measure from (9.5.1). The results are found in Table 9.14.

9.6.3 Running time

As with GPGO, SBQ typically requires significantly more computation to select samples than its competitors. However, its performance justifies its use where the cost of evaluating the integrand is itself very expensive. This might be the case, for example, if we were fitting a very large dataset with a GP, such that evaluating the likelihood of any given hypersample required the expensive Cholesky factorisation of a very large covariance matrix. Indeed, in the limit that this evaluation cost becomes very significant, the more intelligent selection procedure of SBQ will allow our algorithm to evaluate fewer samples and hence give it even faster performance than competitors. Note that both Monte Carlo and maximum likelihood techniques evaluate a very large number of samples that are not ultimately used in performing inference about our integral. BQ, on the other hand, is able to make full use of all samples gathered.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

Armed with the methods above, we return now to the list of challenges presented by sensor networks (Section 1.1). We began by addressing the inference problems of sequential GP prediction and regression, with various proposals made towards allowing real-time computation. The flexible GP framework permits the use of sophisticated models of the multiple correlated variables monitored by sensor networks, even with limited prior information. We can also manage various problems with and failures of the sensor network itself. In particular, we have seen how our GP algorithm is able to make accurate prediction even where data is missing or delayed. In this vein, we have also tackled data that possesses changepoints and faults, using covariance functions designed for the relevant purpose. Our Bayesian approach allows for full posterior distributions to be produced for both predictants and the locations of changepoints/faults.

We next turned to the decision problems presented by data acquisition in sensor networks. Over data both with and without changepoints, we demonstrated active data selection, whereby our algorithm selects the optimal observations according to a defined cost of observation. Determining the optimal locations for those sensors in the first place is a more challenging decision problem, our objective function multi-modal and expensive to evaluate. Motivated by such challenges, we introduced GPGO, a global optimisation algorithm built upon Gaussian processes. With the further

addition of novel covariance functions over sets, we demonstrated how to dynamically select the best possible sets of sensors.

The power and flexibility of the GP framework comes at a cost, however. The large numbers of hyperparameters required to specify a complicated model must be marginalised. To tackle this problem, we introduced the SBQ algorithm, a principled method of determining the set of hyperparameter samples best fit to estimating our required integrals.

Our principled methods have all been extensively tested on both real sensor networks and canonical test problems. In all cases, we have demonstrated notable improvements on heuristic alternatives.

10.2 Related Work

It remains to highlight other work that has foreshadowed our own.

Gaussian process regression has a long history of use within geophysics and geospatial statistics (where the process is known as kriging [Cressie, 1991]), but has only recently been applied within sensor networks. One example is the use of multivariate Gaussians to represent correlations between different sensors and sensor types for energy efficient querying of a sensor network [Deshpande et al., 2004].

Our work differs in that we use GPs to model dynamic fields, measured at multiple sensors. It is thus closely related to other work using GPs to perform regression over multiple responses [Teh et al., 2005, Boyle and Frean, 2005]. However, our focus is to derive a computationally efficient algorithm, and thus, we use a number of novel computational techniques to allow the re-use of previous calculations as new sensor observations are made. We additionally use a novel Bayesian Quadrature technique to marginalise the hyperparameters that describe the correlations and delays between sensors. Finally, we use the variance of the GP's predictions in order to perform active data selection.

Our approach has several advantages relative to sequential state-space models

[Girard et al., 2003, Jazwinski, 1970, Wang et al., 2006]. Firstly, these models require the discretisation of the time input, representing a discarding of potentially valuable information. Secondly, their sequential nature means they must necessarily perform difficult iterations in order to manage missing or late data, or to produce long-range forecasts. In our GP approach, what observations we have are readily managed, regardless of when they were made[1]. Equally, the computation cost of all our predictions is identical, irrespective of the time or place we wish to make them about. Finally, a sequential framework requires an explicit specification of a transition model. In our approach, we are able to learn a model from data even if our prior knowledge is negligible. The benefits of our approach relative to a Kalman Filter are empirically supported by comparative experiments.

Previous work has also investigated the use of censored sensor readings within a GP framework [Ertin, 2007], or the similar problems involved in classification and ordinal regression [Chu and Ghahramani, 2006]. Our approach differs in a number of respects. Firstly, we consider the potential combination of both censored and uncensored observations. Our work also proposes a principled Bayesian Quadrature method for adapting our models to the data, considering the contributions from a number of samples in hyperparameter space, rather than simply taking a single sample.

The problem of detecting and locating abrupt changes in data sequences has been studied under the name *changepoint detection* for decades. A large number of methods have been proposed for this problem; see Basseville and Nikiforov [1993], Brodsky and Darkhovsky [1993], Csorgo and Horvath [1997], Chen and Gupta [2000] and the references therein for more information. Relatively few algorithms perform prediction simultaneously with changepoint detection, although sequential Bayesian methods do exist for this problem [Chernoff and Zacks, 1964, Adams and MacKay, 2007]. However, these methods, and most methods for changepoint detection in

[1]Note that if data is not time-stamped, however, then we would need to marginalise out the uncertain ages of those measurements, which will form new hyperparameters of our model.

general, make the assumption that the data stream can be segmented into disjoint sequences, such that in each segment the data represent i.i.d. observations from an associated probability distribution. The problem of changepoints in dependent processes has received less attention. Both Bayesian [Carlin et al., 1992, Ray and Tsay, 2002] and non-Bayesian [Muller, 1992, Horváth and Kokoszka, 1997] solutions do exist, although they focus on retrospective changepoint detection alone; their simple dependent models are not employed for the purposes of prediction. Sequential and dependent changepoint detection has been performed [Fearnhead and Liu, 2007] only for a very limited set of changepoint models. Our GP approach offers a flexible and powerful alternative.

Although the application of GPs to perform seizure detection in EEG signals is not new [Faul et al., 2007], our approach is significantly more principled. We explicitly model the fault using appropriate covariances, and marginalise over the hyperparameters using GBQ. This marginalisation is key to our success: as a consequence, we can characterise the model uncertainty by determining the full posterior for the relevant hyperparameters. These include, in particular, the location and characteristics of changepoints and faults. This uncertainty then is naturally folded into our uncertainty about the signal.

The problems of discarding data, or sparsifaction, [Quiñonero-Candela and Rasmussen, 2005, Lawrence et al., 2003, Kapoor and Horvitz, 2007] and active data selection [MacKay, 1992, Seo et al., 2000] have been the topic of much previous research. Our domain, on-line tracking, simplifies these problems relative to much previous work, owing to the fact that we have a well-defined point of interest. This will either be the current values of our environmental variables, or some set of future values according to the degree of lookahead. Note also that our uncertainty in this value reflects our underlying uncertainty about the correct model, due to our principled marginalisation. More observations will be scheduled and/or fewer discarded if there is a significant degree of uncertainty about the model. We have also performed active data selection over faults and changepoints, our principled approach adapting our sampling to the

changes in our data.

Our GPGO algorithm has clear benefits over older, similar work [Jones et al., 1998, Lizotte, 2008]. Our formulation of optimisation as a sequential decision problem allows us to state the exact Bayes optimal policy for a specified degree of required confidence. This can either be approximated using a simple myopic approach, or, computation permitting, a more thorough multiple-step lookahead policy evaluated. We further use a computationally efficient sequential GP and employ Bayesian Quadrature to give a principled method of managing hyperparameters. This latter fact allows us to extract maximal information from all our function evaluations. Further benefits are found in our ability to exploit prior knowledge about the objective function, as in the cases that we suspect it may be periodic or corrupted by noise. The GP also allows the incorporation of any available derivative observations, giving a means to address conditioning issues.

Kondor and Jebara [2003] present an alternative to our method of constructing a covariance between sets. They do so by empirically fitting Gaussians to each set, and then using the Bhattacharyya covariance between those two Gaussians as a covariance between the sets. This, however, allows no prior information about the structure of the underlying field to be exploited, and is hence not ideal for sensor selection.

Other approaches using GPs for sensor selection [Krause et al., 2008] typically assume greater prior knowledge of how different sensor sets will perform. They are also unable to fully exploit the information yielded by any experimental trials of sensor layouts. In contrast, our formulation allows us great flexibility, allowing the use of as much or as little knowledge that may exist, and permitting its application to problems of many different types. In its full generality, our method aims to optimise any function over point sets.

Finally, our management of hyperparameters (in GBQ and SBQ) owes a great debt to the work of O'Hagan [1991], Minka [2000] and Rasmussen and Ghahramani [2003]. What we term simple Bayesian quadrature is what O'Hagan [1991] called Bayes-Hermite quadrature and Rasmussen and Ghahramani [2003] referred to as Bay-

esian Monte Carlo. We have, however, extended that work to manage a ratio of correlated integrals that occurs commonly in prediction problems, giving rise to what we term predictive Bayesian quadrature. That requires the novel expedient of sampling over entire log-likelihood functions. Determining the optimal sampling strategy for such a ratio of integrals leads to SBQ. SBQ is influenced both by the locations of our samples $\boldsymbol{\phi}_s$ and also the resulting values of our likelihood and prediction functions, $\boldsymbol{r}_s$ and $\boldsymbol{q}_{\star,s}$. The sampling strategy for simple Bayesian quadrature has no such dependence on the actual function evaluations, which seems undesirable for our problem.

10.3 Future Work

We consider the work above to lay the foundations for many future avenues of research.

For our sequential GP tracking problems, our approach will readily allow the integration of data from multiple sources. For example, we might integrate data from wind and tide sensors in order to improve our inference about both. All that would be required in order to effect such experiments is to learn an appropriate covariance between disparate sources, just as we currently learn covariances between sensor locations.

Our fault and changepoint detection experiments could be enriched with some simple extensions. Firstly, we have thus far only considered marginalising over the characteristics of changepoints of known type. By introducing a *changepoint-type* hyperparameter, however, we might readily marginalise over multiple possible types of changepoint that might occur within a data source. This is particularly relevant to fault detection, in which a sensor might be capable of failing in many different modes. If doing so, we might also introduce a *catch-all* fault model, designed to capture any unexpected behaviour not otherwise described by a specified fault model.

A number of extensions to our GPGO work present themselves. Firstly, we may wish to tackle problems for which we have multiple modes of observation. Building

upon our approach to noisy optimisation, this could include functions for which we have a suite of variously costly, but also variously noisy, possible measurements. Applications are to be found in multiple-fidelity optimisation and heterogeneous sensor-selection problems. Similarly, we may have to choose between an observation of the function itself or one of its derivatives, allowing for an interesting alternative to other gradient-based optimisation routines. Essentially, these problems all reduce simply to the introduction of an additional input dimension to our problem, specifying which type of observation we take – a simple extension to the methods discussed above.

In applying our GPGO algorithm to set selection problems, we have additional future possibilities. Firstly, our algorithm could be used to manage a set of mobile sensors. Here it would be trivial to include a cost proportional to the total distance moved by all sensors into the GPGO loss function. We could also define a cost associated with taking an additional sensor, and allow our algorithm to search over sets of all sizes. In effect, our algorithm would select the optimal set size, trading off the cost of larger sets against the potentially better objective function values they may return. Such extensions to the cost function can easily be incorporated into our scheme due to our straightforward Bayesian approach.

Our work on Bayesian quadrature is ripe with potential improvements. Firstly, we might consider the problem of determining the optimal number of samples to take, just as we've just described the problem of determining the optimal number of sensors. Indeed, the analogy here is a deep one: samples inform us of an integrand just as sensors might inform us of a temperature field. As such, our methods for set selection might be deployed in the context of selecting sets of hyperparameter samples. However, as it stands, our SBQ approach acknowledges a great deal of prior information concerning how differing sample sets are likely to benefit our quadrature. It might be difficult to incorporate this into the set selection approach.

On that topic, more sophisticated non-parametric Bayesian priors for our likelihood and prediction functions might also be investigated. The assumptions that we build into such priors are currently limited – we typically assume GPs with simple

covariances, enforcing only smoothness. In reality, we may have a fairly good idea of the shape of the integrand, for example, we might know it is a Gaussian (although not in the variable of integration). It seems undesirable to assume that the only significant information we have about the value of an integral is that contained in various samples of the integrand.

One possible alternative approach is to treat integration itself as an unknown functional, and to perform inference directly about that functional. That means we would use the known integrals of other, similar functions in order to determine what the integral of the function at hand is. Here, of course, the notion of similarity would have to be determined by the various characteristics of the function – just as a human mathematician recognises the structure of a function (that it is the product of two terms, for example) in deciding how to attempt to integrate it analytically. We would like to induce a smooth transition from analytic integration to numerical integration.

Appendix A

Identities

A.1 Gaussian Identities

The following identities [Rasmussen and Williams, 2006] are almost indispensible when dealing with Gaussian distributions, which we denote in the usual way as

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{A}) \triangleq \frac{1}{\sqrt{\det 2\pi\mathbf{A}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}} \mathbf{A}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right) .$$

Probably the most important properties of a multivariate Gaussian distribution are that its marginals and conditionals are both themselves Gaussian. That is, if we have

$$p(\boldsymbol{x}, \boldsymbol{y} \mid I) \triangleq \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\nu}\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^{\mathsf{T}} & \mathbf{B}\end{bmatrix}\right) ,$$

then its marginal and conditional distributions are respectively

$$p(\boldsymbol{x} \mid I) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{A}) \tag{A.1.1}$$

$$p(\boldsymbol{x} \mid \boldsymbol{y}, I) = \mathcal{N}\big(\boldsymbol{x}; \boldsymbol{\mu} + \mathbf{C}\,\mathbf{B}^{-1}(\boldsymbol{y}-\boldsymbol{\nu}), \mathbf{A} - \mathbf{C}\,\mathbf{B}^{-1}\,\mathbf{C}^{\mathsf{T}}\big) . \tag{A.1.2}$$

We now turn to another important property of the Gaussian distribution, illustrated in Figure A.1. If we have

$$\begin{aligned} p(\boldsymbol{x} \mid I) &\triangleq \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{A}) \\ p(\boldsymbol{y} \mid \boldsymbol{x}, I) &\triangleq \mathcal{N}(\boldsymbol{y}; \mathbf{M}\,\boldsymbol{x} + \boldsymbol{c}, \mathbf{L}) , \end{aligned} \tag{A.1.3}$$

then the joint distribution can be written as

$$p(\boldsymbol{x}, \boldsymbol{y} \mid I) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \mathbf{M}\,\boldsymbol{\mu} + \boldsymbol{c}\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{A}\,\mathbf{M}^{\mathsf{T}}\\ \mathbf{M}\,\mathbf{A} & \mathbf{L} + \mathbf{M}\,\mathbf{A}\,\mathbf{M}^{\mathsf{T}}\end{bmatrix}\right) , \tag{A.1.4}$$

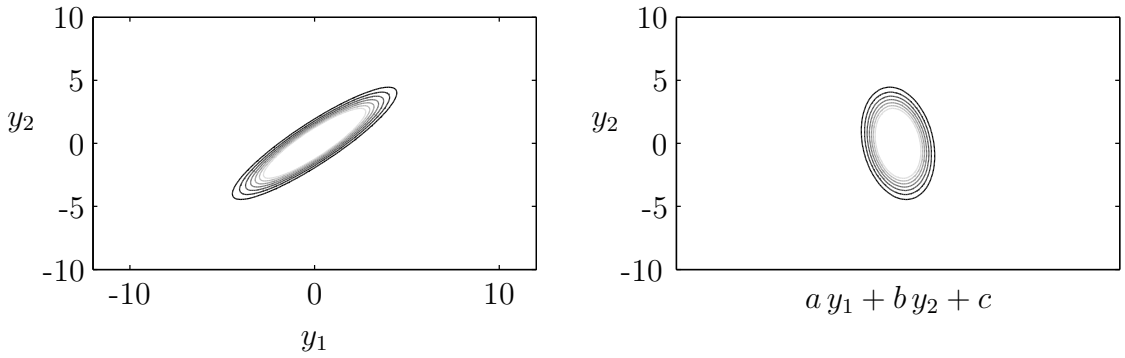


Figure A.1: Any variables over which we have a multivariate Gaussian are jointly Gaussian with any linear transformation of those variables.

and so, using (A.1.1) and (A.1.2)

$$p(\,\boldsymbol{y} \mid I\,) = \mathcal{N}\big(\boldsymbol{y}; \mathbf{M}\,\boldsymbol{\mu} + \boldsymbol{b}, \mathbf{L} + \mathbf{M}\,\mathbf{A}\,\mathbf{M}^{\mathsf{T}}\big) \tag{A.1.5}$$

$$p(\,\boldsymbol{x} \mid \boldsymbol{y},\, I\,) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu} + \boldsymbol{\Gamma}(\boldsymbol{y} - \mathbf{M}\,\boldsymbol{\mu} - \boldsymbol{c}), \mathbf{A} - \boldsymbol{\Gamma}\,\mathbf{M}\,\mathbf{A}) \tag{A.1.6}$$

where

$$\boldsymbol{\Gamma} = \mathbf{A}\,\mathbf{M}^{\mathsf{T}}\left(\mathbf{L} + \mathbf{M}\,\mathbf{A}\,\mathbf{M}^{\mathsf{T}}\right)^{-1}$$

Similarly, note that a Gaussian distribution can be rewritten by applying the change of coordinates given by any unitary matrix $\mathbf{M}$

$$p(\,\boldsymbol{x} \mid I\,) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{A}) = \mathcal{N}\big(\mathbf{M}\,\boldsymbol{x}; \mathbf{M}\,\boldsymbol{\mu}, \mathbf{M}\,\mathbf{A}\,\mathbf{M}^{\mathsf{T}}\big)\ .$$

That is, if we transform using a unitary matrix $\mathbf{M}$, as per

$$p(\,\boldsymbol{x} \mid I\,) \triangleq \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{A})$$

$$p(\,\boldsymbol{y} \mid \boldsymbol{x},\, I\,) \triangleq \delta(\boldsymbol{y} - \mathbf{M}\,\boldsymbol{x})\ ,$$

we have

$$p(\,\mathring{\boldsymbol{x}} = \boldsymbol{x} \mid I\,) = p(\,\mathring{\boldsymbol{y}} = \mathbf{M}\,\boldsymbol{x} \mid I\,)$$

.

Finally, imagine that we have a probability distribution that is a weighted mixture of Gaussians

$$p(\,\boldsymbol{x} \mid I\,) = \sum_i \rho_i\,\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_i}, \mathbf{A}_i)\ .$$

The mean of this mixture is

$$\boldsymbol{m}(\mathring{\boldsymbol{x}}\,|\,I) \triangleq \int \boldsymbol{x}\, p(\,\boldsymbol{x}\mid I\,)\;\mathrm{d}\boldsymbol{x} = \sum_i \rho_i\,\boldsymbol{\mu}_i\,, \tag{A.1.7}$$

and its covariance is

$$\begin{aligned}\mathbf{C}(\mathring{\boldsymbol{x}}\,|\,I) &\triangleq \int \boldsymbol{x}\,\boldsymbol{x}^{\mathsf{T}}\, p(\,\boldsymbol{x}\mid I\,)\;\mathrm{d}\boldsymbol{x} - \boldsymbol{m}(\mathring{\boldsymbol{x}}\,|\,I)\;\boldsymbol{m}(\mathring{\boldsymbol{x}}\,|\,I)^{\mathsf{T}} \\ &= \sum_i \rho_i\,\left(\mathbf{A}_i + \boldsymbol{\mu}_i\,\boldsymbol{\mu}_i^{\mathsf{T}}\right) - \boldsymbol{m}(\mathring{\boldsymbol{x}}\,|\,I)\;\boldsymbol{m}(\mathring{\boldsymbol{x}}\,|\,I)^{\mathsf{T}}\;. \end{aligned} \tag{A.1.8}$$

A.2 Kronecker Identities

We use $\mathbf{I}_d$ to represent the identity matrix of dimension d. Similarly, $\mathbf{1}_{m,n}$ is a matrix containing all ones of dimensions $m \times n$; $\mathbf{1}_d$ is the square matrix of dimension d containing all ones. Now for a simple use of the mixed product property of the Kronecker product. Consider any matrix $\mathbf{A}$ with row (first) dimension Q (note that $\mathbf{A}$ could be a column vector of dimension Q), the mixed-product property of the Kronecker product gives:

$$\begin{aligned}\mathbf{1}_{P,1} \otimes \mathbf{A} &= (\mathbf{1}_{P,1}\,\mathbf{1}_{1,1}) \otimes (\mathbf{I}_Q\,\mathbf{A}) \\ &= (\mathbf{1}_{P,1} \otimes \mathbf{I}_Q)\,(\mathbf{1}_{1,1} \otimes \mathbf{A}) \\ &= (\mathbf{1}_{P,1} \otimes \mathbf{I}_Q)\,\mathbf{A} \end{aligned} \tag{A.2.1}$$

Similarly, it can be shown that for a matrix $\mathbf{B}$ with column (second) dimension Q

$$\mathbf{1}_{1,P} \otimes \mathbf{B} = \mathbf{B}\,(\mathbf{1}_{1,P} \otimes \mathbf{I}_Q) \tag{A.2.2}$$

Appendix B

Cholesky Factor Updates and Downdates

B.1 Cholesky Factor Update

We have a positive definite matrix, represented in block form as

$$\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix}$$

and its Cholesky factor,

$$\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}.$$

Given a new positive definite matrix, which differs from the old only in the insertion of some new rows and columns,

$$\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathsf{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathsf{T}} & V_{2,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix},$$

we wish to efficiently determine its Cholesky factor,

$$\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}.$$

For $\mathbf{A}$ triangular, we define $\boldsymbol{x} = \mathbf{A} \setminus \boldsymbol{b}$ as the solution to the equations $\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b}$ as found by the use of backwards or forwards substitution. The following rules are

readily obtained

$$S_{1,1} = R_{1,1} \tag{B.1.1}$$

$$S_{1,2} = R_{1,1}^{\mathsf{T}} \setminus V_{1,2} \tag{B.1.2}$$

$$S_{1,3} = R_{1,3} \tag{B.1.3}$$

$$S_{2,2} = \mathrm{chol}(V_{2,2} - S_{1,2}^{\mathsf{T}} S_{1,2}) \tag{B.1.4}$$

$$S_{2,3} = S_{2,2}^{\mathsf{T}} \setminus (V_{2,3} - S_{1,2}^{\mathsf{T}} S_{1,3}) \tag{B.1.5}$$

$$S_{3,3} = \mathrm{chol}(R_{3,3}^{\mathsf{T}} R_{3,3} - S_{2,3}^{\mathsf{T}} S_{2,3})\,. \tag{B.1.6}$$

By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the insertion of rows and columns in any position. It's worth noting the equivalent method of updating a matrix inverse is given by use of the *inversion by partitioning* formulae [Press et al., 1992, Section 2.7].

B.2 Data Term Update

We have all terms defined in Section B.1, in addition to the data

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$$

and the product

$$\begin{bmatrix} C_1 \\ C_3 \end{bmatrix} \triangleq \begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_3 \end{bmatrix}.$$

To efficiently determine the updated product

$$\begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} \triangleq \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix},$$

we have

$$D_1 = C_1 \tag{B.2.1}$$

$$D_2 = S_{2,2}^{-\mathsf{T}} \left(Y_2 - S_{1,2}^{\mathsf{T}} C_1 \right) \tag{B.2.2}$$

$$D_3 = S_{3,3}^{-\mathsf{T}} \left(R_{3,3}^{\mathsf{T}} C_3 - S_{2,3}^{\mathsf{T}} D_2 \right). \tag{B.2.3}$$

B.3 Log-Gaussian Update

We have all terms defined in Sections B.1 and B.2, in addition to

$$\begin{aligned} K &= \log \mathcal{N}\left(\begin{bmatrix} Y_1 \\ Y_3 \end{bmatrix} ; \begin{bmatrix} 0 \\ 0 \end{bmatrix} , \begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix} \right) \\ &= -\frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} R_{1,1}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} R_{3,3}) - \frac{1}{2} C_1^{\mathsf{T}} \, C_1 - \frac{1}{2} C_3^{\mathsf{T}} \, C_3 \, . \end{aligned}$$

We can then calculate the updated term

$$\begin{aligned} L &= \log \mathcal{N}\left(\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} ; \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} , \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathsf{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathsf{T}} & V_{2,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix} \right) \\ &= -\frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} S_{1,1}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} S_{2,2}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} S_{3,3}) \\ &\quad - \frac{1}{2} D_1^{\mathsf{T}} \, D_1 - \frac{1}{2} D_2^{\mathsf{T}} \, D_2 - \frac{1}{2} D_3^{\mathsf{T}} \, D_3 \\ &= K + \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} R_{3,3}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} S_{2,2}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} S_{3,3}) \\ &\quad + \frac{1}{2} C_3^{\mathsf{T}} \, C_3 - \frac{1}{2} D_2^{\mathsf{T}} \, D_2 - \frac{1}{2} D_3^{\mathsf{T}} \, D_3 \end{aligned} \tag{B.3.1}$$

B.4 Cholesky Factor Downdate

We have a positive definite matrix, represented in block form as

$$\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathsf{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathsf{T}} & V_{2,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix}$$

and its Cholesky factor,

$$\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix} .$$

Given a new positive definite matrix, which differs from the old only in the deletion of some new rows and columns,

$$\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathsf{T}} & V_{3,3} \end{bmatrix} ,$$

we wish to efficiently determine its Cholesky factor

$$\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix} .$$

The following rules are readily obtained

$$R_{1,1} = S_{1,1} \tag{B.4.1}$$

$$R_{1,3} = S_{1,3} \tag{B.4.2}$$

$$R_{3,3} = \mathrm{chol}(S_{2,3}^{\mathsf{T}} S_{2,3} + S_{3,3}^{\mathsf{T}} S_{3,3})\,. \tag{B.4.3}$$

Note that the special structure of equation (B.4.3) can be exploited for the efficient resolution of the required Cholesky operation, as, for example, in the MATLAB function `cholupdate` [The MathWorks, 2007]. By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the deletion of rows and columns in any position.

Appendix C

Kalman Filters

C.1 A Kalman Filter is a Gaussian Process

Imagine that we have a Markov chain, as illustrated in Figure C.1. A Kalman Filter (KF) is defined as the probability distribution for which

$$p(\, f_1 \mid I \,) \triangleq \mathcal{N}(f_1; \nu, \Lambda) \tag{C.1.1a}$$

$$p(\, f_k \mid f_{k-1},\, I \,) \triangleq \mathcal{N}(f_k; G\, f_{k-1}, Q) \tag{C.1.1b}$$

$$p(\, y_k \mid f_k,\, I \,) \triangleq \mathcal{N}(y_k; H\, f_k, R) \tag{C.1.1c}$$

Here both f_k and y_k may in fact be vector valued. Considering time series analysis, this allows $f_2 \triangleq (g_4, g_3, g_2)$ to be dependent purely on $f_1 \triangleq (g_3, g_2, g_1)$, for example – our simple framework (C.1.1) is equally applicable to higher-order Markov chains. Now, Gaussian distributed variables are joint Gaussian with any linear transformation of them, as per (A.1.4). As such, all our variables, $\boldsymbol{f} \triangleq (f_1, f_2, f_3 \ldots)$ and $\boldsymbol{y} \triangleq (y_1, y_2, y_3, \ldots)$ are joint Gaussian

$$p(\, \boldsymbol{f},\, \boldsymbol{y} \mid I \,) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{y} \end{bmatrix} ; \begin{bmatrix} \boldsymbol{\mu}_f \\ \boldsymbol{\mu}_y \end{bmatrix} , \begin{bmatrix} \mathbf{K}_f & \mathbf{K}_{f,y} \\ \mathbf{K}_{f,y}^{\mathsf{T}} & \mathbf{K}_y \end{bmatrix} \right) \tag{C.1.2}$$

where our mean vector is generated by the functions

$$\mu_f(t_k) \triangleq G^k \nu \tag{C.1.3a}$$

$$\mu_y(t_k) \triangleq H G^k \nu \tag{C.1.3b}$$

and our covariance matrix by

$$K_f(t_j, t_k) \triangleq G^j \, \Lambda \, G^{\mathsf{T},k} + \sum_{a=1}^{\min(j,k)} G^{j-a} \, Q \, G^{\mathsf{T},k-a} \tag{C.1.4a}$$

$$K_{f,y}(t_j, t_k) \triangleq K_f(t_j, t_k) \, H^{\mathsf{T}} \tag{C.1.4b}$$

$$K_y(t_j, t_k) \triangleq H \, K_f(t_j, t_k) \, H^{\mathsf{T}} + \delta_{j,k} \, R \tag{C.1.4c}$$

As such, the KF is a Gaussian process (GP) over $\boldsymbol{f}$ and $\boldsymbol{y}$ with mean and covariance functions given by (C.1.3) and (C.1.4) respectively. This covariance is defined by the process and observational models specified in (C.1.1). The KF covariance differs significantly from the kinds of covariance functions traditionally employed in GP inference, functions such as the squared exponential and rational quadratic. In particular, note that the covariance of (C.1.4) is non-stationary; the covariance grows with increasing time. Note also that this covariance differs from a standard geometric kernel in the addition of a *process noise* term, dependent on Q. Essentially, the KF kernel assumes that our variable will evolve according to the linear mapping G, and then incorporates the process noise term to allow for the fact that, of course, it will not. Whether this assumption is appropriate is entirely dependent on the system under consideration.

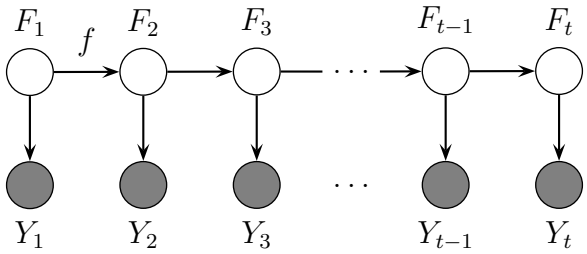


Figure C.1: Markov chain

What is the use of such an unusual covariance? Well, the Markov chain representation is of interest because it gives rise to a sparse precision (inverse covariance) matrix over $\boldsymbol{f}$ and $\boldsymbol{y}$. The precision matrix tells us how pairs of nodes are correlated, conditioned on all other nodes [MacKay, 2006]. Clearly then, under a Markov chain, only neighbours will have non-zero elements in the precision matrix.

This property gives rise to very useful computational savings. We define $y_{1:k} \triangleq (y_1, \ldots, y_k)$ and

$$p(\, f_j \mid y_{1:k},\, I\,) = \mathcal{N}\big(f_j; m_{j|k}, C_{j|k}\big) \tag{C.1.5}$$

Now, we know from our Markov chain and (C.1.4) that

$$p(\, f_{k-1},\, f_k \mid y_{1:k-1},\, I\,) = \mathcal{N}\left(\begin{bmatrix} f_{k-1} \\ f_k \end{bmatrix}; \begin{bmatrix} m_{k-1|k-1} \\ G\, m_{k-1|k-1} \end{bmatrix}, \begin{bmatrix} C_{k-1|k-1} & C_{k-1|k-1}\, G^{\mathsf{T}} \\ G\, C_{k-1|k-1} & G\, C_{k-1|k-1}\, G^{\mathsf{T}} + Q \end{bmatrix}\right) \tag{C.1.6}$$

which gives the *prediction* step of a KF

$$m_{k|k-1} = G\, m_{k-1|k-1} \tag{C.1.7a}$$

$$C_{k|k-1} = G\, C_{k-1|k-1}\, G^{\mathsf{T}} + Q \tag{C.1.7b}$$

Hence the history of our observations $y_{1:k-1}$ affects our inference about f_k only through our predictions about f_{k-1}. As such, we do not need to store our entire observation history, only the most recent set of predictions. We can also quickly update in light of a new observation as:

$$p(\, f_k,\, y_k \mid y_{1:k-1},\, I\,) = \mathcal{N}\left(\begin{bmatrix} f_k \\ y_k \end{bmatrix}; \begin{bmatrix} m_{k|k-1} \\ H\, m_{k|k-1} \end{bmatrix}, \begin{bmatrix} C_{k|k-1} & C_{k|k-1}\, H^{\mathsf{T}} \\ H\, C_{k|k-1} & H\, C_{k|k-1}\, H^{\mathsf{T}} + R \end{bmatrix}\right) \tag{C.1.8}$$

which gives the *fusion* step of a KF

$$m_{k|k} = m_{k|k-1} + C_{k|k-1}\, H^{\mathsf{T}}\, \left(H\, C_{k|k-1}\, H^{\mathsf{T}} + R\right)^{-1}\, \left(y_k - H\, m_{k|k-1}\right) \tag{C.1.9a}$$

$$C_{k|k} = C_{k|k-1} - C_{k|k-1}\, H^{\mathsf{T}}\, \left(H\, C_{k|k-1}\, H^{\mathsf{T}} + R\right)^{-1}\, H\, C_{k|k-1} \tag{C.1.9b}$$

The prediction and fusion steps form the heart of KF-style inference. We sequentially produce predictions for increasing f_k, each dependent only on the previous prediction and the new observation. Note, however, this can be supplemented with the use of the KF covariance (C.1.4) for GP-style inference. In particular, we can use this covariance function to incorporate information about the function at any time to produce predictions about the function at any other time, without requiring tedious

sequential updates. This provides a simple way to manage missing data or far-horizon predictions. This covariance can also be used in a GP over continuous time, in which j and k are permitted to be continuous themselves, providing a generalisation of the discrete-time KF. The covariance (C.1.4) could also be combined with other covariances. For example, you could do inference about a function of both time, over which you have a KF covariance function, and space, for which you have a squared exponential covariance function.

Note that all of the claims made in this section will continue to hold if H, G, Q and R are themselves functions of time index k. We restrict ourselves to the case above simply for the brevity of expressing our functions in (C.1.4).

C.2 Near Constant Acceleration Covariance

A popular form of KF covariance is the so-called *near constant acceleration model* (CAM) [Jazwinski, 1970]. Essentially, this tracks a variable f which is subject to the constraint that its second derivative wrt time $\ddot{f}$ undergoes Brownian motion:

$$p\Big(\ \ddot{f}_1 \ \Big|\ I\ \Big) \triangleq \mathcal{N}\Big(\ddot{f}_1; \nu, \Lambda\Big) \tag{C.2.1a}$$

$$p\Big(\ \ddot{f}_k \ \Big|\ \ddot{f}_{k-1},\ I\ \Big) \triangleq \mathcal{N}\Big(\ddot{f}_k; \ddot{f}_{k-1}, Q\Big) \tag{C.2.1b}$$

From (C.1.3) and (C.1.4), then, we have

$$\mu_{\ddot{f}}(t_k) \triangleq \nu \tag{C.2.2a}$$

$$K_{\ddot{f}}(t_j, t_k) = \Lambda + \min(j,k)\, Q \tag{C.2.2b}$$

Naturally, we are usually more interested in tracking f than $\ddot{f}$, so we now need to determine the mean and covariance over f. As per Section 4.7, we can do exactly this, giving the GP over f as having the mean and covariance

$$\mu_f(t_k) \triangleq \frac{1}{2} t_k^2\, \nu + \sum_{a=0}^{1} \alpha_a\, t_k^a \tag{C.2.3a}$$

$$K_f(t_j, t_k) = \frac{1}{4} t_j^2 t_k^2\, \Lambda + \frac{1}{6} t_j^2 t_k^2\, \min(j,k)\, Q + \sum_{a=0}^{1} \beta_a(t_k)\, t_j^a + \sum_{b=0}^{1} \gamma_b(t_j)\, t_k^b \tag{C.2.3b}$$

That is, if we differentiate the covariance twice by t_j and twice by t_k, we recover the covariance over $\ddot{f}$. The α's are further hyperparameters that may be determined by, for example, initial conditions on f or $\dot{f}$. Similarly, the β's and γ's are functions specified by such hyperparameters.

Appendix D

Hyperparameter Posterior Mean

For a point estimate for our hyperparameters, we can use Bayesian quadrature to estimate the posterior mean for a hyperparameter [Garnett et al., 2010a] ϕ

$$m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{z}_d, I\Big) = \frac{\int \phi\, p\big(\boldsymbol{z}_d|\phi, I\big)p\big(\phi|I\big)\mathrm{d}\phi}{\int p\big(\boldsymbol{z}_d|\phi, I\big)p\big(\phi|I\big)\mathrm{d}\phi}\ . \tag{D.0.1}$$

Once again, we will assume the Gaussian prior (7.1.2)

$$p(\,\phi\mid I\,) \triangleq \mathcal{N}\big(\phi; \nu^{(\phi)}, \Lambda^{(\phi)}\big)$$

for our hyperparameters. Essentially, we take exactly the same approach as in Section 7.4. We again take our GP for $\tilde{r}$, with zero mean and Gaussian covariance (7.1.4)

$$K^{(f)}(\phi_1, \phi_2) \triangleq h^{(f)}\,\mathcal{N}\big(\phi_1; \phi_2, W^{(f)}\big)\,.$$

and use it to perform inference about $\mathring{\varrho}''$

$$\mathring{\varrho}'' \triangleq m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{z}_d, \boldsymbol{r}_{\boldsymbol{:}},\, I\Big) = \frac{\int \phi\, r(\phi)\, p\big(\phi|I\big)\mathrm{d}\phi}{\int r(\phi)\, p\big(\phi|I\big)\mathrm{d}\phi}\ .$$

We have

$$\begin{aligned} m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{r}_s,\, \boldsymbol{z}_d, I\Big) &= \iint m\Big(\mathring{\phi}\,\Big|\,\rho''\Big)\, p(\,\varrho''\mid \tilde{\boldsymbol{r}}_{\boldsymbol{:}},\, I\,)\, p(\,\tilde{\boldsymbol{r}}_{\boldsymbol{:}}\mid \tilde{\boldsymbol{r}}_s,\, I\,)\,\mathrm{d}\varrho''\,\mathrm{d}\tilde{\boldsymbol{r}}_{\boldsymbol{:}} \\ &= \int \varrho''\,\delta(\varrho'' - \mathring{\varrho}'')\,\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{\boldsymbol{:}}; \boldsymbol{m}^{(\tilde{r})}_{\boldsymbol{:}|s}, \mathbf{C}^{(\tilde{r})}_{\boldsymbol{:}|s}\Big)\,\mathrm{d}\varrho''\,\mathrm{d}\tilde{\boldsymbol{r}}_{\boldsymbol{:}} \\ &= \int \mathring{\varrho}''\,\mathcal{N}\Big(\tilde{\boldsymbol{r}}_{\boldsymbol{:}}; \boldsymbol{m}^{(\tilde{r})}_{\boldsymbol{:}|s}, \mathbf{C}^{(\tilde{r})}_{\boldsymbol{:}|s}\Big)\,\mathrm{d}\tilde{\boldsymbol{r}}_{\boldsymbol{:}} \end{aligned}$$

Here we have used

$$m\Big(\mathring{\phi}\,\Big|\,\rho''\Big) \triangleq \int \phi\, p(\,\phi\mid \varrho''\,)\,\mathrm{d}\phi = \varrho''\,.$$

D.1 MAP Approximation For Log-likelihood Function

As before, we might wish to take the MAP approximation for r (7.3.11) to give us

$$m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{r}_s,\,\boldsymbol{z}_d,\,I\Big) \simeq \frac{\bar{\mathfrak{y}}_s^{(r)\mathsf{T}}\,\mathbf{K}^{(r)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1}\,\boldsymbol{r}_s}{\mathfrak{y}_s^{(r)\mathsf{T}}\,\mathbf{K}^{(r)}(\boldsymbol{\phi}_s,\boldsymbol{\phi}_s)^{-1}\,\boldsymbol{r}_s},$$

where $\mathfrak{y}_s^{(r)}$ is of the form expressed in (7.1.7) and we define

$$\begin{aligned}\bar{\mathfrak{y}}_s^{(r)} &\triangleq \Big[\int \phi\, K^{(f)}(\phi,\phi_i)\, p\big(\phi|I\big)\mathrm{d}\phi;\ i\in s\Big]^{\mathsf{T}}\\ &= \big[\mathcal{N}\big(\phi_i;\nu^{(\phi)},\Lambda^{(\phi)}+W^{(\mathrm{f})}\big)\,\big(\nu^{(\phi)}+\Lambda^{(\phi)}(\Lambda^{(\phi)}+W^{(\mathrm{f})})^{-1}(\phi_i-\nu^{(\phi)})\big);\ i\in s\big]^{\mathsf{T}}\,.\end{aligned} \tag{D.1.1}$$

D.2 Quadrature Over Log-likelihood Functions

Alternatively, if we use our sample log-likelihood functions $(\tilde{\boldsymbol{r}}_{:})_l$, and define

$$\begin{aligned}\mathring{\alpha}'' &\triangleq m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{r}_s,\,\boldsymbol{z}_d,\,I\Big)\\ \boldsymbol{\varrho}_l'' &\triangleq \varrho''\big((\tilde{\boldsymbol{r}}_{:})_l\big) \triangleq \Big[m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{r}_{:},\,\boldsymbol{z}_d,\,I\Big)\,;\,\boldsymbol{r}_{:}\in(\tilde{\boldsymbol{r}}_{:})_l\Big]^{\mathsf{T}}\end{aligned}$$

we have

$$m\Big(\mathring{\phi}\,\Big|\,\boldsymbol{r}_s,\,\boldsymbol{z}_d,\,\boldsymbol{\varrho}_l'',\,I\Big) = m(\mathring{\alpha}''\,|\,\boldsymbol{\varrho}_l'',I) = \mathfrak{y}_l^{(\varrho'')\mathsf{T}}\mathbf{K}^{(\varrho'')}\big((\tilde{\boldsymbol{r}}_{:})_l,(\tilde{\boldsymbol{r}}_{:})_l\big)^{-1}\boldsymbol{\varrho}_l'' \tag{D.2.1}$$

where $\mathfrak{y}_l^{(\varrho'')\mathsf{T}}$ is of the form (7.1.7) with $\nu^{(\tilde{\boldsymbol{r}}_{:})} = \boldsymbol{m}_{:|s}^{(\tilde{r})}$ and $\Lambda^{(\tilde{\boldsymbol{r}}_{:})} = \mathbf{C}_{:|s}^{(\tilde{r})}$. (D.2.1) represents our 'best' estimate for (D.0.1).

If our sample likelihood functions are again of the form of a conditional GP mean (7.3.15), we have, for $i\in l$

$$\varrho_i'' = \varrho''\big((\tilde{r}_{:})_i\big) =\simeq \frac{\bar{\mathfrak{y}}_{[s,c]}^{(r)\mathsf{T}}\,\mathbf{K}^{(r)}(\boldsymbol{\phi}_{[s,c]},\boldsymbol{\phi}_{[s,c]})^{-1}\,\boldsymbol{r}_{[s,c_i]}}{\mathfrak{y}_{[s,c]}^{(r)\mathsf{T}}\,\mathbf{K}^{(r)}(\boldsymbol{\phi}_{[s,c]},\boldsymbol{\phi}_{[s,c]})^{-1}\,\boldsymbol{r}_{[s,c_i]}},$$

Appendix E

Bayesian Quadrature With Other Hyperparameter Priors

We now present results [Garnett et al., 2010a] corresponding to those in Section 7.3, Section 7.4 and Appendix D for non-Gaussian priors $p(\,\phi \mid I\,)$. We will continue to use Gaussian covariances of the form

$$K^{(f)}(\phi_1, \phi_2) \triangleq h^{(f)}\,\mathcal{N}\big(\phi_1; \phi_2, W^{(f)}\big)\ .$$

E.1 Bounded Uniform Prior

For a one-dimensional hyperparameter ϕ known only to lie between two bounds l and u, we take the uniform distribution over that region,

$$p(\,\phi \mid I\,) = \frac{\sqcap(\phi; l, u)}{\Delta}\,,$$

where $\Delta \triangleq u - l$ and $\sqcap(\theta; l, u)$ is used to denote the rectangular function

$$\sqcap(\phi; l, u) \triangleq \begin{cases} 1 & (l < \phi < u) \\ 0 & (\text{otherwise}) \end{cases}\ .$$

This requires us to replace (7.3.13) with

$$\mathfrak{Y}^{(f,g)}_{u_{\text{left}}, v_{\text{right}}}(i, j) = \frac{\big(h^{(f)} h^{(g)}\big)^2}{\Delta} \mathcal{N}\big(\phi_i; \phi_j, W^{(f)} + W^{(g)}\big) \Big(\Phi(u; m, C) - \Phi(l; m, C)\Big)\,,$$

for $(i,j) \in (v_{\text{left}} \times v_{\text{right}})$ and where

$$\begin{aligned} m &\triangleq W^{(f)}(W^{(f)}+W^{(g)})^{-1}\phi_i + W^{(g)}(W^{(f)}+W^{(g)})^{-1}\phi_j \\ &= \frac{W^{(f)}\phi_i + W^{(g)}\phi_j}{W^{(f)}+W^{(g)}} \\ C &\triangleq W^{(f)} - W^{(f)}(W^{(f)}+W^{(g)})^{-1}W^{(f)} \\ &= W^{(g)} - W^{(g)}(W^{(f)}+W^{(g)})^{-1}W^{(g)} \\ &= \frac{W^{(g)}W^{(f)}}{W^{(f)}+W^{(g)}}\,. \end{aligned}$$

Similarly, we replace (7.1.7) with

$$\mathfrak{y}_v^{(f)}(i) = \frac{h^{(f)}}{\Delta}\Big(\Phi\big(u;\phi_i,W^{(f)}\big) - \Phi\big(l;\phi_i,W^{(f)}\big)\Big)$$

for $i \in v$.

Finally, we replace (D.1.1) with

$$\bar{\mathfrak{y}}_v^{(r)}(i) = \frac{h^{(f)}}{\Delta}\Big(\phi_i\Big(\Phi\big(u;\phi_i,w^2\big) - \Phi\big(l;\phi_i,w^2\big)\Big) - w^2\Big(\mathcal{N}\big(u;\phi_i,w^2\big) - \mathcal{N}\big(l;\phi_i,w^2\big)\Big)\Big)$$

for $i \in v$.

E.2 Discrete Prior

Occasionally we may also want to consider a discrete hyperparameter ϕ. In this case we will take the uniform prior

$$P(\,\phi \mid I\,) = \frac{1}{\Delta}\,,$$

where ϕ takes values from the discrete set $\{\phi_1, \ldots, \phi_\Delta\}$. This requires us to replace (7.3.13) with

$$\mathfrak{Y}^{(f,g)}_{v_{\text{left}},v_{\text{right}}}(i,j) = \frac{1}{\Delta}\sum_{d=1}^{\Delta} K^{(f)}(\phi_i,\phi_d)\,K^{(g)}(\phi_d,\phi_j)\,,$$

for $(i,j) \in (v_{\text{left}} \times v_{\text{right}})$. Similarly, we replace (7.1.7) with

$$\mathfrak{y}_v^{(f)}(i) = \frac{1}{\Delta}\sum_{d=1}^{\Delta} K^{(f)}(\phi_i,\phi_d)$$

for $i \in v$.

Finally, we replace (D.1.1) with

$$\bar{\mathfrak{y}}_v^{(r)}(i) = \frac{1}{\Delta} \sum_{d=1}^{\Delta} \phi_d \, K^{(f)}(\phi_i, \phi_d)$$

for $i \in v$.

Bibliography

P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway, 1997. URL `http://www.math.ntnu.no/~omre/TMA4250/V2007/abrahamsen2.ps`. 2nd edition.

R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. Technical report, University of Cambridge, Cambridge, UK, 2007. arXiv:0710.3742v1 [stat.ML].

D. M. Appleby. Probabilities are single-case, or nothing. *Optics and Spectroscopy*, 99:447, 2005. URL `http://arXiv.org/quant-ph/0408058`.

M. Basseville and I. Nikiforov. *Detection of abrupt changes: theory and application*. Prentice Hall, 1993.

J. Berger. The case for objective Bayesian analysis. *Bayesian Analysis*, 1(3):385–402, 2006.

C. Bishop et al. *Pattern recognition and machine learning*. Springer New York:, 2006.

P. Boyle and M. Frean. Dependent Gaussian processes. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 217–224. MIT Press, Cambridge, MA, 2005.

G. L. Bretthorst. The near-irrelevance of sampling frequency distributions. In W. von der Linden et al., editor, *Maximum Entropy and Bayesian Methods*, pages 21–46. Kluwer Academic Publishers, the Netherlands, 1999.

British Atmospheric Data Centre. UK Meteorological Office MIDAS Land Surface Stations data (1853-current). Available from: `http://badc.nerc.ac.uk/data/ukmo-midas`, 2009.

B. Brodsky and B. Darkhovsky. *Nonparametric Methods in Change-Point Problems*. Springer, 1993.

W. L. Buntine. Operations for learning with graphical models. *Journal of Artifical Intelligence Research*, 2:159, 1994. URL `http://arXiv.org/cs/9412102`.

B. P. Carlin, A. E. Gelfand, and A. F. M. Smith. Hierarchical Bayesian analysis of changepoint problems. *Applied statistics*, 41(2):389–405, 1992.

R. Carter, J. Gablonsky, A. Patrick, C. Kelley, and O. Eslinger. Algorithms for Noisy Problems in Gas Transmission Pipeline Optimization. *Optimization and Engineering*, 2 (2):139–157, 2001.

J. Chen and A. Gupta. *Parametric Statistical Change Point Analysis*. Birkháuser Verlag, 2000.

H. Chernoff and S. Zacks. Estimating the Current Mean of a Normally Distributed Variable Which is Subject to Changes in Time. *Annals of Mathematical Statistics*, 35:999–1028, 1964.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6(1):1019, 2006.

R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946. URL `http://link.aip.org/link/?AJP/14/1/1`.

N. A. C. Cressie. *Statistics for spatial data*. John Wiley & Sons, 1991.

M. Csorgo and L. Horvath. *Limit theorems in change-point analysis*. John Wiley & Sons, 1997.

A. P. Dawid, M. Stone, and J. V. Zidek. Critique of E. T. Jaynes's 'Paradoxes of probability theory'. Technical report, Department of Statistics, University College London, September 1996.

B. de Finetti. La prévision: Ses lois logiques, ses sources subjectives. In *Annales de l'Institut Henri Poincaré 7*, pages 1–68. Paris, 1937. Translated into English by Henry E. Kyburg Jr., Foresight: Its Logical Laws, its Subjective Sources. In Henry E. Kyburg Jr. and Howard E. Smokler (1964, Eds.), Studies in Subjective Probability, 53-118, Wiley, New York.

B. de Finetti. Probabilities of probabilities: a real problem or a misunderstanding? In B. C. Aykac A, editor, *New developments in the application of Bayesian methods*, pages 1–10, 1977.

D. C. Dennett. *Consciousness Explained*. Little, Brown and Company, Boston, New York, 1991.

A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004)*, pages 588–599, 2004.

S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, November 2000. ISBN 0471056693.

E. Ertin. Gaussian Process Models for Censored Sensor Readings. In *Statistical Signal Processing, 2007. SSP'07. IEEE/SP 14th Workshop on*, pages 665–669, 2007.

S. Faul, G. Gregorcic, G. Boylan, W. Marnane, G. Lightbody, and S. Connolly. Gaussian Process Modeling of EEG for the Detection of Neonatal Seizures. *IEEE Transactions on Biomedical Engineering*, 54(12):2151–2162, 2007.

P. Fearnhead and Z. Liu. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605, 2007.

Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 108–114. ACM New York, NY, USA, 2003.

R. Garnett, M. A. Osborne, and S. Roberts. Sequential Bayesian prediction in the presence of changepoints. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 2010a. doi: 10.1093/comjnl/bxq003.

R. Garnett, M. A. Osborne, and S. Roberts. Bayesian optimization for sensor set selection. In *International Conference on Information Processing in Sensor Networks (IPSN 2010)*, 2010b. (To appear).

M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Softw.*, 29(4):469–480, 2003.

A. Genz. Numerical Computation of Multivariate Normal Probabilities. *Journal of Computational and Graphical Statistics*, 1(2):141–149, 1992.

M. N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, Cambridge University, 1997. URL `http://www.inference.phy.cam.ac.uk/mng10/GP/thesis.ps`.

A. Girard, C. Rasmussen, J. Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2003.

T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

R. Goldsmith and N.-E. Sahlin. The role of second-order probabilities in decision making. In P. Humphreys, O. Svenson, and A. Vari, editors, *Analysing and Aiding Decision Processes*, North-Holland, Amsterdam, 1983.

P. Gregory. *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica Support*. Cambridge University Press, April 2005.

H. M. Gutmann. A Radial Basis Function Method for Global Optimization. *Journal of Global Optimization*, 19(3):201–227, 2001.

J. K. Hart and K. Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, 2006.

D. Heath and W. Sudderth. De Finetti's Theorem on exchangeable variables. *The American Statistician*, 30(4):188–189, November 1976.

D. Heckerman. A tutorial on learning with Bayesian Networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

K. Holmström. New Optimization Algorithms and Software. *Theory of Stochastic Processes*, 1(2):55–63, 1999.

L. Horváth and P. Kokoszka. The effect of long-range dependence on change-point estimators. *Journal of Statistical Planning and Inference*, 64(1):57–81, 1997.

D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. *Journal of Global Optimization*, 34 (3):441–466, 2006.

W. Huyer and A. Neumaier. SNOBFIT–Stable Noisy Optimization by Branch and Fit. *ACM Transactions on Mathematical Software*, 35, 2008.

E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, April 2003.

A. Jazwinski. *Stochastic processes and filtering theory*. Academic Press New York, 1970.

H. Jeffreys. *Theory of Probability*. Oxford University Press, third edition, November 1998.

H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007): 453–461, September 1946.

F. V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

S. Ji, B. Krishnapuram, and L. Carin. Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):522–532, 2006.

D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

R. Jürgens, W. Becker, and H. Kornhuber. Natural and drug-induced variations of velocity and duration of human saccadic eye movements: evidence for a control of the neural pulse generator by local feedback. *Biological Cybernetics*, 39(2):87–96, 1981.

A. Kapoor and E. Horvitz. On discarding, caching, and recalling samples in active learning. In *Uncertainty in Artificial Intelligence*, 2007.

R. Kass and L. Wasserman. The selection of prior distributions by formal rules. *Journal of the American Statistical Association*, 91:1343 – 1370, September 1996.

C. Kelley. Users Guide for imfil Version 0.7, 2008. See `http://www4.ncsu.edu/~ctk/MATLAB_IMFIL07/manual.pdf`.

K. H. Knuth. Lattice duality: The origin of probability and entropy. *Neurocomputing*, 67: 245–274, 2005.

R. Kondor and T. Jebara. A kernel between sets of vectors. In *Proceedings of the International Conference on Machine Learning*, volume 20, page 361, 2003.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, July 1996.

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems*, 15: 609–616, 2003.

S. Leary, A. Bhaskar, and A. Keane. A Derivative Based Surrogate Model for Approximating and Optimizing the Output of an Expensive Computer Simulation. *Journal of Global Optimization*, 30(1):39–58, 2004.

H. Lee. Variational Bayesian hidden Markov models with mixtures of Gaussian emission. Technical report, University of Oxford, Oxford, UK, 2009. `http://www.robots.ox.ac.uk/~mosb/Lee2009.pdf`.

S. M. Lee and S. J. Roberts. Multivariate time series forecasting in incomplete environments. Technical Report PARG-08-03. Available at `www.robots.ox.ac.uk/~parg/publications.html`, University of Oxford, December 2008.

E. Levina and P. Bickel. The earth movers distance is the Mallows distance: Some insights from statistics. In *Proc. ICCV*, volume 2, pages 251–256. Citeseer, 2001.

D. Lizotte. *Practical Bayesian optimization*. PhD thesis, University of Alberta, 2008.

T. J. Loredo. Computational technology for Bayesian inference. In D. M. Mehringer, R. L. Plante, and D. A. Roberts, editors, *ASP Conference Series 172: Astronomical Data Analysis Software and Systems*, volume 8, pages 297–, San Francisco, 1999.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

D. J. MacKay. The humble Gaussian distribution, June 2006. URL `http://www.inference.phy.cam.ac.uk/mackay/humble.ps.gz`.

D. J. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 84–92. Springer-Verlag, 1998. URL `ftp://www.inference.phy.cam.ac.uk/pub/mackay/gpB.ps.gz`.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, June 2002. ISBN 0521642981.

T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

T. P. Minka. Deriving quadrature rules from Gaussian processes. Technical report, Statistics Department, Carnegie Mellon University, 2000.

M. Molga and C. Smutnicki. Test functions for optimization needs, 1995. See `www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf`.

H. Muller. Change-points in nonparametric regression analysis. *Ann. Statist*, 20(2):737–761, 1992.

R. Murray-Smith and B. Pearlmutter. Transformations of Gaussian process priors. *Lecture notes in computer science*, 3635:110, 2005.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993. URL `http://www.cs.toronto.edu/~R./ftp/review.pdf`.

R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, Dept. of Statistics, University of Toronto, 1997. URL `http://www.cs.toronto.edu/~R./ftp/mc-gp.pdf`.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, July 2006.

A. O'Hagan. Bayes-hermite quadrature. *Journal of Statistical Planning and Inference*, 29: 245–260, 1991.

A. O'Hagan. Some Bayesian numerical analysis. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 4*, pages 345–363. Oxford University Press, 1992.

A. O'Hagan. Monte Carlo is fundamentally unsound. *The Statistician*, 36:247–249, 1987.

R. Olea. Sampling design optimization for spatial functions. *Mathematical Geology*, 16(4): 369–392, 1984.

M. A. Osborne, A. Rogers, S. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 109–120, April 2008. URL `http://eprints.ecs.soton.ac.uk/15122/`.

M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimisation. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009. Available at `http://lion.disi.unitn.it/intelligent-optimization/LION3/online_proceedings/94.pdf`.

M. A. Osborne, R. Garnett, and S. Roberts. Active data selection for sensor networks with faults and changepoints. In *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia*. ACM New York, NY, USA, 2010a. (To appear).

M. A. Osborne, A. Rogers, S. J. Roberts, S. D. Ramchurn, and N. Jennings. Bayesian Gaussian process models for multi-sensor time-series prediction. In *Inference and Learning in Dynamic Models*. Cambridge University Press, 2010b. Available at `http://www.robots.ox.ac.uk/~mosb/osborne_final.pdf`.

J. Pearl. *Causality : Models, Reasoning, and Inference*. Cambridge University Press, March 2000. ISBN 0521773628.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988. ISBN 1558604790.

J. Pinheiro and D. Bates. Unconstrained parameterizations for variance-covariance matrices. *Statistics and Computing*, 6:289–296, 1996.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0521437148.

J. Quiñonero-Candela and C. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1959, 2005.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, Cambridge, MA, 2003.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

B. Ray and R. Tsay. Bayesian methods for change-point detection in long-range dependent processes. *Journal of Time Series Analysis*, 23(6):687–705, 2002.

S. Reece, R. Garnett, M. A. Osborne, and S. J. Roberts. Anomaly detection and removal using non-stationary Gaussian processes. Technical report, University of Oxford, Oxford, UK, 2009. Available at `http://www.robots.ox.ac.uk/~reece/reece2009.pdf`.

S. J. Roberts. Extreme value statistics for novelty detection in biomedical data processing. In *Science, Measurement and Technology, IEE Proceedings-*, volume 147, pages 363–367, 2000.

S. J. Roberts, R. Everson, I. Rezek, P. Anderer, and A. Schlögl. Tracking ICA for eye-movement artefact removal. In *Proc. EMBEC'99, Vienna*, 1999.

A. Rogers, M. Osborne, S. Ramchurn, S. Roberts, and N. Jennings. Information Agents for Pervasive Sensor Networks. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 294–299. IEEE Computer Society Washington, DC, USA, 2008.

O. Ruanaidh, W. Fitzgerald, and K. Pope. Recursive Bayesian location of a discontinuity in time series. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1994. on IEEE International Conference-Volume 04*, pages 513–516. IEEE Computer Society, 1994.

Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, second edition, 2003. ISBN 0137903952.

M. J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

A. Schmidt and A. O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 65(3):743–758, 2003.

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: active data selection and test pointrejection. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, 2000.

J. Skilling. Nested sampling for Bayesian computations. In *Proceedings of the Valencia / ISBA 8th World Meeting on Bayesian Statistics*, Benidorm (Alicante, Spain), June 2006.

M. L. Stein. *Interpolation of Spatial Data*. Springer Verlag, New York, 1999.

Y. Teh, M. Seeger, and M. Jordan. Semiparametric latent factor models. *Workshop on Artificial Intelligence and Statistics*, 10, 2005.

The MathWorks. MATLAB R2007a, 2007. Natick, MA.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems 18*, pages 1441–1448. The MIT Press, 2006. Proc. NIPS'05.

B. Whitcher, S. Byers, P. Guttorp, and D. Percival. Testing for homogeneity of variance in time series: Long memory, wavelets and the Nile River. *Water Resources Research*, 38 (5):10–1029, 2002.

Z. Zhu and M. Stein. Spatial sampling design for prediction with estimated parameters. *Journal of Agricultural, Biological, and Environmental Statistics*, 11(1):24–44, 2006.