Inference on Markov Random Fields: Methods and Applications

Thibaut Lienart

University College, University of Oxford

A thesis submitted for the degree of Doctor of Philosophy

Michaelmas 2017

Abstract

This thesis considers the problem of performing inference on undirected graphical models with continuous state spaces. These models represent conditional independence structures that can appear in the context of Bayesian Machine Learning. In the thesis, we focus on computational methods and applications. The aim of the thesis is to demonstrate that the factorisation structure corresponding to the conditional independence structure present in high-dimensional models can be exploited to decrease the computational complexity of inference algorithms. First, we consider the smoothing problem on Hidden Markov Models (HMMs) and discuss novel algorithms that have sub-quadratic computational complexity in the number of particles used. We show they perform on par with existing state-of-the-art algorithms with a quadratic complexity. Further, a novel class of rejection free samplers for graphical models known as the Local Bouncy Particle Sampler (LBPS) is explored and applied on a very large instance of the Probabilistic Matrix Factorisation (PMF) problem. We show the method performs slightly better than Hamiltonian Monte Carlo methods (HMC). It is also the first such practical application of the method to a statistical model with hundreds of thousands of dimensions. In a second part of the thesis, we consider approximate Bayesian inference methods and in particular the Expectation Propagation (EP) algorithm. We show it can be applied as the backbone of a novel distributed Bayesian inference mechanism. Further, we discuss novel variants of the EP algorithms and show that a specific type of update mechanism, analogous to the mirror descent algorithm outperforms all existing variants and is robust to Monte Carlo noise. Lastly, we show that EP can be used to help the Particle Belief Propagation (PBP) algorithm in order to form cheap and adaptive proposals and significantly outperform classical PBP.

Inference on Markov Random Fields: Methods and Applications

Thibaut Lienart

Department of Statistics

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Supervisors:	Prof. Arnaud Doucet	(University of Oxford)
	Prof. Yee Whye Teh	(University of Oxford)
Examiners:	Prof. Francois Caron	(University of Oxford)
	Prof. Richard Turner	(University of Cambridge)

University College

Michaelmas 2017

Contents

Acknowledgments

I am indebted to both my supervisors Prof. Arnaud Doucet and Prof. Yee Whye Teh for their patience and their support. Working with two leading professors in Bayesian computational methods was sometimes intimidating, often challenging and always inspiring. I am also grateful to the Scatcherd European scholarship fund, EPSRC, the department of Statistics and the Alan Turing Institute for their generous financial support throughout my studies.

During my DPhil., I was fortunate to meet, work, and train with amazing people, all of whom played a role in helping me bring this thesis to fruition while staying sane: members of University College, PhD students from the department of Statistics and the Alan Turing Institute, OULRC and UCBC rowers, OUCC cyclists, OUKBC boxers, my colleagues at Cambridge Spark and, more recently, the Computational Privacy Group at the Imperial College.

I am particularly indebted to four of my closest friends: Adrien, Nikola, Armin and Alexander for sharing the pains and excitements of pursuing a DPhil. in a mathematical topic and for their unfaltering support. Many more friends deserve to be mentioned: Petra, Hannah, Julie, Matt, Geoffroy, Francois-Xavier, David, Ali…the list goes on and all, in their own way, contributed to this thesis.

Finally, I would like to thank my family and my girlfriend Camille; although the topic of this thesis will likely remain somewhat of a mystery to them, they were always there to listen to my worries, provide encouragements and help me see the light at the end of the tunnel.

Cette thèse est dédiée à ma nièce, Léa.

Notations

We list here notations used throughout the document that are considered to be commonly used in computational statistics. Non-standard notations will be introduced in the document explicitly.

Vectors, matrices and functions

For a vector $x \in \mathbb{R}^d$ we write $\|x\|_p$ the p-norm of x with $\|x\|_p^p = \sum_{i=1:d} |x_i|^p$ for $p \in [1, \infty)$. For $x, y \in \mathbb{R}^d$, we write $\langle x, y \rangle = \sum_{i=1:d} x_i y_i$ the inner product of x and y. For a nonnegative vector x_u we write $x_u \propto x$ if there is a normalisation constant Z such that $x = Z^{-1} x_u$ has components summing to one. We write z^+ to denote $\max\{0, z\}$. We denote the transpose of a matrix $A \in \mathbb{R}^{d \times d}$ by A^t. A is said to be semi positive definite if $\langle x, Ax \rangle \geq 0$ for all $x \in \mathbb{R}^d$ and positive definite if the inequality holds strictly for all $x \neq 0$. We denote by $\mathbb{S}_+^d \subset \mathbb{R}^{d \times d}$ the set of symmetric positive definite matrices and by $\mathbb{S}_-^d$ the set of matrices B such that $-B \in \mathbb{S}_+^d$. For a real-valued measurable function f on a set $\mathcal{X} \subseteq \mathbb{R}^d$, we write $\int f(x) \mathrm{d}x$ to denote the integral of f on the whole of $\mathcal{X}$. We write $\|f\|_p = (\int |f(x)|^p \, \mathrm{d}x)^{1/p}$ the p-norm of f for $p \in [1, \infty)$. We write $L^1(\mathcal{X})$ the set of functions such that $\|f\|_1 < \infty$ and $\mathcal{P}(\mathcal{X}) \subset L^1(\mathcal{X})$ denotes the restriction to nonnegative functions integrating to one (probability density functions). For a nonnegative function $g_u \in L^1(\mathcal{X})$ we write $g_u \propto g$ to indicate that there is a normalisation constant Z such that $g = Z^{-1} g_u$ with $g \in \mathcal{P}(\mathcal{X})$. For a real-valued, differentiable function f on $\mathcal{X}$, we write ∇f its gradient and $\nabla^2 f$ its Hessian provided f is twice differentiable.

Probabilities and expected values

We consider continuous sample spaces $\mathcal{X} \subseteq \mathbb{R}^d$ and, by default, we consider the associated Borel σ-algebra. Correspondingly, we use the shorthand $(\mathcal{X}, \mathbb{P})$ to denote a probability space with $\mathbb{P}$ the probability measure. We will solely consider probability measures that admit a probability density function p with respect to a base measure ν with therefore $\mathbb{P}(x \in A) = \int_A p(x) \nu(\mathrm{d}x)$ for any open subset $A \subset \mathcal{X}$. We write $X \sim p$

to denote a random variable associated to p and $\{X^{(i)}\}_{i=1}^N \sim_{\text{iid}} p$ for a set of independent such random variables. In particular, $\mathcal{N}(\,\cdot\,;\mu,\Sigma)$ denotes the multivariate normal distribution with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{S}_+^d$, $\delta_a(\cdot)$ denotes the Dirac-delta distribution with point-mass at a, $\mathcal{B}(\rho)$ is the Bernoulli with success-rate ρ and $\mathcal{M}(\alpha_1,\ldots,\alpha_d)$ is the multinomial distribution over d cases where α_i denotes the success-rate of case i. For such a probability space $(\mathcal{X},\mathbb{P})$ and a vector-valued mapping $\varphi : \mathcal{X} \to \mathbb{R}^d$, we write the expectation and the covariance matrix of φ with respect to p as $(\mathbb{E}_p[\varphi(X)])_i = \int_{\mathcal{X}} \varphi_i(x)p(x)\nu(\mathrm{d}x)$ and $\mathbb{V}_p[\varphi(X)] = \mathbb{E}_p[\varphi(X)\varphi(X)^t] - \mathbb{E}_p[\varphi(X)]\mathbb{E}_p[\varphi(X)]^t$ respectively. We write $\mathrm{KL}(p,q) = \mathbb{E}_p[\log p(X)] - \mathbb{E}_p[\log q(X))]$ the Kullback-Leibler divergence between two probability distribution functions p and q on $\mathcal{X}$.

Convex analysis

Let $B \subset \mathbb{R}^n$ denote the Euclidean unit ball with $B = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$. For any subset $C \subseteq \mathbb{R}^n$, we write $\mathrm{int}\,(C)$ the interior of C with $\mathrm{int}\,(C) = \{x \in C \mid \exists \epsilon > 0, x + \epsilon B \subset C\}$. Correspondingly we write $\mathrm{cl}(C)$ the closure of C with $\mathrm{cl}(C) = \cap\{C + \epsilon B \,|\, \epsilon > 0\}$. A set C is convex if for any $x, y \in C$, $(1-\lambda)x + \lambda y \in C$ for all $\lambda \in (0,1)$. Let Ω be an arbitrary connected and nonempty subset of $\mathbb{R}^n$; a function $f : \Omega \to \mathbb{R}$ is said to be convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$. For an arbitrary function $f : \Omega \to \mathbb{R}$, we denote by $f^\star : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ the convex-conjugate of f with $f^\star(y) = \sup_{y\in\Omega}[\langle x, y\rangle - f(x)]$. The set of minimisers of a function f on a open, nonempty set C is denoted by $\arg\min_{x\in C} f(x)$ i.e. the set $\{x \in C \mid f(y) \geq f(x),\, \forall y \in C\}$. For a differentiable and strictly convex function $\varphi : C \to \mathbb{R}$, we denote by $B_\varphi(x,y)$ the Bregman divergence associated to φ between x and y in C with $B_\varphi(x,y) = \varphi(x) - \varphi(y) - \langle x - y, \nabla\varphi(y)\rangle$.

Miscellaneous

We denote an undirected graph by a set of vertex labels $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For a node $u \in \mathcal{V}$ we write $\partial u = \{v \in \mathcal{V} \,|\, (u,v) \in \mathcal{E}\}$ the neighbourhood of node u.

We denote the complexity of an algorithm by $\mathcal{O}(h(N))$ to indicate that it scales in N like $h(N)$ for large N. If the algorithm is $\mathcal{O}(N)$ (resp. $\mathcal{O}(N^2)$) we say the algorithm has linear (resp. quadratic) complexity in N. If the complexity is between linear and quadratic, e.g.: $\mathcal{O}(N \log N)$, we say the algorithm is sub-quadratic. We use the same notations to describe the convergence of algorithms.

Abbreviations

We list below abbreviations that are used in this document by alphabetical order and link to where they are first used or introduced in the document.

BIF	Backward Information Filter	(section 3.1)
BP/LBP	(Loopy) Belief Propagation	(section 5.4)
BPS/LBPS	(Local) Bouncy Particle Sampler	(section 2.3)

EP	Expectation Propagation	(section 5.3)
EPBP	Expectation Particle Belief Propagation	(section 7.2)
ESS	Effective Sample Size	(section 2.1)
FFBS	Forward Filtering Backward Smoothing	(section 2.1)
IPP	Inhomogeneous Poisson Process	(section 2.3)
KL	Kullback-Leibler divergence	(section 5.1)
GMMC/LMMC	Local/Global Moment Matching Conditions	(section 5.3)
HMM(s)	Hidden Markov Model(s)	(section 1.2)
MCMC	Markov Chain Monte Carlo	(section 2.1)
MRF(s)	Markov Random Field(s)	(section 1.1)
NBP	Nonparametric Belief Propagation	(section 7.1)
PBP	Particle Belief Propagation	(section 7.1)
PD	Predictive Density	(section 1.2)
PDMP	Piecewise Deterministic Markov Process	(section 2.3)
(I)PP	(Inhomogeneous) Poisson Process	(section 2.3)
RMS(E)	Root Mean Squared (Error)	(section 3.3)
SMC	Sequential Monte Carlo	(section 2.1)
SMS	Sampling via Moment Sharing	(section 6.2)
SNEP	Stochastic Natural gradient EP	(section 6.2)
TFS	Two Filter Smoothing	(section 3.1)

List of publications

A large part of this thesis hinges on the following papers:

1. T. Lienart, Y. W. Teh, and A. Doucet. *Expectation Particle Belief Propagation*. NIPS, 2015.

2. L. Hasenclever, S. Webb, T. Lienart, S. Vollmer, B. Lakshminarayanan, C. Blundell, and Y. W. Teh. *Distributed Bayesian Learning with Stochastic Natural Gradient Expectation Propagation and the Posterior Server*. JMLR, (18):1–37, 2017.

3. J. Bierkens, A. Bouchard-Côté, A. Doucet, A. B. Duncan, P. Fearnhead, T. Lienart, G. Roberts and S. J. Vollmer. *Piecewise Deterministic Markov Processes for Scalable Monte Carlo on Restricted Domains*. Stats. & Prob. Let., 2017.

1 | Introduction

In this thesis, we consider the framework of Bayesian inference in Machine Learning where one is interested in combining prior knowledge about the parameters of a given model with the likelihood of the observed data given the model. This combination of prior and likelihood leads to a posterior distribution on the parameters which is the key object of interest in Bayesian inference. This can be written

$$p(\theta \mid \mathcal{D}) \quad \propto \quad p(\mathcal{D} \mid \theta) p(\theta) \tag{1.1}$$

where $p(\theta)$ is the prior distribution function over the parameter of interest θ, $p(\mathcal{D} \mid \theta)$ denotes the likelihood of the data $\mathcal{D}$ for a given model parameter and $p(\theta \mid \mathcal{D})$ is the posterior distribution over the parameters.

We focus on the problem of estimating expected values with respect to such a posterior distribution and in particular with respect to its marginals. Further, we consider the case where the posterior factorises in a specific way and consider methods that attempt to leverage the factorisation structure to offer estimators at a reduced computational cost. The thesis concentrates on computational methods to tackle the problem in a number of important cases as well as applications and experiments.

The thesis is divided in two parts. In the first part – *Sampling Methods* – we consider "exact" methods that attempt to produce samples from the true marginals and lead to estimators of expectations of interest that are exact in the limit of infinite computational power and infinite precision in the representation of numbers. These methods can work well when the dimensionality of the state-space associated with the marginals of interest is not too high.

In the second part – *Approximate Bayesian Inference* – we consider approximate methods that attempt to find proxies for the marginals in restricted probability distribution spaces. These methods can offer a viable alternative to sampling methods when the latter become too expensive to consider.[1]

In this introductory chapter we introduce the concept of Markov Random Fields and present a few classical examples.

1.1 Markov Random Fields

A *Markov random field* (MRF) is a graph structure that represents joint distributions functions that factorise in a specific way or, in other words, to represent a set of conditional dependences between a set of random variables. In this document, we consider graphs determined by a finite index set $\mathcal{V}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ connecting those nodes and *potential functions* on its cliques (fully connected components of the graph). Each node $i \in \mathcal{V}$ corresponds to one random variable taking values in a sample space $\mathcal{X}_i \subseteq \mathbb{R}^{d_i}$ for some dimension $d_i \in \mathbb{N}$.

We restrict ourselves to graphs with pairwise interactions i.e., uniquely determined

[1] The separation between "exact" and "approximate" methods is a bit arbitrary. Another perspective can be to distinguish between "sampling" versus "distributional" approaches.

by potential functions on their nodes and edges.[2] We write

$$\psi_i : \mathcal{X}_i \to \mathbb{R}^+, \quad \text{and} \quad \psi_{ij} : \mathcal{X}_i \times \mathcal{X}_j \to \mathbb{R}^+,$$

respectively a potential function on a node $i \in \mathcal{V}$ and on an edge $(i,j) \in \mathcal{E}$. Such a graph represents the factorisation structure of a class of probability density functions on the product space $\mathcal{X} := \prod_{i\in\mathcal{V}} \mathcal{X}_i$ that can be written as

$$p(x) \quad \propto \quad \prod_{i\in\mathcal{V}} \psi_i(x_i) \prod_{j\in\partial i} \psi_{ij}(x_i, x_j), \tag{1.2}$$

where $x = (x_i)_{i\in\mathcal{V}}$ and $\partial i := \{j \,|\, (i,j) \in \mathcal{E}\}$ denotes the *neighbourhood* of the ith node. A simple pairwise MRF is illustrated in figure 1.1.

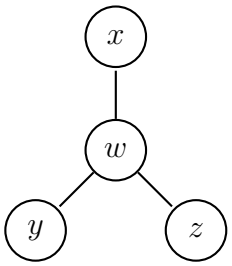


Figure 1.1: *Illustration of a simple MRF corresponding to distributions over 4 random variables admitting the following factorisation structure:* $p(w,x,y,z) \propto \psi_w(w)\psi_x(x)\psi_y(y)\psi_z(z)\psi_{wx}(w,x)\psi_{wz}(w,z)\psi_{wy}(w,y)$.

In this work, we are mainly concerned in determining or approximating *marginals* on the MRF i.e. the distributions $p_{\mathcal{I}}(x_{\mathcal{I}})$ for $\mathcal{I} \subseteq \mathcal{V}$ with

$$p_{\mathcal{I}}(x_{\mathcal{I}}) \quad := \quad \int p(x)\, \mathrm{d}x_{\mathcal{V}\setminus\mathcal{I}}, \tag{1.3}$$

where the integral is implicitly taken over $\prod_{i\notin I} \mathcal{X}_i$.

[2] Considering only pairwise MRF is not too constraining since any MRF can be expressed as a pairwise MRF by the introduction of auxiliary variables although this may lead to a very complex graph (Weiss and Freeman, 2000; Wainwright and Jordan, 2008).

A particular case of interest is $\mathcal{I} = \{r\}$ corresponding to the case of singleton marginals (Wainwright and Jordan, 2008, section 2.3). The integrals of the form (1.3) are typically intractable but exploiting the underlying factorisation structure can lead to good approximation methods.

We distinguish between two classes of undirected graph structures: the connected acyclic ones (*trees*) and the rest (*loopy graphs*). As we will show, computing marginals on a tree is relatively simple in comparison to doing so on loopy graphs. Among the trees, two graph structures are of particular interest in this document: the *chain graph* and the *star graph*. We also illustrate one specific type of loopy graph: the *grid graph*.

1.2 Examples of MRFs

In this section, we present a brief overview of three examples of graph structures on which we will focus most of our effort throughout this document. We also present the type of applications they are connected with.

1.2.1 Hidden Markov Models

Chain graphs, where the random variables on every node take value in the same state-space $\mathcal{X}$, form the underlying structure of *Hidden Markov Models* (HMM) an important class of models. These can be illustrated as follows:

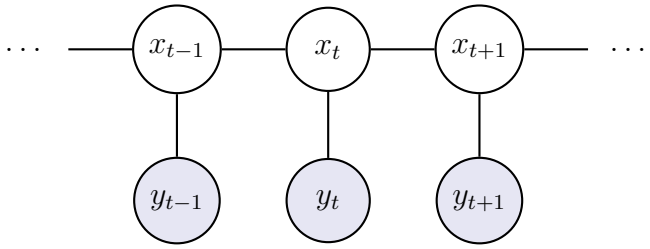


Figure 1.2: *HMM with states* $\{x_t\}_{t=1}^T$ *and observations* $\{y_t\}_{t=1}^T$.

HMMs are used in a broad range of applications from modelling time series data to speech processing (see for example Ghahramani (2001); Gales and Young (2007);

Zucchini et al. (2016)). In HMMs, the node potentials correspond to the likelihood of the corresponding observation and the edge potentials correspond to the *transition density*:

$$\psi_t(x_t) \;=\; p(y_t \,|\, x_t), \quad \text{and} \quad \psi_{t-1,t}(x_{t-1}, x_t) \;=\; p(x_t \,|\, x_{t-1}).$$

A prior $\pi_0 \in \mathcal{P}(\mathcal{X})$ is assumed to be given for the first node so that we can write $p(x_1 \,|\, y_1) \propto \pi_0(x_1) p(y_1 \,|\, x_1)$.

In the case of the HMM, obtaining or approximating the singleton marginals on the nodes from 1 to T given the observations $y_{1:T}$ is known as a *smoothing problem*. The marginals or *smoothing densities* can be written $p(x_t \,|\, y_{1:T})$ to make explicit the dependence on all available observations.

A related problem is the *filtering problem* where one is only interested in building the last singleton marginal or, to put the problem in the same framework as before, to build marginals taking into account only the observations available until the point considered. This can be useful in an online setting where one is streaming data. The filtering densities are written $p(x_t \,|\, y_{1:t})$. Correspondingly, we are often interested in representing the prediction density obtained by integrating the filtering densities with the transition density:

$$p(x_{t+1} \,|\, y_{1:t}) \;\propto\; \int p(x_{t+1} \,|\, x_t) p(x_t \,|\, y_{1:t}) \,\mathrm{d}x_t. \tag{1.4}$$

The *linear-Gaussian* case is a particular instance of HMM, usually expressed as:

$$\begin{cases} \pi_0(x_1) &= \mathcal{N}(x_1; \mu_0, Q_0) \\ p(x_t \,|\, x_{t-1}) &= \mathcal{N}(x_t; A_t x_{t-1} + a_t, Q_t) \\ p(y_t \,|\, x_t) &= \mathcal{N}(y_t; B_t x_t + b_t, R_t) \end{cases}$$

where μ_0 as well as the Q_t, R_t, A_t, a_t, B_t and the b_t are fixed and deterministic. In such a case, an analytical expression for both the filtering and the smoothing densities can be obtained through the well-known *Kalman filter* and *Kalman smoother* (Anderson and Moore, 1979).

However, in the general case where the transition is nonlinear and/or non-Gaussian, the marginals are typically intractable. Approximation algorithms such as *Sequential Monte Carlo methods* (SMC) have been developed in order to generate approximate samples from these marginals and consequently be able to form Monte Carlo estimators. We introduce these methods in more detail in chapter 2.

1.2.2 Star graphs

In this document, we define *star graphs* as corresponding to a structure with a single random variable (possibly high-dimensional) with a node potential that factors into several likelihood terms. The structure is illustrated in figure 1.3.

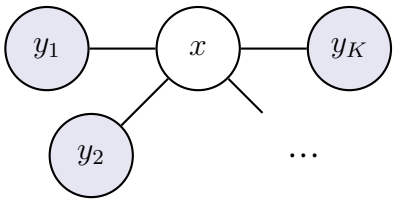


Figure 1.3: *Star graph with hidden state x and observations $\{y_k\}_{k=1}^K$.*

In this case, the singleton marginal can simply be written as:

$$p(x \mid y) \quad \propto \quad \pi_0(x) \prod_{i=1}^{K} p(y_k \mid x)$$

where the y_k are subsets of the entire observed data and π_0 is a prior on the hidden state. This can be a useful representation for distributed inference where each of the observation nodes can correspond to a distinct physical machine with a portion of the relevant data. We exploit this structure to suggest a novel way to approach distributed

Bayesian inference in chapter 6.

1.2.3 Grid and loopy graphs

The examples listed above do not exhibit cycles. An example of a common MRF structure with cycles which is encountered, for instance, in image processing, is the *grid graph* illustrated in figure 1.4.

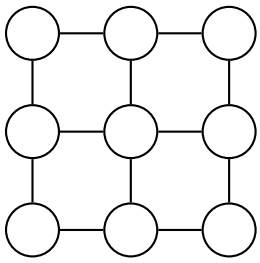


Figure 1.4: *Generic structure of a grid graph.*

In image processing, grid graphs can be used to model interactions between the pixels of an image (Blake et al., 2011). In the case of image denoising for example, for each pixel k a noisy value y_k is observed and we can have a model for the likelihood of y_k given the true pixel value x_k in the form $p(y_k \mid x_k)$. These form the node potentials. Additionally, we may have a similarity measure which can be used to penalise neighbouring pixels being very dissimilar. These form the edge potentials.
The problem of finding or approximating the singleton marginals then corresponds to finding the likelihood of a particular pixel taking a specific value given all the noisy observations available. We consider the problem of approximating marginals on such loopy graphs in chapter 7.

1.3 Exact vs Inexact Methods and Structure of the Thesis

While the unifying theme of this thesis is the inference problem on Markov Random Fields, we consider two rather different approaches forming the two main parts of

the thesis. In the first approach – *sampling methods* – we consider methods to produce samples from the target densities (marginals of a MRF) from which Monte Carlo estimators can be computed. These methods take the form of stochastic processes which, asymptotically, are guaranteed to generate samples from the target distribution. These methods are considered "exact" because, asymptotically, they yield samples from the correct distribution. The difficulty of course is to assess whether the generating process has converged in the context of finite compute time and finite precision arithmetics.

In the second approach – *approximated Bayesian inference* – we consider methods to produce proxies to the target densities and compute expected values with respect to these proxies. These methods are considered "inexact" because even in the limit of infinite computational power or infinite precision arithmetic they do not lead to the correct expected values.

Formally, let us denote p a target distribution of interest and φ a test function and put ourselves in the realistic context of finite computational power and precision. The aim is to compute the expected value of φ with respect to p or $\mathbb{E}_p[\varphi(X)]$. The first class of methods generates incorrect samples $\{\hat{x}_1, \dots, \hat{x}_n\}$ which are hopefully statistically not too far from exact (but usually inaccessible) samples $\{x_1, \dots, x_n\}$ drawn iid from p. As will be covered in section 2.1, the expected value of interest will be approximated as $\mathbb{E}_p[\varphi(X)] \approx n^{-1} \sum_{i=1:n} \varphi(\hat{x}_i)$. The second class of methods attempts to find a simple distribution q which is close in some sense to the target distribution p, the expected value of interest is consequently approximated as $\mathbb{E}_p[\varphi(X)] \approx \mathbb{E}_q[\varphi(X)]$.

These two approaches rely on different sets of tools. The first one relies primarily on sampling/particle methods while the second relies more heavily on optimisation tools. To simplify the presentation of the thesis, we therefore have two background chapters beginning each of the two parts. Each part is then formed of two chapters covering a specific method and discussing applications.

Part I

Sampling Methods

2 | Background

In this chapter, we start with a brief overview of Monte Carlo estimators and associated sampling methods focusing in particular on Sequential Monte Carlo methods for the smoothing and filtering problem on Hidden Markov Models. We also introduce an alternative class of methods relying upon Piecewise Deterministic Markov Processes and in particular the Bouncy Particle Sampler (BPS).

2.1 Monte Carlo Methods

In this section we review briefly the foundations of Monte Carlo methods as well as a few key algorithms which will be discussed in the rest of the document.

2.1.1 From quadrature to sampling

We consider the problem of computing the expected value of a test function φ taking value over a space $\mathcal{X}$ with respect to a distribution $\pi \in \mathcal{P}(\mathcal{X})$ i.e.: $I := \mathbb{E}_\pi[\varphi(X)]$. Assuming it cannot be computed analytically, a general approach is to consider a quadra-

ture rule of the form:

$$I \approx \widehat{I}_N = \sum_{i=1}^{N} w(X^{(i)})\varphi(X^{(i)}) \tag{2.1}$$

for some fixed points $X^{(i)} \in \mathcal{X}$ and corresponding weights $w(X^{(i)}) \in \mathbb{R}_+$. When the number of dimensions is low, we can consider deterministic quadrature rules such as the Gauss-Hermite quadrature (see e.g.: Davis and Rabinowitz (1975)). However, as the dimensionality increases, the performance of these deterministic rules becomes catastrophic even for a large number of integration points (a well-known effect related to what Bellman called the *curse of dimensionality* in dynamic programming (Bellman, 1957; Bengtsson et al., 2008)). In such cases, a broadly studied approach is the Monte Carlo integration with

$$X^{(i)} \sim_{\text{iid}} \pi, \quad \text{and} \quad w(X^{(i)}) = N^{-1}. \tag{2.2}$$

The strong law of large numbers then indicates that $\widehat{I}_N \to I$ almost surely with approximation error scaling like $\mathcal{O}(N^{-1/2})$ independently of the dimensionality of the problem (see e.g. Robert and Casella (2004)). This is to be compared with deterministic rules which typically have approximation error scaling like $\mathcal{O}(N^{-\alpha/d})$ for a fixed $\alpha > 0$ depending on the quadrature rule (Caflisch, 1998). The problem then becomes one of drawing iid samples from π which is often also an intractable problem. Note that (2.2) in fact defines an *empirical measure* $\hat{\pi}$ with

$$\hat{\pi}(x) := N^{-1} \sum_{i=1}^{N} \delta_{X^{(i)}}(x), \tag{2.3}$$

and computing $\hat{I}_N$ amounts to taking the expected value of $\varphi(X)$ with respect to $\hat{\pi}$.

2.1.2 Importance sampling

In *importance sampling* (IS), samples are drawn from a *proposal distribution* $q \in \mathcal{P}(\mathcal{X})$ that is easy to sample from (e.g.: a Gaussian) and is similar to the target distribution π. The quadrature weights are then adjusted to reflect that the samples are not drawn from the true distribution:

$$X^{(i)} \sim_{\text{iid}} q(\cdot), \quad \text{and} \quad w(X^{(i)}) \propto W(X^{(i)}) = \frac{\pi(X^{(i)})}{q(X^{(i)})}, \tag{2.4}$$

where the weights $w(X^{(i)})$ are normalised to sum up to one.

Provided the support of q includes that of π i.e., $\pi(x) > 0 \Rightarrow q(x) > 0$, the resulting *importance sampling estimator* can be shown to be consistent using the strong law of large numbers; i.e., the estimator converges almost surely to the true expected value:

$$\sum_{i=1}^{N} w(X^{(i)})\varphi(X^{(i)}) \xrightarrow[N\to\infty]{\text{a.s.}} \mathbb{E}_\pi[\varphi(X)], \tag{2.5}$$

for a test function φ and the estimator is finite provided $\mathbb{E}_\pi[\varphi(X)^2 w(x)] < \infty$ (Robert and Casella, 2004, chapter 3.3). Note that (2.4) also defines an empirical measure $\hat{\pi}$ with

$$\hat{\pi}(x) = \sum_{i=1}^{N} w(X^{(i)})\delta_{X^{(i)}}(x). \tag{2.6}$$

Effective sample size

One way to assess the quality of an importance sampling estimator is to consider the ratio of the variance of the corresponding Monte Carlo estimator and the variance of the IS estimator. The general ratio is hard to handle and depends upon the test func-

tion with respect to which the expected value is computed. Kong (1992) suggested considering the following proxy known as the *effective sample size*:

$$\text{ESS} \quad = \quad \frac{N}{1 + \mathbb{V}_q[W(X)]}, \tag{2.7}$$

where $\mathbb{V}_q[W(X)]$ is the variance of the importance sampling weights. Using the sample variance of the normalised weights and simplifying the expression, the proxy that is usually considered for the ESS nowadays is

$$\text{ESS} \quad = \quad \left(\sum_{i=1}^{N} w(X^{(i)})^2 \right)^{-1}. \tag{2.8}$$

This metric is bounded from below by 1 – the degenerate case where a single particle is carrying all the weight – and from above by N – the ideal case where the samples are comparable to ideal Monte Carlo samples.

2.1.3 Classical sampling algorithms

In this point we mention briefly a few standard algorithms used to attempt to generate samples from a distribution. Later on, we will compare alternative methods to these standard algorithms. The list and descriptions are not meant to be exhaustive and we refer to Robert and Casella (2004); Green et al. (2015) for a more complete overview.

Metropolis-Hasting

The Metropolis-Hasting algorithm (MH) is one of the most prominent member of the class of Markov Chain Monte Carlo methods (MCMC). In MCMC methods, one considers a sequence (chain) of random variables $X^{(1)}, X^{(2)}, \ldots$ following a conditional probability density or *transition kernel* K with $X^{(n+1)} \sim K(X^{(n)}, \cdot)$. The kernel is chosen in such a way that the chain admits a stationary distribution, i.e.: a distribution π such

that for n large enough, $X^{(n)} \sim \pi$ implies that $X^{(n+1)} \sim \pi$. Further, the kernel is chosen in such a way that the stationary distribution corresponds to the target distribution.

The MH algorithm corresponds to the transition kernel below where q is a proposal distribution and p_u is the (unnormalised) target (Robert and Casella, 2004, chapter 6).

Algorithm 1 *Metropolis-Hasting transition*

1: sample $Y^{(n)}$ from $q(\cdot|X^{(n)})$
2: compute the acceptance rate

$$\rho \leftarrow \min\left\{\frac{p_u(Y^{(n)})}{p_u(X^{(n)})}\frac{q(X^{(n)}|Y^{(n)})}{q(Y^{(n)}|X^{(n)})}, 1\right\}$$

3: generate $\beta \sim \mathcal{B}(\rho)$ (Bernoulli)
4: compute the new state: $X^{(n+1)} \leftarrow \beta Y^{(n)} + (1-\beta)X^{(n)}$

In the MH algorithm, different choices of q will lead to different algorithms that may be easier or harder to implement. For example taking a *symmetric* proposal with $q(x|y) = q(y|x)$ leads to a simplified acceptance ratio (see Green et al. (2015) for a discussion).

Gibbs sampling

Gibbs sampling can be applied when targeting a multivariate distribution $p \in \mathcal{P}(\mathcal{X})$ where $\mathcal{X}$ is d-dimensional and where it is possible to generate samples from the target's full conditionals $p_1, \ldots, p_d$ (Robert and Casella, 2004, chapter 7).

Algorithm 2 *Gibbs iteration*

1: given $x^{(n)} = (x_1^{(n)}, \ldots, x_d^{(n)})$
2: **for** $k = 1 : d$ **do**
3: sample $X_k^{(n+1)}$ from $p_k(x_k \,|\, x_{-k}^{(n)})$ with $x_{-k}^{(n)} := (x_1^{(n+1)}, \ldots, x_{k-1}^{(n+1)}, x_{k+1}^{(n)}, \ldots, x_d^{(n)})$
4: **end for**

The potential advantage of the method is that, at any given point, it only considers uni-dimensional distributions. The major disadvantage of the method is that it goes

sequentially over each dimensions rendering the method extremely slow as soon as the dimensionality of the target distribution becomes substantial.

Hamiltonian Monte Carlo sampling

The Hamiltonian Monte Carlo method (HMC) generates samples along a trajectory on the the parametric surface $(x, -U(x))$ where $-U(x) := \log \pi_u(x)$ and π_u is the (unnormalised) target distribution. The trajectory can be interpreted as the time evolution of a specific physical system described by a Hamiltonian depending on the target distribution (potential) and a momentum term encouraging exploration of the state-space (kinetic energy) with *momentum variable* p. The Hamiltonian is defined as

$$H(x,p) \quad := \quad U(x) + \frac{1}{2}\left\langle p, M^{-1}p \right\rangle \tag{2.9}$$

where M is a mass matrix that can be set to the identity or adjusted depending on knowledge of the problem's geometry (Betancourt, 2017; Barp et al., 2018). Since finding the exact trajectory of a Hamiltonian system is usually intractable, numerical integrators can be used and the leapfrog method is often used in HMC.

Algorithm 3 *HMC iteration*

1: draw a momentum variable $P^{(n)} \sim \mathcal{N}(0, M)$
2: compute L steps of the leapfrog integrator from $(X^{(n)}, P^{(n)}) \to (X^\star, P^\star)$
3: compute the acceptance ratio

$$\rho \leftarrow \min\left\{\exp\left(-\Delta H\right), 1\right\}$$

where $\Delta H \leftarrow H(X^\star, P^\star) - H(X^{(n)}, P^{(n)})$.
4: generate $\beta \sim \mathcal{B}(\rho)$
5: compute the new state: $X^{(n+1)} \leftarrow \beta X^\star + (1-\beta)X^{(n)}$

This method generates a Markov chain with stationary distribution corresponding to the target distribution π. It is a very popular method in computational Bayesian inference and tends to be preferred over approaches such as the MH algorithm (Neal,

2011). We will show later in chapter 4 than in some cases a method that exploits the factorisation structure of the target distribution explicitly may lead to better results.

2.2 Sequential Monte Carlo Methods

2.2.1 Sequential importance sampling

In the context of HMMs, we are usually interested in estimating a sequence of target distributions $\{\pi_t(x_{1:t})\}_{t=1}^T$ which admit the following factorisation structure:

$$\pi_t(x_{1:t}) \quad \propto \quad \pi_t(x_t \,|\, x_{1:t-1})\pi_{t-1}(x_{1:t-1}).$$

This led to the development of *sequential Monte Carlo* (SMC) methods (Robert and Casella, 2004, chapter 14.3). The underlying principle of sequential importance sampling is the same as that of importance sampling except that a different proposal is considered at every step t taking into account the previous draw of particles and the evolution of the system:

$$q_t(x_{1:t}) \quad \propto \quad q_t(x_t \,|\, x_{1:t-1})q_{t-1}(x_{1:t-1}).$$

Following this form, new samples or *particles* $X_t^{(i)}$ can be drawn from $q_t(x_t \,|\, X_{1:t-1}^{(i)})$ and the weights corresponding to the trajectories $\{X_{1:t}^{(i)}\}$ then need to be multiplied by a factor $\alpha_t^{(i)}$ with

$$\alpha_t^{(i)} := \frac{\pi_t(X_{1:t}^{(i)})}{\pi_{t-1}(X_{1:t-1}^{(i)})q_t(X_t^{(i)} \,|\, X_{1:t-1}^{(i)})}. \tag{2.10}$$

The variance of an IS estimator is directly related to the variance of the associated importance weights. In order to counter the increase of variance induced by the se-

quential IS procedure, the proposal at step t should be such that the variance of the update factors α_t is as small as possible. In particular, the *optimal proposal* (Doucet and Johansen, 2011) keeps it at zero with

$$q_t^{\text{opt}}(x_t \mid x_{1:t-1}) \quad := \quad \pi_t(x_t \mid x_{1:t-1}). \tag{2.11}$$

Additionally, since we cannot typically sample easily from the optimal proposal, we have to resort to approximating distributions.

2.2.2 Particle filtering

In the filtering problem on a HMM, the target densities are $\pi_t(x_{1:t}) = p(x_{1:t} \mid y_{1:t})$ and their marginals. The incremental update factors are given by

$$\begin{aligned} \alpha_t(x_{1:t}) &= \frac{p(x_{1:t} \mid y_{1:t})}{p(x_{1:t-1} \mid y_{1:t-1}) q_t(x_t \mid x_{1:t-1})} \\ &= \frac{p(x_t, y_t \mid x_{1:t-1}, y_{1:t-1}) p(x_{1:t-1} \mid y_{1:t-1}) p(y_{1:t-1})}{p(x_{1:t-1} \mid y_{1:t-1}) q_t(x_t \mid x_{1:t-1}) p(y_{1:t})} \\ &\propto \frac{p(y_t \mid x_t) p(x_t \mid x_{t-1})}{q_t(x_t \mid x_{1:t-1})}, \end{aligned} \tag{2.12}$$

where we exploited the conditional dependence structure of the HMM. Consequently, the optimal proposal is

$$q_t^{\text{opt}}(x_t \mid x_{t-1}) \quad \propto \quad p(y_t \mid x_t) p(x_t \mid x_{t-1}), \tag{2.13}$$

which could also have been obtained from applying (2.11).

Beyond simply corresponding to sequential importance sampling for the filtering problem, the particle filter adds an extra step: resampling. This step alleviates the problem of weight degeneracy whereby the variance of estimators grows over time when one considers a suboptimal proposal distribution such as the bootstrap proposal. The

resampling step consists of resampling particles with replacement depending upon their weights. This, in practice, is done when the ESS comes under a pre-assigned threshold (Del Moral et al., 2006). Several schemes exist with the simplest one being the *Multinomial Resampling* where n particle indices are drawn from a Multinomial with weights corresponding to the n weights of the initial particles.
Resampling is a crucial step needed to make SMC methods viable (Hol et al., 2006; Doucet and Johansen, 2011). However, the methods discussed in this thesis do not depend upon the resampling scheme and therefore we will assume in the sequel that a resampling scheme is applied without discussing it further. In the experiments we use the standard multinomial resampling.

A skeleton of a particle filter algorithm is given below.

Algorithm 4 *Particle filter*

1: sample $X_1^{(i)}$ from q_1 for $i = 1, \ldots, N$
2: compute and normalise the weights $w_1(X_1^{(i)}) \propto \pi_0(X_1^{(i)})p(y_1 \mid X_1^{(i)})/q_1(X_1^{(i)})$
3: **for** $t = 2 : T$ **do**
4: sample $X_t^{(i)}$ from $q_t(\cdot \mid X_{t-1}^{(i)}, y_t)$ with $q_t \approx q_t^{\text{opt}}$ for $i = 1, \ldots, N$
5: update and normalise the weights $w_t^{(i)} \propto \alpha_t^{(i)} w_{t-1}^{(i)}$
6: resample the particles if necessary
7: **end for**
8: **return** weighted set of particles $\{X_{1:T}^{(i)}, w_{1:T}^{(i)}\}_{i=1}^N$

The computational complexity of algorithm 4 is $\mathcal{O}(TN)$. Indeed, at each step t, the algorithm samples N particles and computes their corresponding weights which has linear complexity in the number of particles.

Sampling directly from the optimal proposal is often an intractable problem by itself. In the context of filtering, an alternative choice is the *bootstrap proposal* (Doucet and Johansen, 2011) with

$$q^{\text{bs}}(x_t \mid x_{t-1}) \quad := \quad p(x_t \mid x_{t-1}), \tag{2.14}$$

which is often easier to sample from. In that case, the update factor simply reduces to $\alpha_t^{(i)} \propto p(y_t \mid X_t^{(i)})$. However, since the likelihood and the transition density may not be well aligned, those update factors can vary a lot incurring an increase in the variance of the resulting estimator. This problem is particularly prevalent in high dimensions when the distributions considered tend to be more concentrated.

2.2.3 Particle smoothing

In the smoothing problem on a HMM, the target densities are $p(x_t \mid y_{1:T})$. These densities can be expressed as the marginals of joint densities over subsequent states:

$$p(x_t \mid y_{1:T}) = \int p(x_t, x_{t+1} \mid y_{1:T}) \, \mathrm{d}x_{t+1}.$$

Exploiting the conditional dependence structure of the HMM, the integrand can be factorised leading to

$$\begin{aligned} p(x_t \mid y_{1:T}) &= \int p(x_t \mid x_{t+1}, y_{1:t}) p(x_{t+1} \mid y_{1:T}) \, \mathrm{d}x_{t+1} \\ &= p(x_t \mid y_{1:t}) \int \frac{p(x_{t+1} \mid x_t)}{p(x_{t+1} \mid y_{1:t})} p(x_{t+1} \mid y_{1:T}) \, \mathrm{d}x_{t+1}. \end{aligned} \tag{2.15}$$

This relation leads to the *forward filtering, backward smoothing* (FFBS) algorithm (Hürzeler and Künsch, 1998; Doucet et al., 2000). Plugging the particle approximation to the filtering distribution in equation (2.15) gives:

$$\begin{aligned} \hat{p}(x_t \mid y_{1:T}) &= \hat{p}(x_t \mid y_{1:t}) \int \frac{p(x_{t+1} \mid x_t)}{\int p(x_{t+1} \mid x_t) \hat{p}(x_t \mid y_{1:t}) \, \mathrm{d}x_t} \hat{p}(x_{t+1} \mid y_{1:t+1}) \, \mathrm{d}x_{t+1} \\ &= \sum_{i=1}^{N} w_{t \mid T}^{(i)} \delta_{X_t^{(i)}}(x_t), \end{aligned} \tag{2.16}$$

where the smoothing weights are given recursively by

$$w_{t\,|\,T}^{(i)} \propto w_t^{(i)} \sum_{j=1}^{N} w_{t+1\,|\,T}^{(j)} \left[\frac{p(X_{t+1}^{(j)} \,|\, X_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(X_{t+1}^{(j)} \,|\, X_t^{(k)})} \right]. \tag{2.17}$$

In essence, the FFBS algorithm simply recycles a particle filter updating its weights according to equation (2.17).

The computation of the updated smoothing weights has complexity $\mathcal{O}(TN^2)$ since, at each step, we need to consider the matrix of all pairwise interactions between the particles at two subsequent steps: $[p(X_{t+1}^{(j)} \,|\, X_t^{(i)})]_{i,j=1}^{N}$. Since the FFBS algorithm recycles the particles from a particle filter, the performances of the resulting estimators can suffer if the support of the filtering distribution at step t is significantly distinct from that of the smoothing distribution at step t.

2.2.4 Convergence of particle filters and smoothers

The resampling mechanism considered in SMC methods means that the particles are no longer statistically independent which makes convergence results harder to get than for vanilla sequential importance sampling (where convergence results can be extended from those of importance sampling). However, there is a Central Limit Theorem (CLT) for the estimator of $\mathbb{E}_{\pi_t}[\varphi]$ using SMC in the broad sense showing convergence in distribution to the true expected value (Chopin, 2004).

For the specific case of the particle filter, stronger results exist: under mild conditions the empirical measure $\hat{p}(x_t \,|\, y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{X_t^{(i)}}(x_t)$ converges almost surely to the true filtering distribution $p(x_t \,|\, y_{1:t})$ and estimators of expected values taken with respect to it converge in the mean-squared sense as $N \to \infty$ (Crisan and Doucet, 2002). These results can be extended particle smoothing methods as was done for example in Godsill et al. (2004).

2.3 Sampling with Piecewise Deterministic Markov Processes

The *Bouncy Particle Sampler* (BPS) of Bouchard-Côté et al. (2015) is inspired from an algorithm in the physics literature (Peters and de With, 2012) and belongs to a wider class of sampling methods based on Piecewise Deterministic Markov Processes (PDMP) (Bierkens et al., 2016, 2017; Wu and Robert, 2017; Vanetti et al., 2017).

The BPS algorithm considers a target distribution π on $\mathbb{R}^d$, proportional to a non-negative function γ which can be evaluated pointwise: $\pi(x) = Z^{-1}\gamma(x)$ but the normalisation constant Z is intractable. The objective is again the computation of expected values of the form $\mathbb{E}_\pi[\varphi]$ for an arbitrary test function $\varphi : \mathbb{R}^d \to \mathbb{R}$.

Defining the *energy* function U as $U(x) = -\log\gamma(x)$ and assuming it is continuously differentiable, the algorithm generates a piecewise linear path based on ∇U. Each segment is specified by an initial position $x^{(i)} \in \mathbb{R}^d$, a length $\tau_{i+1} \in \mathbb{R}^+$ and a velocity $v^{(i)}$ with the recurrence $x^{(i+1)} = x^{(i)} + v^{(i)}\tau_{i+1}$. The times where the velocity changes are given by the cumulative sums, $t_i = \sum_{j=1}^{i}\tau_j$ with $t_0 = 0$. The continuous positions along the path are therefore given by

$$x(t) \;=\; x^{(i)} + v^{(i)}(t - t_i), \quad \text{for} \quad t \in [t_i, t_{i+1}). \tag{2.18}$$

The authors show that, if the lengths $\{\tau_i\}_{i\geq 1}$ are governed by a specific Inhomogenous Poisson Process (IPP) and if the speeds are modified for each segment according to a simple reflection mechanism, then the target expected values can be approximated via an integral of the test function along the path given by (2.18):

$$\begin{aligned}\mathbb{E}_\pi[\varphi] &\approx T^{-1}\int_0^T \varphi(x(t))\mathrm{d}t \\ &\approx T^{-1}\left(\sum_{i=1}^{n-1}\int_0^{\tau_i}\varphi(x^{(i-1)} + v^{(i-1)}s)\,\mathrm{d}s + \int_0^{t_n - T}\varphi(x^{(n-1)} + v^{(n-1)}s)\mathrm{d}s\right),\end{aligned}$$

and this estimator converges almost surely to the true expected value with $T \to \infty$ (Bouchard-Côté et al., 2015, theorem 1).

In the BPS algorithm, the IPP governing the lengths of the trajectory segments has intensity function $\lambda : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ with

$$\lambda(x, v) \quad = \quad \langle \nabla U(x), v \rangle^+ . \tag{2.19}$$

At the end of each segment, the trajectory "bounces" and the velocity is reflected against the local level set of the energy U:

$$v' \quad = \quad R(x)v \quad = \quad v - 2\frac{\langle \nabla U(x), v \rangle \, \nabla U(x)}{\|\nabla U(x)\|_2^2}. \tag{2.20}$$

This is illustrated at the figure 2.1 below where $\hat{n} := \nabla U(x)/\|\nabla U(x)\|$.

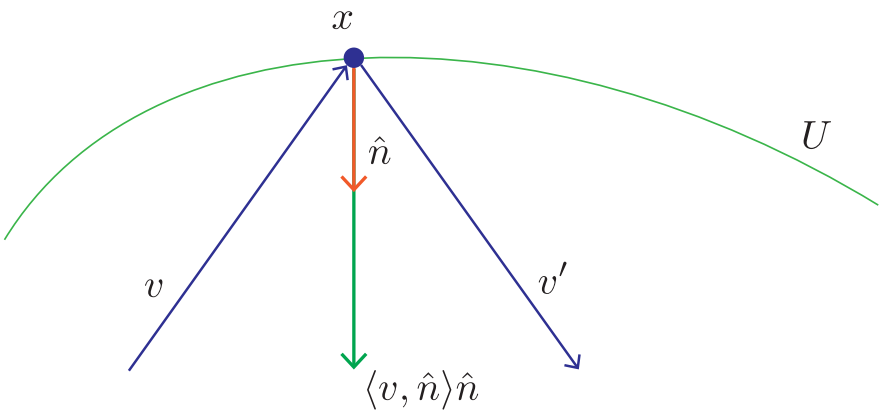


Figure 2.1: *Illustration of the specular reflection. The new velocity vector v' is obtained by doing a specular reflection of the old vector v against the level set of U passing by x (green line). This is obtained by subtracting twice the projection (green arrow) of the velocity on the normal vector to U at x (orange arrow). Figure best seen in colour.*

The authors show that the velocities also need to be "refreshed" according to the arrival times of a (homogenous) Poisson process (PP) of intensity $\lambda^{\text{ref}} \geq 0$, a parameter of the BPS. This ensures the algorithm explores the entirety of the space. The basic

BPS algorithm is reproduced in algorithm 5 below.

Algorithm 5 *Basic BPS*

initialise $(x^{(0)}, v^{(0)})$, set T, trajectory length, set $t = 0$ and $i = 1$
while $t < T$ **do**
    simulate first arrival time $\tau_{\text{bounce}} \sim \text{PP}(\chi(t))$ with

$$\chi(t) = \lambda(x^{(i-1)} + v^{(i-1)}t, v^{(i-1)})$$

    simulate $\tau_{\text{ref}} \sim \text{Exp}(\lambda^{\text{ref}})$
    let $\tau_i \leftarrow \min(\tau_{\text{bounce}}, \tau_{\text{ref}})$
    compute the next position $x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau_i$
    if $\tau_i = \tau_{\text{ref}}$ **then**
        sample next velocity $v^{(i)} \sim \mathcal{N}(0, I)$
    else if $\tau_i = \tau_{\text{bounce}}$ **then**
        reflect the velocity $v^{(i)} \leftarrow R(x^{(i)})v^{(i-1)}$
    end if
    let $t \leftarrow t_i \leftarrow t_{i-1} + \tau_i$
    update $i \leftarrow i + 1$
end while

The process for two subsequent time steps is illustrated at the figure 2.2 below.

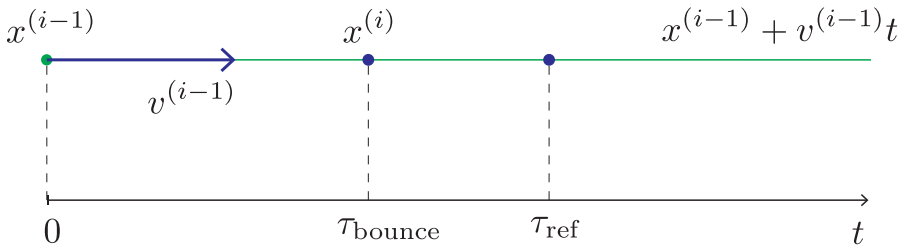


Figure 2.2: *Illustration of the process to go from $x^{(i-1)}$ to $x^{(i)}$ in the BPS: a bouncing time τ_{bounce} is drawn from the IPP with intensity $\chi(t)$ along the trajectory $x^{(i-1)} + v^{(i-1)}t$ (green line); a reference time τ_{ref} is drawn from an exponential, the minimum of the two (here τ_{bounce}) leads to the position of the next point $x^{(i)}$ where either a bounce or a refreshment of the velocity will happen (here a bounce). Figure best seen in colour.*

As we saw, the algorithm requires to sample the first arrival time of an IPP of intensity $\chi(t) = \lambda(x + vt, v)$ where $\lambda(x, v) = \langle \nabla U(x), v \rangle^+$. Sampling from an IPP is a well explored topic in the literature and the authors summarise three possible approaches which we list briefly below:

- *Time-scale transformation*: this approach requires computing the inverse quantile of $\Xi(t) := \int_0^t \chi(s)\mathrm{d}s$ which is usually intractable but it can be computed numerically if the target is strictly log-concave and differentiable. Given V drawn from a Uniform on $[0, 1]$, the first arrival time of the IPP is given by $\Xi^{-1}(-\log(V))$.
- *Adaptive thinning*: this approach requires an upper bounds $\overline{\chi}_s(t)$ with $\overline{\chi}_s(t) = 0$ for $t < s$ and $\overline{\chi}_s(t) \geq \chi(t)$ for $s \leq t \leq s + \Delta(s)$ where Δ is a positive function (in the standard case, $\Delta = +\infty$). Additionally, it requires the ability to sample from the IPP corresponding to $\overline{\chi}_s$. Let τ be such a sample with $\tau \leq s + \Delta$, there is then an accept-reject step with rate $\chi(\tau)/\overline{\chi}(\tau)$. Ideally, Δ and $\chi/\overline{\chi}_s$ are large (so that we do not have to simulate too many candidates which would incur an increased computational cost).
- *Superposition*: if the energy U can be decomposed in a sum $U(x) = \sum_{j=1}^m U^{[j]}(x)$ and that each term can be targeted (via one of the first two methods), then we can use the thinning method with bound $\sum_{j=1}^m \chi^{[j]}(t)$.

The key elements when trying to deploy the Bouncy Particle Sampler in practice are to select an appropriate λ_{ref} (too low and the processus is essentially a random walk, too high and some regions of space may be left unexplored for a long time) and to implement an appropriate way to sample the first arrival time of the IPP which will depend on the target distribution.

An advantage of the BPS over methods such as HMC is that it can work seamlessly on constrained domains as is shown in Bierkens et al. (2017).

3 | Backward Information Smoothing

In this chapter, we focus on the problem of approximating the smoothing distributions for a Hidden Markov Model (HMM) with continuous state-space. We assume that a sequence of empirical densities approximating the filtering distributions has already been obtained through a particle filtering step. We had seen at point 2.2.3 that the *forward filtering backward smoothing* (FFBS) algorithm can produce an approximation to the smoothing distributions by recycling a particle filter and updating its weights. As shown in point 2.2.3, the FFBS algorithm has quadratic complexity and may suffer from the lack of resampling step if the smoothing distribution and the filtering distribution have significantly different support.

In this chapter, we introduce a class of algorithms that can be obtained from another factorisation of the smoothing distributions than that used to obtain the FFBS. We also introduce algorithms with a computational complexity that is sub-quadratic in N and compare these algorithms in numerical experiments.

3.1 Two Filter Smoothing

3.1.1 Two filter formula

An alternative to the FFBS approach is to exploit the *two filter formula*. To derive it, note that the smoothing distribution can be written in the following way using the definition of conditional probability:

$$p(x_t \mid y_{1:T}) \quad = \quad \frac{p(x_t, y_{1:t-1}, y_{t:T})}{p(y_{1:T})}.$$

Then, exploiting the conditional dependences of the HMM, we get

$$\begin{aligned} p(x_t \mid y_{1:T}) &\propto p(y_{t:T} \mid x_t, y_{1:t-1}) p(x_t, y_{1:t-1}) \\ &\propto p(x_t \mid y_{1:t-1}) p(y_{t:T} \mid x_t). \end{aligned} \tag{3.1}$$

This last factorised form is the two filter formula which leads to the *two filter smoothing* (TFS) algorithm (Bresler, 1986; Kitagawa, 1996). The first term in (3.1) is the prediction density (PD), the second term $p(y_{t:T} \mid x_t)$ is called the *backward information filter* (BIF). To simplify developments we introduce the following (non-standard) notations for the predictive density and the backward information filter:

$$\mathsf{PD}_t(x_t) \quad := \quad p(x_t \mid y_{1:t-1}), \quad \text{and} \quad \mathsf{BIF}_t(x_t) \quad := \quad p(y_{t:T} \mid x_t). \tag{3.2}$$

The prediction density is obtained by integrating the filtering density multiplied by the transition density:

$$\mathsf{PD}_t(x_t) \quad = \quad \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \, \mathrm{d}x_{t-1}. \tag{3.3}$$

an approximation to the prediction density can therefore easily be obtained by plugging a particle representation of the filtering density in (3.3).

3.1.2 Targeting the backward information filter

The backward information filter cannot be directly targeted in a SMC framework in general as it may not be proportional to a distribution in x_t. In Briers et al. (2010), the authors therefore suggest to introduce a sequence of artificial *normalisation densities* $\{\gamma_t\}_{t=1}^T \in \mathcal{P}(\mathcal{X})$ such that, for each step t, the product of γ_t and BIF_t is normalisable. In other words, the normalising distribution γ_t allows to define a *normalised backward information filter* in $\mathcal{P}(\mathcal{X})$ with

$$\widetilde{\mathsf{BIF}}_t(x_t) \quad := \quad Z_t^{-1}\gamma_t(x_t)\mathsf{BIF}_t(x_t) \tag{3.4}$$

for some normalising constant Z_t^{-1}. Any normalisation $\gamma_t \in \mathcal{P}(\mathcal{X})$ can be chosen as long as it verifies the *support condition* i.e.: as long as its support covers that of the backward information filter:

$$\mathsf{BIF}_t(x_t) \quad > \quad 0 \quad \Longrightarrow \quad \gamma_t(x_t) \quad > \quad 0. \tag{3.5}$$

In order to target the sequence of normalised BIF in a SMC framework, it is useful to show that it verifies a factorisation structure similar to that of the filtering distribution. For this, we start by noting that

$$\begin{aligned} p(y_{t:T} \,|\, x_t, x_{t+1}) &= p(y_{t+1:T} \,|\, x_t, x_{t+1}, y_t)p(y_t \,|\, x_t, x_{t+1}) \\ &= p(y_t \,|\, x_t)\mathsf{BIF}_{t+1}(x_{t+1}). \end{aligned} \tag{3.6}$$

On the other hand, we also have that

$$\begin{aligned} p(y_{t:T} \mid x_t, x_{t+1}) &= \frac{p(y_{t:T}, x_t, x_{t+1})}{p(x_{t+1} \mid x_t) p(x_t)} \\ &= \frac{p(x_{t+1} \mid x_t, y_{t+1:T}) \mathsf{BIF}_t(x_t)}{p(x_{t+1} \mid x_t)}, \end{aligned} \tag{3.7}$$

where we have used the conditional independence structure of the HMM. Combining (3.6) and (3.7) then leads to

$$\mathsf{BIF}_t(x_t) p(x_{t+1} \mid x_t, y_{t+1:T}) = p(y_t \mid x_t) p(x_{t+1} \mid x_t) \mathsf{BIF}_{t+1}(x_{t+1}). \tag{3.8}$$

If we now introduce the normalisation densities γ_t and γ_{t+1} and the corresponding normalisation constants Z_t and Z_{t+1} in (3.8) we get

$$\widetilde{\mathsf{BIF}}_t(x_t) p(x_{t+1} \mid x_t, y_{t+1:T}) = \frac{Z_{t+1} \gamma_t(x_t)}{Z_t \gamma_{t+1}(x_{t+1})} p(y_t \mid x_t) p(x_{t+1} \mid x_t) \widetilde{\mathsf{BIF}}_{t+1}(x_{t+1}).$$

By construction of the normalisation densities, we can integrate the previous expression over x_{t+1} to obtain

$$\widetilde{\mathsf{BIF}}_t(x_t) \propto \gamma_t(x_t) p(y_t \mid x_t) \int p(x_{t+1} \mid x_t) \frac{\widetilde{\mathsf{BIF}}_{t+1}(x_{t+1})}{\gamma_{t+1}(x_{t+1})} \, \mathrm{d}x_{t+1}.$$

Noting that $\mathsf{BIF}_T(x_T) = p(y_T \mid x_T)$ and iterating brings

$$\widetilde{\mathsf{BIF}}_t(x_t) = \int \gamma_t(x_t) \prod_{\ell=t}^{T-1} p(x_{\ell+1} \mid x_\ell) \prod_{k=t}^{T} p(y_k \mid x_k) \, \mathrm{d}x_{t+1:T}.$$

We can thus define a joint distribution $\widetilde{\mathsf{BIF}}_t(x_{t:T})$ as the integrand of the above expression with the following sequential structure:

$$\widetilde{\mathsf{BIF}}_t(x_{t:T}) \propto \frac{\gamma_t(x_t) p(x_{t+1} \mid x_t) p(y_t \mid x_t)}{\gamma_{t+1}(x_{t+1})} \widetilde{\mathsf{BIF}}_{t+1}(x_{t+1:T}), \tag{3.9}$$

which lends itself well to the SMC framework with the optimal proposal:

$$q_t^{\mathsf{opt}}(x_t \mid x_{t+1}) \quad \propto \quad \frac{\gamma_t(x_t)p(x_{t+1} \mid x_t)p(y_t \mid x_t)}{\gamma_{t+1}(x_{t+1})}. \tag{3.10}$$

3.1.3 Two filter smoothing algorithm

Plugging a particle representation of the filtering distribution in (3.3), the prediction density can be approximated by

$$\widehat{\mathsf{PD}}_t(x_t) \quad = \quad \sum_{i=1}^{N} w_{t-1}^{(i)} p(x_t \mid X_{t-1}^{(i)}). \tag{3.11}$$

Correspondingly, we can consider a particle representation of the normalised BIF obtained by following a backward particle filter with M particles on (3.9). Let us denote these by $\widehat{\widetilde{\mathsf{BIF}}}_t$ with weights $\tilde{w}_{t+1}^{(j)}$ and particles $\tilde{X}_{t+1}^{(j)}$, i.e.:

$$\widehat{\widetilde{\mathsf{BIF}}}_t(x_t) \quad = \quad \sum_{j=1}^{M} \tilde{w}_{t+1}^{(j)} \delta_{\tilde{X}_{t+1}^{(j)}}(x_t). \tag{3.12}$$

By dividing these by the corresponding normalisation density γ_t, we can obtain approximations to the original BIF:

$$\widehat{\mathsf{BIF}}_t(x_t) \quad = \quad \widehat{\widetilde{\mathsf{BIF}}}_t(x_t)/\gamma_t(x_t).$$

Combining both the estimators of the prediction density and the backward information filter at step t, we can obtain a particle representation of the smoothing distribution with

$$\hat{p}(x_t \mid y_{1:T}) \quad = \quad \zeta_t^{-1} \widehat{\mathsf{PD}}_t(x_t) \widehat{\mathsf{BIF}}_t(x_t), \tag{3.13}$$

where ζ_t is a normalisation constant. That is a mixture of NM terms and a further step may be desirable to reduce the number of components in the resulting representation of the smoothing distribution. In particular, if we take $M = N$, at each step we have to consider a mixture with a quadratic number of terms in N and this, independently of the choice of normalising densities meaning that the algorithm has inherently a quadratic complexity. We consider this choice in what follows.

3.2 Backward Information Smoothing

In this section, we explore the choice of normalisation densities and suggest using an approximation to the predictive density. It came to our knowledge that this choice had in fact also been studied independently and earlier than this work by Taghavi (2012). We show how the complexity of the resulting algorithm is quadratic in the number of particles. We also discuss two modified versions with sub-quadratic complexity. The first one, inspired by Fearnhead et al. (2010) and also discussed by Taghavi (2012), has linear complexity but the resulting estimators cannot be shown to converge. The second one, inspired by our paper (Lienart et al., 2015), has complexity $\mathcal{O}(NM)$ where M is sub-linear in N and leads to convergent estimators (in the sense of Chopin (2004)).

3.2.1 Choice of normalisation densities

The choice of normalisation densities in the TFS algorithm is, in theory, only constrained by the support condition (3.5). However, the quality of the corresponding estimators for a finite sample size depends significantly on it. Combining (3.1) and (3.9), we get

$$p(x_t \,|\, y_{t:T}) \quad \propto \quad \frac{\mathsf{PD}_t(x_t)}{\gamma_t(x_t)} \int \widetilde{\mathsf{BIF}}_t(x_{t:T}) \, \mathrm{d}x_{t+1:T}$$

so that we can write

$$p(x_{t:T} \mid y_{1:T}) \quad \propto \quad \frac{\mathsf{PD}_t(x_t)\widetilde{\mathsf{BIF}}_t(x_{t:T})}{\gamma_t(x_t)}. \tag{3.14}$$

This expression suggests picking γ_1 to be the prior distribution on the first state $\pi_0(x_1)$ since that leads directly to

$$p(x_{1:T} \mid y_{1:T}) \quad = \quad \widetilde{\mathsf{BIF}}_1(x_{1:T}). \tag{3.15}$$

The last two equations (3.14) and (3.15) are crucial for the rest of the analysis. The first one suggests that if we pick γ_t to be close to the predictive density then, then each term $\widetilde{\mathsf{BIF}}_t(x_{t:T})$ forms an estimator for the partial joint smoothing densities $p(x_{t:T} \mid y_{1:T})$; the second one indicates that upon selecting γ_1 to be the prior for the initial state, we end up targeting exactly the joint distribution of interest, no matter which admissible sequence $\{\gamma_t\}_{t=2}^T$ we picked earlier.

This suggests to build a good estimator of the prediction density, based partly or entirely on a particle estimator of the filtering density; use it as normalisation density and target the corresponding normalised BIF recursively. We explored this idea and showed that it significantly outperforms the default choice used in Briers et al. (2010); Fearnhead et al. (2010) where the authors also suggest taking γ_1 as the prior π_0 and build the subsequent γ_t by propagation through the transition dynamic of the HMM:

$$\gamma_t(x_t) \quad = \quad \int p(x_t \mid x_{t-1})\gamma_{t-1}(x_{t-1})\mathrm{d}x_{t-1} \tag{3.16}$$

Note that this choice also suffers from a tractability issue in a system with a transition that is more complex than the linear and Gaussian one considered in Fearnhead et al. (2010) where, in any case, the optimal Kalman smoother should be preferred. We show in the appendix (point A.1.1) how this normalising density can be obtained in the case

of a simple linear Gaussian dynamic.

Upon selecting γ_t to be the approximation of the predictive density (3.11) based entirely on a a particle estimator of the filtering density, we get what Taghavi calls the *backward informations smoothing* (BIS) algorithm (Taghavi, 2012). This choice verifies the support condition (3.5) provided the support of the transition density covers the support of the BIF. Typically, the support of the transition density is the whole space $\mathcal{X}$ which guarantees this. The optimal importance function for targeting the corresponding normalised BIF with this specific choice of normalising density ($\gamma_t(x_t) = \widehat{\mathsf{PD}}_t(x_t)$) is obtained by considering (3.9):

$$\tilde{q}_t^{\mathsf{opt}}(x_t \mid \tilde{X}_{t+1}^{(j)}) \propto \frac{\widehat{\mathsf{PD}}_t(x_t)}{\widehat{\mathsf{PD}}_{t+1}(\tilde{X}_{t+1}^{(j)})} p(\tilde{X}_{t+1}^{(j)} \mid x_t) p(y_t \mid x_t), \tag{3.17}$$

where the $\{\tilde{X}_{t+1}^{(j)}\}_{j=1}^N$ are the particles from the previous smoothing step. These steps are illustrated in algorithm 6.

Algorithm 6 *Backward information smoother with quadratic complexity*

1: run a PF targeting $\{p(x_t \mid y_{1:t})\}_{t=1}^T$ with N particles $\{X_t^{(i)}, w_t^{(i)}\}_{t,i=1}^{T,N}$
2: let $\{\tilde{X}_T^{(j)}, \tilde{w}_T^j\}$ be the result of resampling (if required) the final PF particles
3: **for** $t = T-1 : 2$ **do**
4: sample $\tilde{X}_t^{(j)}$ from $\tilde{q}_t(\cdot \mid \tilde{X}_{t+1}^{(j)})$ with $\tilde{q}_t \approx \tilde{q}_t^{\mathsf{opt}}$ for $j = 1, \dots, N$
5: update and normalise the weights $\tilde{w}_t^{(j)} \propto \tilde{\alpha}_t^{(j)} \tilde{w}_{t+1}^{(j)}$
6: resample the particles if necessary
7: **end for**
8: same operations for $t = 1$ but with $\widehat{\mathsf{PD}}_1 \equiv \pi_0$
9: **return** weighted set of particles $\{\tilde{X}_{1:T}^{(j)}, \tilde{w}_{1:T}^{(j)}\}$

At each step t and for each particle j, Algorithm 6 ideally requires sampling a particle from $\widehat{\mathsf{PD}}_t(x_t) p(\tilde{X}_{t+1}^{(j)} \mid x_t) p(y_t \mid x_t)$ which is a mixture of N terms. Additionally, computing the updating factors for the weights requires computing $\widehat{\mathsf{PD}}_{t+1}(\tilde{X}_{t+1}^{(j)})$ which is also a sum of N terms. Provided we use N particles to target the BIF, the complexity of the BIS algorithm is therefore inherently quadratic in the number of particles. This

is the same computational complexity as that of the FFBS algorithm.

As it is not trivial in general to sample from $p(\tilde{X}_{t+1}^{(j)} \mid x_t)p(y_t \mid x_t)$, let alone when it is combined with $\widehat{\mathsf{PD}}_t(x_t)$, a possibility akin to the bootstrap proposal for the particle filter is to simply sample from $\widehat{\mathsf{PD}}_t(x_t)$. The disadvantage is that this choice does not take into account all the information available (contained in $\tilde{X}_t^{(j)}$ and y_t). Assuming we can sample easily from the transition distribution, sampling from $\widehat{\mathsf{PD}}_t$ can be done easily in two steps: first sample an index $i^\star \sim \mathcal{M}(w_{t-1}^{(1)}, \ldots, w_{t-1}^{(N)})$ where $\mathcal{M}$ denotes a multinomial distribution and the $w_{t-1}^{(i)}$ are the weights of the particle filter; and then sample from the corresponding term $p(x_t \mid X_{t-1}^{(i^\star)})$. The update factor is then given by

$$\tilde{\alpha}_t^{(j)} \quad = \quad \frac{p(\tilde{X}_{t+1}^{(j)} \mid \tilde{X}_t^{(j)})p(y_t \mid \tilde{X}_t^{(j)})}{\widehat{\mathsf{PD}}_{t+1}(\tilde{X}_{t+1}^{(j)})}, \tag{3.18}$$

which has linear complexity in the number of particles for each j so that, as expected, the *bootstrap backward information smoother* also has quadratic complexity.

3.2.2 BIS with linear complexity

We have shown that the BIF algorithm is inherently quadratic in the number of particles given the form of the optimal proposal (3.17). A statistically equivalent way of writing the optimal sampling step is to suggest sampling N particles from a mixture of N^2 terms:

$$\tilde{X}_t^{(j)} \quad \sim_{\text{iid}} \quad \tilde{q}_t^{\mathsf{mix}} \quad \propto \quad \sum_{i,j=1}^{N} \frac{w_{t-1}^{(i)} \tilde{w}_{t+1}^{(j)}}{s_{t+1}^{(j)}} p(x_t \mid X_{t-1}^{(i)}) p(y_t \mid x_t) p(\tilde{X}_{t+1}^{(j)} \mid x_t), \tag{3.19}$$

where $s_{t+1}^{(j)} = \sum_{k=1}^{n} w_t^{(k)} p(\tilde{X}_{t+1}^{(j)} \mid X_t^{(k)})$. Since this is equivalent to sampling from the optimal proposal, no reweighting step is needed, provided we can sample exactly from the mixture.

Sampling from such a mixture can be done in two steps: first, sample a pair of la-

bels $(i^\star, j^\star)$ from a multinomial distributions with N^2 weights $\beta_t^{(i,j)}$ corresponding to the weight of the component (i, j) relative to the whole mixture q_t^{mix}; second, sample from the component $(i^\star, j^\star)$. Of course this does not simplify the problem: we still have to sample N pairs of indices from a multinomial with N^2 pair which is still quadratic in N and, additionally, computing the mixture component weights $\beta_t^{(i,j)}$ is intractable in general.

An approximation suggested first by Briers et al. (2005) in the wider context of sampling from products of mixtures and exploited by Fearnhead et al. (2010); Taghavi (2012) in the context of the TFS algorithm is to approximate i and j independently by representing $\beta_t^{(i,j)} \approx \beta_t^{(i)} \tilde{\beta}_t^{(j)}$. The simplest form for this approximation, is therefore to use $\beta_t^{(i)} = w_{t-1}^{(i)}$ and $\tilde{\beta}_t^{(j)} = \tilde{w}_{t+1}^{(j)}$.

Since sampling from a density proportional to $p(x_t \mid X_{t-1}^{(i^\star)}) p(y_t \mid x_t) p(\tilde{X}_{t+1}^{(j^\star)} \mid x_t)$ may still be intractable, we can resort to importance sampling for that term as well and compute the corresponding importance weight $\tilde{w}_t^{(j)}$ as a result. With this approach, sampling the indices is now an operation with linear computational complexity in the number of particles and the resulting algorithm consequently also enjoys linear complexity. The steps are illustrated in algorithm 7.

Algorithm 7 *BIS with linear complexity*

1: run a particle filter targeting $\{p(x_t \mid y_{1:t})\}_{t=1}^T$ with N particles $\{X_t^{(i)}, w_t^{(i)}\}_{t,i=1}^{T,N}$
2: set $\{\tilde{X}_T^{(j)}, \tilde{w}_T^{(j)}\}_{j=1}^N \leftarrow \{X_T^{(i)}, w_T^{(i)}\}_{i=1}^N$
3: **for** $t = T-1 : 2$ **do**
4: sample N pairs of indices with $i_{1:N}^\star \sim_{\text{iid}} \mathcal{M}(\beta_t^{(i)})$ and $j_{1:N}^\star \sim_{\text{iid}} \mathcal{M}(\tilde{\beta}_t^{(j)})$
5: **for** $k = 1 : N$ **do**
6: sample $\tilde{X}_t^{(k)} \sim \tilde{q}_t \approx \tilde{q}_t^{\text{opt},i_k^\star,j_k^\star}(x_t) \propto p(x_t \mid X_{t-1}^{(i_k^\star)}) p(\tilde{X}_{t+1}^{(j_k^\star)} \mid x_t) p(y_t \mid x_t)$
7: compute the unnormalised weights $\tilde{w}_t^{(k)} \propto \tilde{q}_t^{\text{opt},i_k^\star,j_k^\star}(\tilde{X}_t^{(k)}) / \tilde{q}_t(\tilde{X}_t^{(k)})$
8: **end for**
9: normalise the weights (and resample if necessary)
10: **end for**
11: same operations for $t = 1$ but with $\tilde{q}_1^{\text{opt}}(x_1 \mid \tilde{X}_2^{(j)}) \propto \pi_0(x_1) p(\tilde{X}_2^{(j)} \mid x_1) p(y_1 \mid x_1)$
12: **return** weighted set of particles $\{\tilde{X}_{1:T}^{(j)}, \tilde{w}_{1:T}^{(j)}\}_{j=1}^N$

Note that, in algorithm 7, a resampling step can also be introduced if the ESS drops below a fixed admissible threshold as with the particle filtering algorithm. Note also that the approximation inspired by Briers et al. (2005) used in this algorithm is what leads to an algorithm with linear complexity but is biased and results in estimators that do not enjoy the CLT property of standard SMC estimators.

3.2.3 Convergent BIS with sub-quadratic complexity

We discussed earlier the bootstrap BIS with quadratic complexity where one samples from the approximation to the predictive density $\widehat{\mathrm{PD}}_t$. This can in fact be simplified by compressing the representation of the particle filter from N particles to M particles by resampling. Taking M to be sub-linear in N (e.g.: $M = \mathcal{O}(\log N)$), the overall algorithm is $\mathcal{O}(TMN)$ and sub-quadratic in N.

Algorithm 8 *Bootstrap BIS with sub-quadratic complexity*

1: run a PF targeting $\{p(x_t \mid y_{1:t})\}_{t=1}^T$ with N particles $\{X_t^{(i)}, w_t^{(i)}\}_{t,i=1}^{T,N}$
2: sample $r_{1:M}^{(T-1)}$ from $\mathcal{M}(\{w_{T-1}^{(i)}\}_{i=1}^N)$
3: **for** $t = T-1:2$ **do**
4: sample $r_{1:M}^{(t-1)}$ from $\mathcal{M}(\{w_{t-1}^{(i)}\}_{i=1}^N)$
5: sample $\tilde{X}_t^{(j)}$ from $p(\cdot \mid X_{t-1}^{(r_j^{(t-1)})})$ for $j = 1, \ldots, M$
6: compute the update factors

$$\tilde{\alpha}_t^{(j)} = \frac{p(\tilde{X}_{t+1}^{(j)} \mid \tilde{X}_t^{(j)}) p(y_t \mid \tilde{X}_t^{(j)})}{\sum_{\ell=1}^M w_t^{(r_\ell^t)} p(\tilde{X}_{t+1}^{(j)} \mid X_t^{(r_\ell^t)})}$$

7: update and normalise the weights (and resample if required)
8: **end for**
9: same operations for $t = 1$ but sampling $X_{1:N}^{(1)}$ from π_0 (or recycling from the particle filter)
10: **return** weighted set of particles $\{\tilde{X}_{1:T}^{(j)}, \tilde{w}_{1:T}^{(j)}\}$

Note that in the case where one can sample easily from the backward dynamic of the HMM i.e.: sample from $p(x_t \mid x_{t+1})$ this bootstrap BIS can be reversed and ran forward as well as backward. This is in fact a core element of the idea behind our *expected particle belief propagation* (EPBP) algorithm to target the beliefs on an arbitrary MRF

which we discuss in the second part of this thesis (see chapter 7).

3.2.4 Sub-quadratic FFBS

A similar idea than the one applied at the previous point can be applied in the context of the FFBS algorithm (point 2.2.3). In the FFBS algorithm, a particle filter is run forwards and the weights are then updated in a backward step:

$$w_{t\,|\,T}^{(i)} \propto w_t^{(i)} \sum_{j=1}^{N} w_{t+1\,|\,T}^{(j)} \left[\frac{p(X_{t+1}^{(j)} \,|\, X_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(X_{t+1}^{(j)} \,|\, X_t^{(k)})} \right]. \tag{3.20}$$

We had indicated that the complexity of this algorithm is quadratic in N since we need to consider all the pairwise interactions $p(X_{t+1}^{(j)} \,|\, X_t^{(i)})$. However, we can sample a vector r of M indices following a multinomial with weights $\{w_{t+1\,|\,T}^{(j)}\}_{j=1}^{N}$ and thereafter modify the weight update (3.20) to

$$w_{t\,|\,T}^{(i)} \propto w_t^{(i)} \sum_{j=1}^{M} w_{t+1\,|\,T}^{(r_j)} \left[\frac{p(X_{t+1}^{(r_j)} \,|\, X_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(X_{t+1}^{(r_j)} \,|\, X_t^{(k)})} \right] \tag{3.21}$$

where M is sub-linear in N. The complexity of the corresponding algorithm is then $\mathcal{O}(MN)$. We show in the experimental section that this algorithm compares favourably with the full complexity FFBS.

3.3 Experiments

In this section we discuss the performances of the different smoothing algorithms in different situations. In particular, we show that the default choice of normalisation density (3.16) suggested in (Briers et al., 2010; Fearnhead et al., 2010) can underperform significantly compared to the BIS. It suffices to look at a simple linear-Gaussian model inspired from (Fearnhead et al., 2010) to show this.

We also show that the linear-complexity implementation of the bootstrap BIS compares very favourably with the sub-quadratic and the quadratic implementation at a much reduced computational cost. We then go on to compare the performances of the sub-quadratic smoothing algorithms when the underlying dynamic is non-linear.

The key aim for this section is to present the simplest possible models in which a clear difference in performance among the algorithms is shown. The metric for accuracy that is used is the RMSE, as in the main literature in the field (see e.g.: Arulampalam et al. (2002)). This metric is known to be flawed since Bayesian inference does not try to optimise the RMSE but since, in this chapter, we are comparing Bayesian methods, it already provides a good indicator of performance and it is easy to compute.

3.3.1 Linear Gaussian models

Model 1 (low dimensionality)

The HMM dynamic considered is inspired from (Fearnhead et al., 2010): a simple linear-Gaussian dynamic with a two-dimensional latent space and a one-dimensional observation space. In their article, the authors discuss how the ratio of the amplitude of the noise in the state dynamic to that of the observation dynamic impacts the performances of their algorithm. We reproduce two simple cases where we show that the choice of normalising density suggested in (Briers et al., 2010; Fearnhead et al., 2010) underperforms that of the other smoothers. The dynamic is defined as follows:

$$\begin{cases} x_{t+1} &= Ax_t + \epsilon_t \\ y_{t+1} &= Bx_{t+1} + \eta_{t+1} \end{cases} \tag{3.22}$$

where $\epsilon_t \sim \mathcal{N}(0, Q)$ and $\eta_t \sim \mathcal{N}(0, R)$. The matrices are defined as follows:

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad Q = \frac{\nu}{6}\begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}, \quad R = \tau \tag{3.23}$$

where $\nu, \tau > 0$ define the amplitude of the noise processes. We compare two cases, the first one (case A) has $\nu = 10$ and $\tau = 1$ and the second one (case B) has the opposite assignment $\nu = 1$ and $\tau = 10$. Case A corresponds to a case where Fearnhead et al. (2010) indicate that the state has freedom to follow the observations which helps smoothing algorithms. Case B corresponds to a case where Fearnhead et al. (2010) indicate that the states are highly dependent through time which can cause smoothers to struggle.

We run the dynamic for $T = 50$ steps with $N = 200$ particles and run the experiments $J = 50$ times for each algorithm (the data is randomised each time). We compare the average RMSE across the time steps computed as:

$$E_j^a = \frac{1}{K}\sum_{k=1}^{K} \left\| x_{j,k} - \hat{\mu}_{j,k}^a \right\|_2 \tag{3.24}$$

where j indicates the experiment run ($j = 1, \ldots, J$), the superscript a indicates the algorithm, x_j denotes the ground truth for the jth experiment and $\hat{\mu}_j^a$ the recovered mean from the algorithm. This is a crude measure of quality but is enough to show how the different algorithms compare (and, in particular, that the BIS outperforms Fearnhead's algorithm); Arulampalam et al. (2002) in their highly cited paper mention that the RMSE is widely used in the literature and facilitates quantitative comparison.

In the figures below, *KF* and *KS* refer to the Kalman filter and smoother, *PF* to a bootstrap particle filter, *FFBS* to the forward filtering backward smoothing algorithm applied after the PF and *FFBS*$_S$ to the sub-quadratic implementation (with $M = 25$). The *BIS* symbols refer to the (bootstrap) backward information smoothers and the

subscript indicates the version: Q for quadratic, L for linear, S for sub-quadratic (with $M = 25$). Finally, FH refers to Fearnhead's algorithm (the full description of Fearnhead's algorithm for this model is available in the appendix A.1). The main takeaways of the results presented here is to show that:

1. the choice of normalisation density taken in Fearnhead's algorithm can lead to results that significantly underperform compared to other methods,
2. sub-quadratic methods do not significantly underperform as compared to the quadratic methods.

In figure 3.1 it is already clear that the FH algorithm underperforms. It is also interesting to observe that all three BIS methods offer comparable performances while the BIS_L is computationally the least expensive. The FFBS algorithm performs quite well as well which can be expected as the dynamic is simple and hence the FFBS results does not suffer too much from the lack of resampling in the backward stage.

In figure 3.2 the FH algorithm significantly underperforms compared to all other algorithms considered. The other algorithms behave in a way that is comparable to our discussion of case A with, again, the BIS_L offering good performances while being the least computationally expensive of the particle methods considered. We can also observe that both sub-quadratic implementations (FFBS_S and BIS_L) seem to suffer a bit with performances that are a bit more spread out.

Model 2 (medium dimensionality)

In this second example, we consider a model with a 10-dimensional latent space and a 6-dimensional observational space. The matrix A is defined as the identity plus a perturbation matrix E with elements $e_{ij} = \eta_{ij}/5$ with η_{ij} drawn from a standard Gaussian. The matrix B has elements b_{ij} drawn from a standard Gaussian. Both matrices A and B are normalised to prevent an explosive behaviour from the HMM. The matrix Q

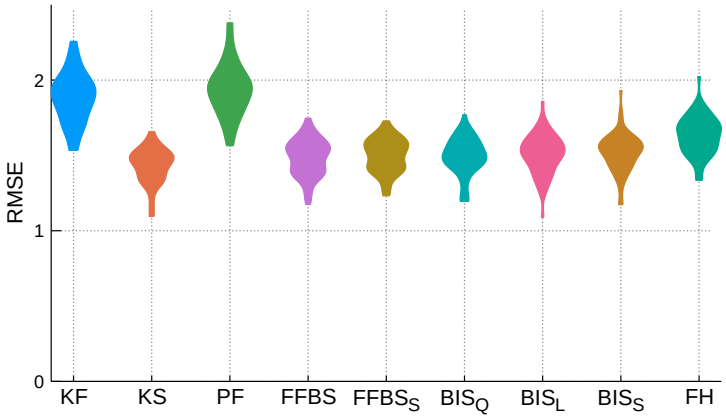


Figure 3.1: *Violin plots comparing the RMSE of the algorithms in* case A *over 50 runs with the optimal Kalman smoother as reference. The FFBS and all BIS have comparable performances while the FH underperforms with generally higher RMSE.*

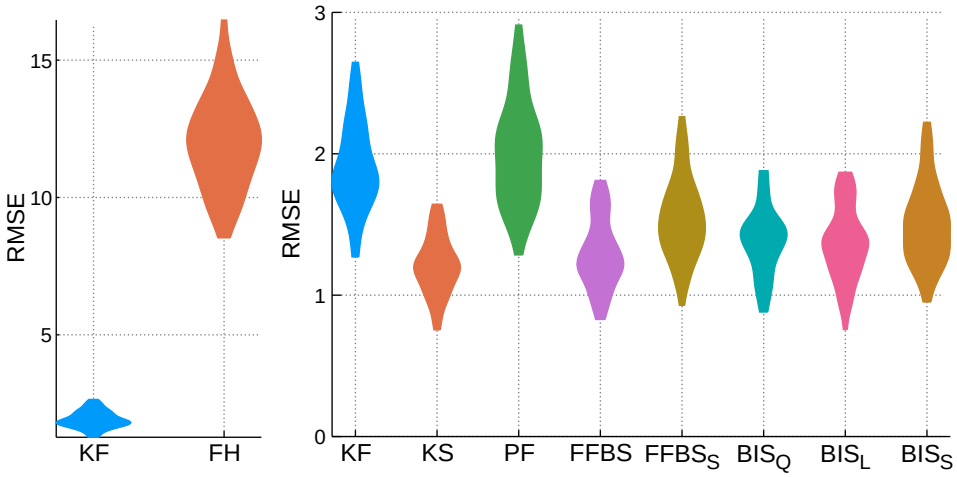


Figure 3.2: *Violin plots comparing the RMSE of the algorithms in* case B *over 50 runs. (**left**) The FH significantly underperforms compared to the other algorithms as can be seen compared to the KF which is reproduced on both graph to give an idea of scale (FH is not reproduced on the right as the scale is significantly different from the other algorithms). (**right**) the other smoothing algorithms offer performances in line with the observations made in case A and do not suffer as much from the decreased noise ratio unlike the FH.*

and R have random elements drawn in a similar fashion and both matrices are made positive definite. Both Q and R are normalised and scaled such that $\|Q\| = 1/5$ and $\|R\| = 1/20$. We denote this model LG-10-6 for further reference.

The point of this example is to show that when the particle filter struggles (with

a lot of particles with very low weight or, correspondingly, a low ESS) then naturally both versions of the FFBS struggle. However, the algorithms which apply a resampling of the representation of the particle filter (BIS_L and BIS_S) perform very well. Indeed, this resampling guides the smoothing densities to regions of higher expected density (through more concentrated normalising densities γ_t) resulting in better performance overall. The performances are illustrated at figure 3.3.

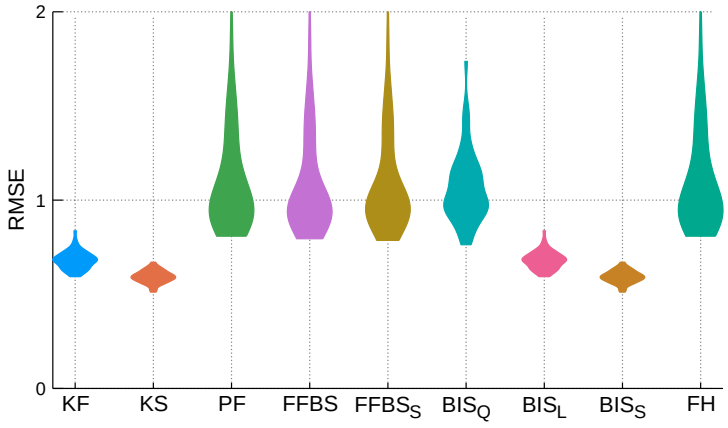


Figure 3.3: *Violin plots comparing the RMSE of the algorithms for model LG-10-6 over 50 runs. In this case, the sub-quadratic implementations of the BIS perform outstandingly well compared to the other algorithms which offer little performance gain over simply considering the particle filter. Note that the flat base for some of the violin plots is an artefact of the plotting toolbox used.*

3.3.2 Nonlinear Gaussian models

In this point, we show how the sub-quadratic algorithms perform when the dynamic of the HMM is not linear but defined as:

$$\begin{cases} x_{t+1} &= f_t(x_t) + \epsilon_t \\ y_{t+1} &= g_{t+1}(x_{t+1}) + \eta_{t+1} \end{cases}. \tag{3.25}$$

for some sets of functions $\{f_t\}_{t=1}^T$ and $\{g_t\}_{t=1}^T$ and with innovation processes ϵ_t and η_t drawn from centred multivariate Gaussians with covariance matrices Q and R.

Model 3

This model is drawn from the paper of Arulampalam et al. (2002) with the dimensionality of both the latent space and the observational space equal to one:

$$\begin{cases} f_t(x) &= x/2 + 25x(1+x^2)^{-1} + 8\cos(1.2t) \\ g_t(x) &= x^2/20 \end{cases} \quad \text{and } Q = 10,\ R = 1. \tag{3.26}$$

We call this model NL-1-1 for further reference. We use $N = 50$ particles and for $K = 100$ time steps (as in the original paper) and run the experiments 50 times with $M = 15$ for the sub-quadratic algorithms.

Figure 3.4 shows the results of the comparison of the RMSE for each algorithm considered. No smoothing algorithm particularly stands out but it shows that the sub-quadratic smoothing algorithms considered do not significantly underperform compared to the full quadratic complexity ones even when the underlying dynamic is non-linear.

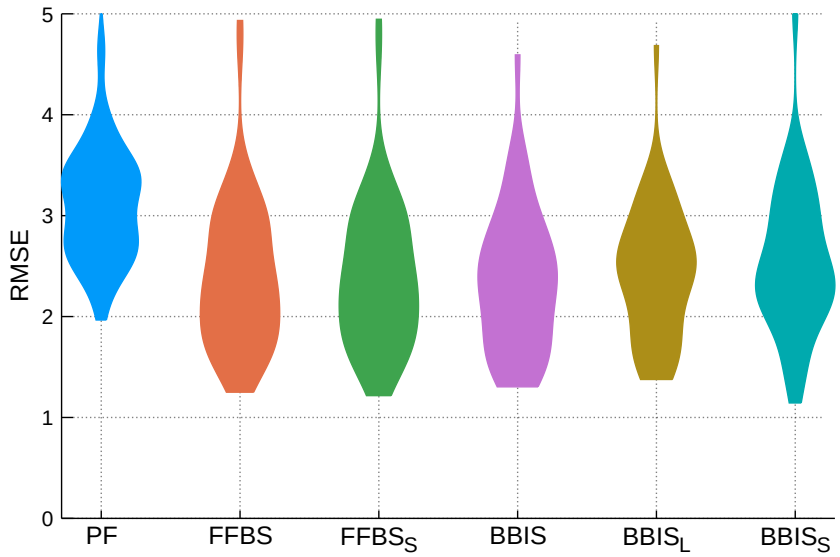


Figure 3.4: *Violin plots comparing the RMSE of the algorithms for model NL-1-1 over 50 runs. In this case, all smoothing algorithms considered offer similar performances showing that sub-quadratic smoothing algorithms can also perform well in the case of a non-linear model.*

3.4 Discussion

In this chapter we suggested and discussed three algorithms with sub-quadratic complexity. Two are approximations to the backward information smoother (BIS_L, BIS_S) using the two filter smoothing algorithm while taking the normalising densities to be approximations of the predictive densities based on a particle filter. The last one (FFBS_S) is an approximation to the forward filtering backward smoothing algorithm.

A criticism of the FFBS is that the smoothing step does not sample new particles and just reweights existing particles which can harm the performance of the resulting estimator. In our experiments however, the FFBS and, more interestingly, its sub-quadratic approximation typically perform well compared to other algorithms.

The BIS does perform a resampling step in the backward step and uses normalisation densities that are more sensible and typically outperform the default choice suggested in (Briers et al., 2010; Fearnhead et al., 2010). However, the optimal backward sampling step is intractable in general and one may have to resort to the bootstrap-BIS as we have in the experiments thereby potentially reducing the advantage of this additional sampling step.

We showed in our experiments that the BIS_L and the BIS_S typically perform on par with the naive implementation. There may be cases in which those algorithms do not perform well but based on the results of the experiments, we would encourage a user to consider the BIS_L as a first smoothing algorithm (provided the dynamic is not linear).

We indicated that the estimators obtained through the BIS_L do not enjoy a CLT due to the biased approximation in the sampling of the mixture indices. This, however, seemed to have little impact on the results of the experiments. A possible explanation for this is that, at the regime considered with relatively few particles, all estimators are biased and there is too much variance to significantly distinguish between them.

Note however that exploiting these algorithms with significantly more particles (a few orders of magnitude more) requires having to deal with numerical instabilities arising from the representation of the range of weights associated with a large number of particles, and this, even with resampling. This is a known issue, particularly when the dimensionality of the problem is higher and some techniques exist to combat the issue (Miguez and Eugenia, 2013).

Finally, we have only considered standard multinomial resampling in our algorithms. It may be that alternative resampling procedures such as the ones mentioned in Hol et al. (2006) lead to improved performances. This could be an avenue for further work as well as a more in depth quantification of the computational performance considering more adequate error metrics such as the log-likelihood on a held out set.

4 | Bouncy Particle Sampler on Markov Random Fields

In this chapter, we cover the Local Bouncy Particle Sampler (LBPS), a version of the BPS introduced in chapter 2, section 2.3 for target distributions that factorise according to a MRF.

The aim of this short chapter is to show that the LBPS algorithm can be used on high-dimensional models such as probabilistic matrix factorisation and performances are not far those of the HMC algorithm thereby showing the potential of PDMP samplers on complex Machine Learning models. For this, we used our open-source package `PDSampler.jl` coded in Julia.[1]

[1] https://github.com/alan-turing-institute/PDSampler.jl

4.1 Local Bouncy Particle Sampler

In Bouchard-Côté et al. (2015), the authors consider the case where the target distribution factorises as

$$\pi(x) \propto \prod_{f \in F} \gamma_f(x_f) \tag{4.1}$$

where x_f is a restriction to a few elements of x, F is a set of *factors* and γ_f is an unnormalised likelihood associated with the factor. In the specific case of a pairwise MRF, the factors are the edges of the graph and the restrictions are the variables corresponding to each nodes. The energy associated to π consequently decomposes as $U \equiv \sum_{f \in F} U_f$.

For each factor, a local intensity λ_f and a local bouncing operator R_f can be defined in the same way as for the Bouncy Particle Sampler (BPS, see section 2.3) except that ∇U_f is set to have zero components for all variables not associated with the factor. We can then define a collection of intensities with

$$\chi_f(t) = \lambda_f(x^{(i-1)} + v^{(i-1)}t, v^{(i-1)}). \tag{4.2}$$

and consider the superposition principle with $\chi \equiv \sum_f \chi_f$ (see point 2.3).

Instead of modifying all velocity variables at a bounce as in the basic BPS, the method samples a factor f with probability $\chi_f(\tau)/\chi(\tau)$ and modifies only the variables connected to the sampled factor. This can significantly reduce the overall computational cost associated with the algorithm and especially so when the underlying MRF has a connection structure that is not too densely connected. Indeed, when an update is triggered at a factor f all factors that share a variable with f are also triggered. If that corresponds to a large portion of the graph, computational gains are

lost compared to simply using the full BPS. The skeleton of the algorithm is shown below.

Algorithm 9 *Local BPS with Priority Queue*

initialise $(x^{(0)}, v^{(0)})$, set $t_{\text{clock}} = 0$
simulate a first arrival time $\tau^f_{\text{bounce}} \sim \text{PP}(\chi_f(t))$ for each factor f
initialise a priority queue Q with the couples $\{(\tau_f, f)\}_{f \in F}$
initialise event lists L_f for each factor with $(x_f^{(0)}, v_f^{(0)}, 0)$
sample $\tau_{\text{ref}} \sim \text{Exp}(\lambda_{\text{ref}})$
while more events requested **do**
    pop (τ_f, f) from Q based on the smallest bounce time
    if $\tau_f > \tau_{\text{ref}}$ **then**
        $t_{\text{clock}} \leftarrow t_{\text{clock}} + \tau_{\text{ref}}$
        sample a new $v \sim \mathcal{N}(0, \mathbb{I})$
        start a new queue Q where the positions for each factor is extrapolated linearly until the refreshment time
    else
        $t_{\text{clock}} \leftarrow t_{\text{clock}} + \tau_f$
        extrapolate x_f linearly until t_{clock}
        add x_f to the list L_f
        for all neighbouring factors f' **do**
            extrapolate x'_f linearly until t_{clock}
            simulate the first arrival time $\tau_{f'}$ of a PP with intensity $\lambda_{f'}(x_{f'} + tv_{f'}, v_{f'})$
            update Q with these candidate bounce times
        end for
    end if
end while

4.2 Probabilistic Matrix Factorisation

Probabilistic Matrix Factorisation (PMF) (Salakhutdinov and Mnih, 2008) is a Bayesian approach to the matrix completion problem. In that problem, we consider a matrix R of size $n \times p$ of ratings r_{ij} corresponding to user i and item j (e.g.: a movie) but we only have access to a mask of R, a very sparse subset of the ratings which we denote $\hat{R}$. The aim of matrix completion methods such as PMF is to try to construct a low-rank factorisation $R \approx U^t V$ based on the known entries, where U is of size $d \times n$ and V of size $d \times p$ and d is very small compared to n or p.

4.2.1 Description of the Model

The model described in Salakhutdinov and Mnih (2008) assumes that each rating r_{ij} is a realisation from a Normal random variable with mean $\langle u_i, v_j \rangle$ and standard deviation σ_r. Further, the model assumes spherical Gaussian priors on all u_i and v_j with standard deviation σ_u and σ_v respectively. Formally:

$$\begin{aligned} r_{ij} &\sim \mathcal{N}(\cdot\,; \langle u_i, v_j \rangle, \sigma_r^2), \qquad (i,j) \in \mathcal{M} \\ u_i &\sim \mathcal{N}(\cdot\,; 0_d, \sigma_u^2 \mathbb{I}_d), \qquad i \in 1, \ldots, n \\ v_j &\sim \mathcal{N}(\cdot\,; 0_d, \sigma_v^2 \mathbb{I}_d), \qquad j \in 1, \ldots, p \end{aligned} \tag{4.3}$$

where $\mathcal{M}$ denotes the entries which are available, 0_d and $\mathbb{I}_d$ denote the zero vector in $\mathbb{R}^d$ and the $d \times d$ identity matrix respectively. The negative log-posterior of interest is therefore:

$$-\log p(U, V|\hat{R}) = \frac{1}{2\sigma_r^2} \sum_{(i,j)\in\mathcal{M}} (r_{ij} - \langle u_i, v_j \rangle)^2 + \frac{1}{2\sigma_u^2} \|U\|_2^2 + \frac{1}{2\sigma_v^2} \|V\|_2^2 \tag{4.4}$$

In our experiments, we assume σ_r, σ_u and σ_v are fixed,[2] more involved models exist including hyper-priors which we will not consider here since our aim is primarily to compare the local BPS and HMC on a large scale model rather than produce the best performing PMF method.[3]

Finally, note that the full dimensionality of the model is $n \times d + p \times d$. Below we consider an example where $d = 10, n = 6000, p = 4000$ and $|\mathcal{M}| = 10^6$ meaning that the full dimensionality of the model is 100000 with 1 million factors. For that kind of

[2] In practice, we compute sensible estimates from the data following the authors' original code available at `http://www.cs.toronto.edu/~rsalakhu/BPMF.html`.

[3] In Salakhutdinov and Mnih (2008), the authors also suggest performing a few steps of Gibbs sampling for the hyperparameters. We do not do this in order to simplify the computations and the comparisons.

scale, a method such as naive Gibbs sampling (see point 2.1.3) is impractical though methods such as block Gibbs could be deployed (see e.g. Geyer (2005, section 2.5)).

4.2.2 Local BPS for the PMF

The factor graph corresponding to the matrix factorisation problem has a simple structure: each rating r_{ij} for $(i, j) \in \mathcal{M}$ corresponds to a factor connected to the variables u_i and v_j. This is illustrated in figure 4.1 below.

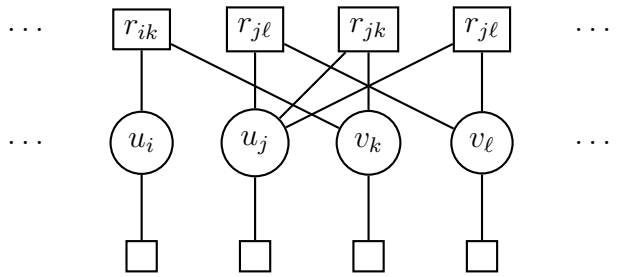


Figure 4.1: *Illustration of the factor-graph corresponding to the matrix completion problem. Each observed rating corresponds to a factor connected to a corresponding u-node and v-node themselves corresponding to d-dimensional variables. Each node also has its own factor corresponding to a spherical Gaussian prior (empty squares).*

The energy associated with one of the rating factor is also obtained by taking the negative log-likelihood associated to r_{ij} (see (4.3)), we denote it $\mathcal{E}_{ij}$ with

$$\mathcal{E}_{ij}(u_i, v_j) \quad := \quad 0.5\sigma_r^{-2}(r_{ij} - \langle u_i, v_j \rangle)^2. \tag{4.5}$$

Correspondingly, the prior-factors are associated with $\mathcal{E}_i^u(u_i) := 0.5\sigma_u^{-2}\|u_i\|_2^2$ and $\mathcal{E}_j^v(v_j) := 0.5\sigma_v^{-2}\|v_j\|_2^2$. The Inhomogeneous Poisson Processes (IPP) associated with Gaussians can easily be sampled from using the inversion method. The IPP associated with the rating factors can also be easily sampled from using the inversion method which we show below.

Sampling the IPP associated with a rating factor

To simplify notation, we consider a single rating r associated with two d-dimensional vectors u and v. We introduce the notation $x := (u; v)$ and write $x_u = u$ and $x_v = v$. We also write $e(x) := (\langle x_u, x_v \rangle - r)$. With these notations, we can write the energy as $\mathcal{E}(x) = 0.5\sigma_r^{-2} e^2(x)$. As for the BPS, the intensity associated with this factor is $\chi(t) = \langle \nabla \mathcal{E}(x + tw), w \rangle^+$ where w is a velocity-vector of the same dimension as x. The gradient with respect to the u and v component can be written as

$$\begin{cases} \nabla_{x_u} \mathcal{E}(x) &= \gamma(x) x_v \\ \nabla_{x_v} \mathcal{E}(x) &= \gamma(x) x_u \end{cases} \tag{4.6}$$

where $\gamma(x) := \sigma_r^{-2} e(x)$. It is straightforward to show that the inner product $\langle \nabla \mathcal{E}(x + tw), w \rangle$ is a third-order polynomial in t with

$$\langle \nabla \mathcal{E}(x + tw), w \rangle = \gamma(x + tw) \left[\langle x_u, w_v \rangle + \langle x_v, w_u \rangle + 2t \langle w_u, w_v \rangle \right] \tag{4.7}$$

The intensity $\chi(t)$ of the IPP is therefore given by the positive part of this third-order polynomial. The factor of t^3 in (4.7) is positive which is already indicative of the shape of the polynomial but the roots of the polynomial need to be studied in order to fully characterise it. It is easy to obtain the root corresponding to the left-parenthesis in (4.7) which we denote t_0 with $t_0 = -d(x, w)/2s(w)$ where $d(x, w) := (\langle x_u, w_v \rangle + \langle x_v, w_u \rangle)$ and $s(w) := \langle w_u, w_v \rangle$. The other two roots are obtained by setting $\gamma(x + tw)$ or equivalently $e(x + tw)$ to zero. All the roots of (4.7) are therefore given by:

$$\begin{cases} t_0 &= -d(x, w)/2s(w) \\ t_{-,+} &= t_0 \pm \sqrt{t_0^2 + e(x)/s(w)} \end{cases} . \tag{4.8}$$

Using the inversion method, sampling from the IPP can be done in two steps: generate a $\lambda \sim \mathrm{Exp}(1)$ and then find $t(\lambda)$ such that

$$\lambda \quad = \quad \Xi(t(\lambda)) \quad = \quad \int_0^{t(\lambda)} \chi(s)\,\mathrm{d}s. \tag{4.9}$$

Since $\chi(t)$ is the positive part of a third-order polynomial, the integral can be computed exactly and the inversion to obtain $t(\lambda)$ amounts to finding a root of a fourth-order polynomial which can be done efficiently using a numerical solver. The situation is illustrated in a case where $t_- > 0$ in figure 4.2 below.

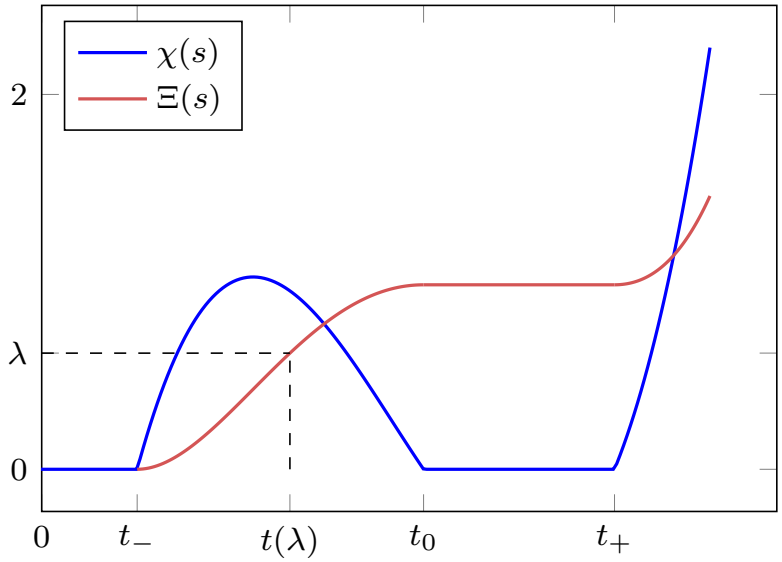


Figure 4.2: *The intensity χ (blue) is the positive part of a third-order polynomial, its integral Ξ (red) is represented and is a piecewise fourth-order polynomial. For λ drawn from an $\mathrm{Exp}(1)$, we can find $t(\lambda)$ corresponding to the first arrival time of interest. (Best viewed in colours.)*

4.2.3 Other algorithms considered

Singular Value Decomposition

As a baseline, we consider the Singular Value Decomposition (SVD) algorithm. Indeed, Eckart and Young's theorem shows that for a matrix M, the truncated SVD is the best

low-rank approximation in the sense of the Frobenius norm. In the matrix completion case, no guarantees are offered by the SVD since we only have access to a sparse mask of the real matrix (Srebro et al., 2004). Let R denote the true matrix we are trying to reconstruct and $\hat{R}$ the sparse matrix containing only the observed ratings (centred around zero). The SVD decomposition of $\hat{R}$ gives $\hat{R} = U\Sigma V^t$ where U is an orthonormal matrix of size $n \times p$, Σ is a diagonal matrix of size $p \times p$ and V is an orthogonal matrix of size $p \times p$. Truncating Σ to its first $d \ll p$ components leads to a rank-d approximation $\tilde{R}$ to $\hat{R}$ which will be much less sparse than $\hat{R}$. Letting $U_{\text{SVD}} := \Sigma^{1/2}U^t$ and $V_{\text{SVD}} := \Sigma^{1/2}V^t$ we have a first baseline factorisation $\tilde{R} = U_{\text{SVD}}^t V_{\text{SVD}} \approx R$.

Note that the problem of computing a truncated SVD decomposition of a large sparse system is a well known problem in numerical linear algebra and very efficient methods exist (Sorensen, 1996).[4]

Hamiltonian Monte Carlo

We had already shown that the negative log posterior in U and V of the model can be directly obtained:

$$-2\log p(U, V|\hat{R}) \quad = \quad \sigma_r^{-2} \sum_{(i,j)\in\mathcal{M}} (r_{ij} - \langle u_i, v_j \rangle)^2 + \sigma_u^{-2}\|U\|_2^2 + \sigma_v^{-2}\|V\|_2^2.$$

Therefore, the gradient in u_i and v_j can easily be computed in closed form and it is straightforward to apply HMC on this model (cf. algorithm 3). The main hurdle is that each gradient evaluation requires a full pass over all observed ratings making iterations computationally expensive.

[4] In the experiments, we use the `svds` function from Julia 0.6 uses the implicitly restarted Lanczos iterations.

4.2.4 Experiments

In our experiments, we consider the MovieLens 1M dataset.[5] corresponding to 1 million ratings of 4000 movies by 6000 users. We preprocess the ratings to centre them around zero and scaled on $[-1, 1]$ and take a random subset of 95% of the ratings as training set using the remaining 5% as test set. Following the literature, we use $d = 30$; we set $\sigma_R = 1$ and $\sigma_U = \sigma_V = 10$. The starting point is drawn from a standard multivariate Normal centred at the solution of the SVD algorithm. The Local BPS experiments use the `PDSampler.jl` package, the HMC simulations use 2 leapfrog steps, the stepsize is set to 0.01.[6] The reported value is the RMSE over the test set. We run each experiments ten times to account for the stochasticity of the algorithms. All results were obtained using Julia 0.6 with a 2.3GHz Intel Core i5 computer.

The results displayed in figure 4.3 seem to show that the LBPS does a little bit better in this case than the HMC algorithm. The improvement is not very significant but confirms that the BPS and the LBPS can perform as well or better than the HMC algorithm (a similar result had been obtained by (Bouchard-Côté et al., 2015) on a much simpler factor graph).

Both methods underperform compared to the test-RMSE obtained with the sparse SVD. The test-RMSE is computed as

$$\text{RMSE} \quad = \quad \sqrt{\sum_{(i,j)\in\mathcal{T}} (r_{ij} - \langle u_i, v_j \rangle)^2 / (np)} \tag{4.10}$$

where $\mathcal{T}$ denotes the indices corresponding to the test set and $\langle u_i, v_j \rangle$ is clipped if it is outside the range of possible values for the ratings.

[5] https://grouplens.org/datasets/movielens/

[6] Note that 2 leapfrog steps should not be expected to give excellent results though here we picked this in order to get reasonable computational times for the experiments.

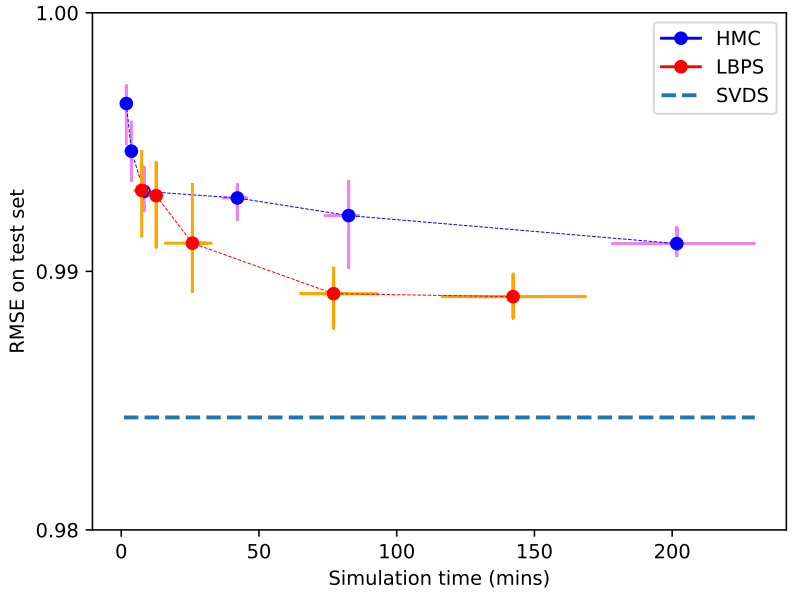


Figure 4.3: *RMSE on the test set of the MovieLens 1M dataset when using Probabilistic Matrix Factorisation with HMC and the LBPS algorithm compared to the result obtained with the Sparse SVD algorithm. The crosses indicate the range of values obtained and ranges of times for each of the 10 runs recorded for different simulation lengths. The dashed lines connects the means.*

The choice of hyper parameters may be suboptimal though after trying a range of possible hyper parameters we did not see major differences. In this case it may just be that the SVDS algorithm performs very well and does not overfit the data rendering the regularisation that appears in the PMF model useless.

In (Salakhutdinov and Mnih, 2008), the authors obtain results with the Bayesian PMF that outperform the SVD significantly but it is unclear what SVD algorithm they use and whether they use it on appropriately re-scaled data. Further they do not discuss the time taken by the algorithms, in our experiments, the SVDS runs several orders of magnitude faster than the LBPS or HMC algorithm though, of course, the SVD does not provide uncertainty estimates. Further, their code does not show the SVD step but suggests a scaling of the ratings on the $[0, 1]$ range for the Bayesian PMF. If the SVD algorithm is also applied on such ratings, it should be expected to perform worse as it

will not be able to exploit the sparsity of the data well.

4.3 Discussion

The main purpose of this chapter was to compare the HMC algorithm to the LBPS on a very high-dimensional model with a rather sparse graphical model structure. We showed in the experiments that the LBPS compares favourably with a simple version of HMC in this case which gives hope for future use of the LBPS on large scale graphical models. It should however be noted that the HMC algorithm we used here is not particularly involved and better tuning could be applied (more leapfrog steps, better sample size) though we wanted to keep the sampler simple to be fair to the local BPS which is still quite unexplored.

A possible interesting avenue of work would be to incorporate the local BPS within a block-Gibbs sampler in order to avoid the triggering of wide-reaching node refreshments when the matrix of ratings is not "sufficiently sparse".

Much remains to be explored on how to better tune Piecewise Deterministic samplers and in particular its local version for MRFs. To the best of our knowledge, this was the first attempt to use the LBPS on a large-scale model in statistics or machine learning.

A key element for the viability of the LBPS for a large scale graphical model is the sparsity of the graph. As we hinted at earlier, if each variable node is connected to many factors then every time a factor is triggered, every factor that shares a variable with it is also triggered. If this is a large portion of the graph, the advantage of exploiting the factorisation structure of the model is severely diminished. In figure 4.4, we look at the distribution of the number of ratings per users. It shows that the distribution is heavy tailed with a significant number of users having over 250 ratings (1149 users). This along with the slow convergence we observed for the LBPS in figure 4.3

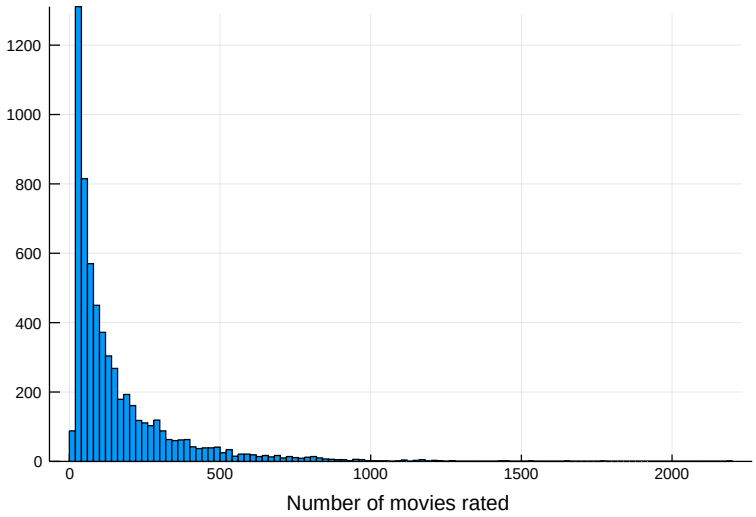


Figure 4.4: *Histogram of the number of ratings per users.*

suggests that the LBPS may be struggling due to the graph not being sparse enough.

Future work could look at other models which offer a sparse conditional dependence structure and compare the LBPS with the BPS. We suspect that for very sparse models the LBPS will outperform the BPS as observed experimentally for Gaussian Random Fields in (Bouchard-Côté et al., 2015).

Part II

Approximate Bayesian Inference

5 | Background

In this chapter, we start with a brief overview of approximate Bayesian inference. We then cover tools necessary to understand and extend the Expectation Propagation algorithm and the Loopy Belief Propagation algorithm. These methods and variants form the core of this part of the thesis.

5.1 Overview of Approximate Bayesian Inference

In this thesis we define approximate Bayesian inference in broad terms as the class of methods attempting to recover a distribution q in a restricted family of distributions $\mathcal{F} \subset \mathcal{P}(\mathcal{X})$ such that q is a "good" proxy for a posterior distribution p of interest. This definition leaves significant room to describe different methods based on the definition of what a "good proxy" means.

Let $D : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}_+$ denote a divergence between two probability distribution functions. Then, the generic variational problem we consider is the minimisation of this divergence over $\mathcal{F}$:

$$q^\star \quad \in \quad \arg\min_{q \in \mathcal{F}} \quad D(q, p). \tag{5.1}$$

Techniques attempting to solve such problems rely upon exploiting at least one of the three main characteristics:

- the definition of the discrepancy measure D,
- the definition of the restricted space $\mathcal{F}$, and
- the structure of the target distribution p.

Many discrepancy measures can be considered such as the total variation distance, the Wasserstein distance or f-divergences (Minka, 2004; Blei et al., 2016; Li and Turner, 2016; Bernton et al., 2017). However, most choices lead to the corresponding problem (5.1) being computationally very expensive to solve in general especially when the space $\mathcal{X}$ is continuous. In this part we focus on the Kullback-Leibler divergence (Kullback and Leibler, 1951) with $\mathrm{KL}(p, q) := \mathbb{E}_p[\log(p(X)/q(X))]$.

5.1.1 Variational inference

The KL divergence has been widely considered in the literature partly because it leads to variational problems that are amenable to gradient descent-based algorithms. In particular, considering the discrepancy $\mathrm{KL}(q, p)$ led to the development of the popular *variational inference* algorithms (Hoffman et al., 2013; Blei et al., 2016).

In the basic *mean-field variational inference* (MFVI), the approximating family of distributions $\mathcal{F}$ is taken to be the class of distributions that fully factorises:

$$\mathcal{F}_{\text{MFVI}} = \left\{ q \in \mathcal{P}(\mathcal{X}) \,|\, q(x_1, \ldots, x_d) = \prod_{i=1}^{d} q_i(x_i) \right\}. \tag{5.2}$$

This choice of distribution space, while rather restrictive, leads to the inference problem (5.1) being tractable. Indeed, writing $q_{\neg i}(x_{\neg i}) = \prod_{j \neq i} q_j(x_j)$, we have

$$\mathrm{KL}(q, p) = \mathbb{E}_{q_i}\left[\log q_i - \mathbb{E}_{q_{\neg i}}\left[\log p\right]\right] + \mathbb{E}_{q_{\neg i}}\left[\sum_{j \neq i} \log q_j\right] \tag{5.3}$$

suggesting to iterate over the components taking each time $q_i \propto \exp(\mathbb{E}_{q^{\neg i}} \log p)$. This scheme can be used to provably decrease the objective function (Hoffman et al., 2013; Kucukelbir et al., 2016; Blei et al., 2016); however, the problem is non-convex and no practical guarantees exist as to what the iterations converge to. In this thesis, we focus on an alternative approximate Bayesian inference class of algorithms which exploits the factorisation structure of the target distribution p directly instead of imposing a factorisation structure a priori like in MFVI.

5.1.2 ADF, EP and LBP

The Assumed Density Filtering (ADF) algorithm and the Expectation Propagation (EP) algorithm are two other popular methods that are also based on the KL divergence. The choice of distribution space is usually the exponential family associated to a sufficient statistic ϕ. Additionally, these methods assume that the target distribution p factorises into terms that are easier to handle than p itself. We introduce these methods in sections 5.2 and 5.3.

The Belief Propagation (BP) algorithm and the Loopy Belief Propagation (LBP) algorithm are message-passing algorithms which target distributions that factorise according to a MRF. Those methods can also be shown to be fixed-point algorithms corresponding to a specific form of divergence (5.1) (Yedidia et al., 2001, 2002).

In this part of the thesis we focus on the use of EP and (L)BP algorithms for inference on MRFs and discuss applications. In point 5.3 we cover the ADF and EP algorithms with exponential family distributions and in point 5.4 we introduce the BP and LBP algorithms.

5.2 Exponential Family and Convexity

In this part we introduce the notations and tools from the theory of exponential families and convex analysis. We refer to Brown (1986) for exponential family theory and to Rockafellar (1970) for convex analysis. The notations used here are inspired from these authors and Wainwright and Jordan (2008) though the latter focuses on discrete state spaces while we focus on continuous state spaces.

5.2.1 Log partition function and natural parameter space

We define the *exponential family* $\mathcal{F}_\phi(\mathcal{X}) \subseteq \mathcal{P}(\mathcal{X})$ associated with a *sufficient statistic* $\phi : \mathcal{X} \to \mathbb{R}^d$ on a set $\mathcal{X} \subseteq \mathbb{R}^n$ as the collection of probability distribution functions indexed by a parameter $\theta \in \mathbb{R}^d$ and of the form

$$q_\theta(x) \quad = \quad \exp\left(\langle \theta, \phi(x) \rangle - A(\theta)\right) \tag{5.4}$$

with respect to a base measure ν on $\mathcal{X}$. The *log partition* function A is defined such as to normalise q_θ:

$$A(\theta) \quad = \quad \log \int \exp \langle \theta, \phi(x) \rangle \, \nu(\mathrm{d}x). \tag{5.5}$$

The set of *natural parameters* $\Omega \subseteq \mathbb{R}^d$ is defined as the set of parameters θ such that $A(\theta)$ is bounded (and, therefore, q_θ is a proper distribution):

$$\Omega \quad := \quad \left\{ \theta \in \mathbb{R}^d \mid A(\theta) < \infty \right\}. \tag{5.6}$$

An exponential family is said to be *regular* when the corresponding set Ω is nonempty and open. Additionally, an exponential family is said to be *minimal* provided there does not exist a nonzero vector $a \in \mathbb{R}^d$ such that $\langle a, \phi(x) \rangle$ is constant ν-almost every-

where. A distribution in a minimal exponential family is therefore one-to-one with its parameter $\theta \in \Omega$. In the rest of this document, we will assume that we are working with a regular and minimal exponential family.

The log-partition function A is convex on Ω. Indeed, let $\theta_1, \theta_2 \in \Omega$ and $\lambda \in [0, 1]$, then using Hölder's inequality with parameters $(1-\lambda)^{-1}$ and λ^{-1}, we can write:[1]

$$\int \exp\langle (1-\lambda)\theta_1, \phi(x)\rangle \exp\langle \lambda\theta_2, \phi(x)\rangle \, \nu(\mathrm{d}x) \\ \leq \exp(A(\theta_1))^{1-\lambda} \exp(A(\theta_2))^{\lambda}. \tag{5.7}$$

Taking the logarithm of that inequality shows that A is convex on Ω and therefore that Ω itself is convex.

5.2.2 Gradient and convex-conjugate of the log-partition function

In a regular exponential family, it is well known (Brown, 1986, theorem 2.2) that the log-partition function is infinitely continuously differentiable and that its gradient and Hessian are related to the first two moments of the sufficient statistic:

$$\nabla A(\theta) = \mathbb{E}_{q_\theta}[\phi(X)], \quad \text{and} \quad \nabla^2 A(\theta) = \mathbb{V}_{q_\theta}[\phi(X)]. \tag{5.8}$$

An important consequence is that, for a minimal exponential family, the covariance matrix $\mathbb{V}_{q_\theta}[\phi(X)]$ is positive definite which in turns implies that A is strictly convex on Ω. Indeed, take any nonzero vector $a \in \mathbb{R}^d$ then by definition, $\langle a, \phi(X)\rangle$ cannot be constant ν-almost everywhere so that $\mathbb{V}_{q_\theta}[\langle a, \phi(X)\rangle] > 0$ for any $\theta \in \Omega$ and therefore:

$$0 < \langle a, \mathbb{V}_{q_\theta}[\phi(X)]a\rangle, \tag{5.9}$$

[1] For two real-valued measurable functions f and g, Hölder's inequality with parameters $p, q \in [1, \infty]$ states that $\|fg\|_1 \leq \|f\|_p \|g\|_q$ provided $p^{-1} + q^{-1} = 1$.

so that $\mathbb{V}_{q_\theta}[\phi(X)]$ is positive definite. Let $\mathcal{M}^\star$ denote the image of Ω under the gradient mapping ∇A or $\mathcal{M}^\star = \nabla A(\Omega)$. Since A is strictly convex, ∇A is a strictly monotone map and forms a bijection between Ω and $\mathcal{M}^\star$. Let $\theta^\star$ be the image of a point $\mu^\star \in \mathcal{M}^\star$ through the inverse map $(\nabla A)^{-1}$. This can be written as

$$\begin{aligned} \theta^\star &= \{\theta \in \Omega \mid \nabla A(\theta) = \mu^\star\}, \\ &= \{\theta \in \Omega \mid \nabla(A(\theta) - \langle \theta, \mu^\star \rangle) = 0\}. \end{aligned} \tag{5.10}$$

Since $(A(\theta) - \langle \theta, \mu \rangle)$ is a strictly convex function in θ, the last expression corresponds to a first-order condition for its minimiser. [2] Therefore, we can write

$$\theta^\star = \arg\min_{\theta \in \Omega} \quad A(\theta) - \langle \theta, \mu^\star \rangle \tag{5.11}$$

which is related to the convex-conjugate of A on $\mathcal{M}^\star$ defined as

$$A^\star(\mu) := \max_{\theta \in \Omega} \quad \langle \theta, \mu \rangle - A(\theta). \tag{5.12}$$

Note that $A^\star$ is itself a convex function (Rockafellar, 1970, Theorem 12.2). Using this definition, we have

$$A^\star(\mu^\star) = \langle \theta^\star, \mu^\star \rangle - A(\theta^\star) \tag{5.13}$$

and $(\theta^\star, \mu^\star)$ is said to form a *dual pair*. Let us now consider an arbitrary $\theta^+ \in \Omega \backslash \{\theta^\star\}$ and the corresponding $\mu^+ = \nabla A(\theta^+) \in \mathcal{M}^\star$. Then, by strict convexity of A, we have

$$A(\theta^\star) > A(\theta^+) + \left\langle \theta^\star - \theta^+, \mu^+ \right\rangle. \tag{5.14}$$

[2] A first-order condition for a minimiser indicates that for a convex, differentiable function f on a set Ω, $\{x \in \Omega \mid \nabla f(x) = 0\} \subseteq \arg\min_x f(x)$. If there exists a unique point $x^\star \in \Omega$ such that $\nabla f(x^\star) = 0$ then it is also the unique minimiser of f and both sets are equal to that point (Rockafellar, 1970, theorem 27.1).

Using the duality property (5.13) and with $\mu^\star = \nabla A(\theta^\star)$, the previous inequality reads

$$\begin{aligned} -A^\star(\mu^\star) + \langle \theta^\star, \mu^\star \rangle &> -A^\star(\mu^+) + \left\langle \theta^+, \mu^+ \right\rangle + \left\langle \theta^\star - \theta^+, \mu^+ \right\rangle . \\ \iff \quad A^\star(\mu^+) &> A^\star(\mu^\star) + \left\langle \mu^+ - \mu^\star, \theta^\star \right\rangle . \end{aligned} \tag{5.15}$$

Since the inequality (5.15) holds for any $\mu^+ \neq \mu^\star$, $\theta^\star$ is said to be a *subgradient* of $A^\star$ at $\mu^\star$. Since the mapping ∇A is one-to-one on Ω, $\theta^\star$ is necessarily the only such subgradient. Therefore, $A^\star$ is differentiable on $\mathcal{M}$ (Rockafellar, 1970, theorem 25.1) and $\nabla A^\star$ is the inverse mapping of $(\nabla A)^{-1}$ from $\mathcal{M}^\star$ to Ω. The relationship is illustrated in figure 5.1 below.

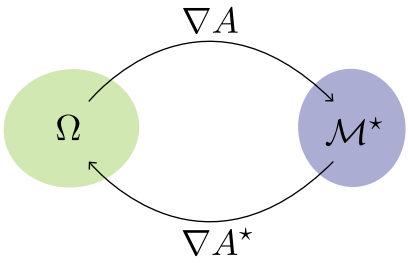


Figure 5.1: *Illustration of the bijection between the set of natural parameters Ω and its image under ∇A: $\mathcal{M}^\star$.*

5.2.3 The mean parameter space

It remains to characterise more precisely the set $\mathcal{M}^\star = \nabla A(\Omega)$. It is a subset of the set of all realisable *mean parameters* associated with ϕ:

$$\mathcal{M}^\star \subseteq \mathcal{M} := \left\{ \mu \in \mathbb{R}^d \mid \exists p \in \mathcal{P}(\mathcal{X}) \text{ s.t. } \mu = \mathbb{E}_p[\phi(X)] \right\} . \tag{5.16}$$

This space $\mathcal{M}$ is also known as the *mean parameter space* associated with ϕ. It is easy to see that this space is convex since $\mathcal{P}(\mathcal{X})$ is convex. More interestingly, it can be shown that, for a minimal exponential family, $\mathcal{M}^\star$ is in fact the interior of $\mathcal{M}$ (Wainwright and Jordan, 2008, theorem 3.3). Note that, since the interior of a convex set is

necessarily convex, $\mathcal{M}^\star$ is itself convex. Finally, note that since $\mathcal{M}^\star = \operatorname{int}(\mathcal{M})$, $A^\star$ can be continuously extended on $\mathcal{M}$ by changing the $\max$ into a $\sup$ in (5.12). In a similar fashion, for a point $\mu \in \mathcal{M} \backslash \mathcal{M}^\star$, we can consider a sequence of points $\mu_1, \mu_2, \ldots$ in $\mathcal{M}^\star$ such that $\lim_{i\to\infty} \mu_i = \mu$ and identify $\nabla A^\star(\mu)$ to $\lim_{i\to\infty} \nabla A^\star(\mu_i)$ by continuity. The extended operator is then defined as a mapping from $\mathcal{M}$ to $\operatorname{cl}(\Omega)$.

5.2.4 The case of the Gaussian distribution

The d-dimensional multivariate Gaussian distribution with mean $\mu_0 \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ corresponds to an exponential family with sufficient statistics $\phi(x) = (x, xx^t/2)$.[3] Correspondingly, the parametrisation in the natural parameter space can be written $\theta = (\Sigma^{-1}\mu_0, -\Sigma^{-1})$ with $\Omega = \mathbb{R}^d \times \mathbb{S}^d_-$ and the parametrisation in the mean parameter space is $\mu = (\mu_0, (\mu_0\mu_0^t + \Sigma)/2)$. The transformation ∇A and $\nabla A^\star$ therefore correspond to the mappings

$$\nabla A : \quad (\theta_1, \theta_2) \;\to\; (-\theta_2^{-1}\theta_1, (\theta_2^{-1}\theta_1\theta_1^t\theta_2^{-1} - \theta_2^{-1})/2), \tag{5.17}$$

$$\nabla A^\star : \quad (\mu_1, \mu_2) \;\to\; ((2\mu_2 - \mu_1\mu_1^t)^{-1}\mu_1, -(2\mu_2 - \mu_1\mu_1^t)^{-1}), \tag{5.18}$$

where $\theta_1 = \Sigma^{-1}\mu_0$, $\theta_2 = -\Sigma^{-1}$, $\mu_1 = \mu_0$ and $\mu_2 = (\mu_0\mu_0^t + \Sigma)/2$.

Therefore, we note that $\mathcal{M}^\star = \{(\mu_1, \mu_2) \,|\, \mu_1 \in \mathbb{R}^d, \mu_2 \in \mathbb{R}^{d\times d}, (2\mu_2 - \mu_1\mu_1^t) \in \mathbb{S}^d_+\}$ and the mean parameter space $\mathcal{M}$ contains pairs such that $(2\mu_2 - \mu_1\mu_1^t)$ is symmetric *semi* positive definite.

[3] We had defined the sufficient statistics as a mapping from $\mathcal{X}$ to $\mathbb{R}^d$. Here however, $\phi(x) = (x, xx^t/2) \in \mathbb{R}^p \times \mathbb{R}^{p\times p}$. Formally, we consider the vector formed of all the components of x and $xx^t/2$ but the results are obviously equivalent.

5.3 Assumed Density Filtering and Expectation Propagation

5.3.1 Approximate inference in the exponential family

We consider the context of approximate inference in the exponential family $\mathcal{F}_\phi(\mathcal{X})$. In the previous section, we showed that a target distribution p could be associated with a point $\mu_p \in \mathcal{M}$ with $\mu_p = \mathbb{E}_p[\phi(X)]$ for a sufficient statistic ϕ. Further, we showed that if the family is minimal and if $\mu_p \in \mathcal{M}^\star$, the mean parameter could be directly associated to a distribution in $\mathcal{F}_\phi(\mathcal{X})$ with parameter θ given by $\nabla A^\star(\mu_p)$. It is therefore natural to consider this distribution q_θ as potentially forming a good approximation to p in $\mathcal{F}_\phi(\mathcal{X})$. This is supported by the fact that this distribution actually minimises the divergence $\mathrm{KL}(p, q_{\theta'})$. Indeed, observe that

$$\mathrm{KL}(p, q_{\theta'}) \quad = \quad \mathbb{E}_p[\log p] - \langle \theta', \mu_p \rangle + A(\theta'), \tag{5.19}$$

which is a strictly convex function in θ'. By definition, a parameter $\theta = \nabla A^\star(\mu_p)$ is such that $\nabla A(\theta) = \mu_p$ and therefore verifies the first order condition. This shows that q_θ minimises the divergence (5.19). It is useful at this point to define a projection operator $\mathbf{P}_\phi : \mathcal{P}(\mathcal{X}) \to \mathrm{cl}(\Omega)$ with

$$\theta \;=\; \mathbf{P}_\phi[p] \quad \Longleftrightarrow \quad \theta \;=\; \nabla A^\star(\mathbb{E}_p[\phi(X)]). \tag{5.20}$$

Provided $\mathbf{P}_\phi[p] \in \Omega$, the parameter θ labels a proper distribution $q_\theta \in \mathcal{F}_\phi(\mathcal{X})$ that minimises the KL (5.19) and equivalently verifies the *global moment matching condition* (GMMC)

$$\text{(GMMC)} \qquad \mathbb{E}_{q_\theta}[\phi(X)] \;=\; \mathbb{E}_p[\phi(X)]. \tag{5.21}$$

It will also be convenient to use the following abuse of notation: for an unnormalised distribution p_u with some normalisation constant Z^{-1} we will write $\mathbf{P}_\phi[p_u]$ to implicitly mean $\mathbf{P}_\phi[Z^{-1}p_u]$.

The application of this projection operator implicitly requires the ability to perform two computations. First, it requires the ability to compute the expected value $\mu_p = \mathbb{E}_p[\phi(X)]$ which is typically intractable since we are in a context where we are trying to approximate p for that very purpose. Then, provided we can compute μ_p, we need to be able to apply the inverse mapping $\nabla A^\star(\mu_p)$ which is not tractable in general. We will introduce below contexts in which the first requirement can be approximately met. The second requirement effectively means that we are constrained to exponential families for which computing the inverse mapping $\nabla A^\star$ can be done explicitly or is cheap to approximate. For continuous, high-dimensional state spaces, it is overwhelmingly the Gaussian distribution that is considered in the literature with either a diagonal or a full covariance matrix (Kuss and Rasmussen, 2005; Herbrich, 2005; Herbrich et al., 2006; Yu et al., 2006; Hernandez-Lobato et al., 2013).

5.3.2 Online Bayesian learning and Assumed Density Filtering

Let us consider the context of online Bayesian learning where one is interested in the evolution of a posterior distribution $p(x \mid y_{1:t})$ given observed iid. data points $y_{1:t}$ up to current time t and a new iid. data point y_{t+1}. The new posterior is given directly by Bayes rule with $p(x \mid y_{1:t+1}) \propto p(y_{t+1} \mid x)p(x \mid y_{1:t})$. In Opper (1998), the author considers that the exact posteriors are intractable but suggests constructing a sequence of approximations in the exponential family $\mathcal{F}_\phi(\mathcal{X})$. Using the notations from the previous point, the algorithm corresponds to a sequence of iteration of the form

$$\theta_{t+1} \quad \leftarrow \quad \mathbf{P}_\phi[p(y_{t+1} \mid \cdot)q_{\theta_t}], \tag{5.22}$$

where, at the first step, q_{θ_0} is replaced by the prior π_0. In words, the posterior up to time t is approximated by the exponential family distribution q_{θ_t} which then serves as prior to form an approximate posterior $p(y_{t+1} \mid x) q_{\theta_t}(x)$. Given that this approximate posterior is not necessarily in the exponential family, it needs to be projected as per (5.22). At time $t+1$, the distribution $q_{\theta_{t+1}}$ can now be interpreted as an approximation of the posterior distribution given all the data up to that time or

$$q_{\theta_{t+1}}(x) \quad \approx \quad p(x \mid y_{1:t}) \quad \propto \quad \pi_0(x) \prod_{s=1}^{t+1} p(y_s \mid x). \tag{5.23}$$

We can therefore reinterpret the algorithm more generally as targeting a distribution p that factorises in a fixed number (say K) of factors:

$$p(x \mid y_{1:t}) \quad \propto \quad \pi_0(x) \prod_{i=1}^{K} t_i(x) \tag{5.24}$$

where the factors t_i are nonnegative. This is the *Assumed Density Filtering* (ADF) algorithm. Note that in this case there is no "natural ordering" as in the original online learning case and the algorithm can be applied after any permutation of the factors.

Algorithm 10 *Assumed Density Filtering*

Let $\theta_1 = \mathbf{P}_\phi[\pi_0 t_1]$
for $i = 2 : K$ **do**
    $\theta_i \leftarrow \mathbf{P}_\phi[q_{\theta_{i-1}} t_i]$ ▷ parameter of the new approximation
end for
return θ_K

Let $\omega_i = (\theta_i - \theta_{i-1})$ denote the difference between subsequent parameters (with $\omega_1 = \theta_1$). Correspondingly, $\theta_K = \sum_{i=1}^{K} \omega_i$, and letting $\tilde{t}_i(x) := \exp \langle \omega_i, \phi(x) \rangle$, which can be interpreted as an approximation to the factor t_i, we can write

$$q_{\theta_K}(x) \quad \propto \quad \pi_0(x) \prod_{i=1}^{K} \tilde{t}_i(x). \tag{5.25}$$

The main issue with using ADF on a MRF is that the order in which the graph is read, or equivalently the order in which the factors t_i are considered, will affect the final approximation q_{θ_K}. A natural extension of the ADF algorithm to take this into account is to perform multiple passes of ADF while considering random orderings of the factors and correct for the fact that each factor is considered multiple times. This is the core idea behind the *Expectation Propagation* algorithm.

5.3.3 Expectation Propagation

In the Expectation Propagation (EP) algorithm (Minka, 2001a,b; Seeger, 2007; Gelman et al., 2014), we consider that we have an initial approximation $q_\theta \in \mathcal{F}_\phi(\mathcal{X})$ that we can write as in (5.25) (typically obtained with a pass of ADF) and seek to iteratively improve its parameter θ. At each step, a factor t_i from the target distribution is considered and, by contrast to ADF, we now consider the projection $\theta_i = \mathbf{P}_\phi[q_\theta t_i/\tilde{t}_i]$. The difference with ADF is therefore the removal of the previous factor approximation $\tilde{t}_i$ from the current approximation q_θ before the projection. ADF can also be reinterpreted as a single pass of EP where, initially, all the factor approximations are set to $\tilde{t}_i \equiv 1$.

Algorithm 11 *Expectation Propagation*

Initialise $\theta, \omega_1, \ldots, \omega_K$ such that $\theta = \sum_{i=1}^K \omega_i \in \Omega$ (e.g. via ADF)
for $k = 1 : N_{\text{EP}}$ **do**
  Let σ be a random permutation of $(1, \ldots, K)$
  for $i = 1 : K$ **do**
    $\theta \leftarrow \mathbf{P}_\phi[q_{\theta^{\text{old}}} t_{\sigma(i)}/\tilde{t}_{\sigma(i)}]$ ▷ *parameter of the new global approximation*
    $\omega_{\sigma(i)} \leftarrow \omega_{\sigma(i)} + \theta - \theta^{\text{old}}$ ▷ *parameter of the new factor approximation* $\tilde{t}_{\sigma(i)}$
    $\theta^{\text{old}} \leftarrow \theta$
  end for
end for
return θ

In Expectation Propagation, we denote by $q_{-i} \propto q_\theta/\tilde{t}_i$ a *cavity distribution* and by $q_i \propto q_{-i} t_i$ a *tilted distribution* (Seeger, 2007; Gelman et al., 2014). Provided the EP iterations converge, the global parameter θ is equal to $\mathbf{P}_\phi[q_i]$ for all i. Put differently,

it is such that all the *local moment matching conditions* (LMMC) are met:

$$\text{(LMMC)} \qquad \mathbb{E}_{q_i}[\phi(X)] \;=\; \mathbb{E}_{q_\theta}[\phi(X)], \quad i = 1, \ldots, K. \tag{5.26}$$

These conditions can be interpreted as a relaxation of the GMMC (5.21) and the EP algorithm can be interpreted as a fixed point algorithm targeting the LMMC. We will expand on this in chapter 6.

5.4 Belief Propagation

In this section we introduce the Belief Propagation (BP) algorithm for tree-structured MRFs which, we show, is an iterative algorithm that can recover functions – the beliefs – proportional to the exact node marginals by passing "messages" through the graph (Pearl, 1988; Wainwright and Jordan, 2008). We also show how this algorithm can be related to the two-filter formula for smoothing on a HMM discussed in chapter 3.

We then introduce the Loopy Belief Propagation (LBP), an extension of the BP algorithm on graphs with cycles. This algorithm does not lead to functions proportional to the exact node marginals but recovers proxies for it which work well in practice (Yedidia et al., 2002; Sudderth et al., 2003).

In line with the rest of this document we concern ourselves with the case where the state-spaces associated with the graph nodes are continuous

5.4.1 Belief Propagation on a tree

The problem of computing singleton marginals on a tree is relatively easy since the recursive structure of the problem can be exploited. Let s denote a node of interest so that we are targeting the marginal $p(x_s)$. The full joint on the tree can then be written

relative to that node:

$$p(x) \quad \propto \quad \psi_s(x_s) \prod_{t \in \partial s} \psi_{ts}(x_t, x_s) \left[\psi_t(x_t) \phi_{\partial t \setminus s}(x_t, x_{\partial t \setminus s}) \right], \tag{5.27}$$

where the $\phi_{\partial t \setminus s}$ are given recursively by

$$\phi_{\partial t \setminus s}(x_t, x_{\partial t \setminus s}) \quad = \quad \prod_{r \in \partial t \setminus s} \psi_{rt}(x_r, x_t) \left[\psi_r(x_r) \phi_{\partial r \setminus t}(x_r, x_{\partial r \setminus t}) \right],$$

and $\phi_\emptyset \equiv 1$ (for a leaf node). This decomposition is illustrated at figure 5.2.

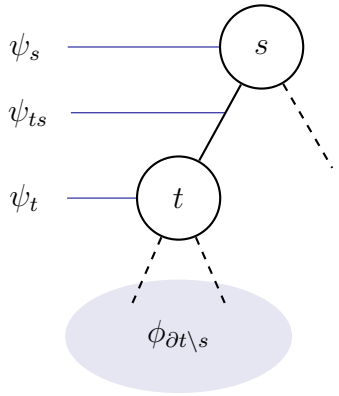


Figure 5.2: *Illustration of the elements of the factorisation (5.27): the node potential ψ_s of the node of interest, the potential corresponding to the edge connecting it to one of its neighbour t: ψ_{st}, the potential at that node, and the product of all subsequent potentials attached to that node $\phi_{\partial t \setminus s}$.*

Using this recursive factorisation, and integrating out all variables apart from x_s, the marginal of interest can be written as

$$p_s(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in \partial s} \int \psi_{ts}(x_t, x_s) \left[\psi_t(x_t) \underbrace{\int \phi_{\partial t \setminus s}(x_t, x_{\partial t \setminus s}) \, \mathrm{d}x_{\partial t \setminus s}}_{=: \kappa(x_t)} \right] \mathrm{d}x_t.$$

Exploiting the recursive structure of $\phi_{\partial t \setminus s}$, we can expand the κ term:

$$\kappa(x_t) \quad = \quad \prod_{r \in \partial t \setminus s} \int \psi_{rt}(x_r, x_t) \left[\psi_r(x_r) \int \phi_{\partial r \setminus t}(x_r, x_{\partial r \setminus t}) \mathrm{d}x_{\partial r \setminus t} \right] \mathrm{d}x_r.$$

Consequently, we can write

$$p_s(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in \partial s} m_{ts}(x_s)$$

where the *messages* m_{ts} are defined as

$$m_{ts}(x_s) \quad := \quad \int \psi_{ts}(x_t, x_s) M_{ts}(x_t) \, \mathrm{d}x_t, \tag{5.28}$$

and the *pre-messages* M_{ts} as

$$M_{ts}(x_t) \quad := \quad \psi_t(x_t) \prod_{r \in \partial t \setminus s} m_{rt}(x_t), \tag{5.29}$$

with, for a leaf node α, $M_{\alpha s} \equiv \psi_\alpha$. This is the BP algorithm which can be described more compactly as starting with the pre-messages on the leaf-nodes of the tree and propagating using the following equations:

$$\begin{cases} M_{st}(x_s) & = & \psi_s(x_s) \prod_{r \in \partial s \setminus t} m_{rs}(x_s) \\ m_{ts}(x_s) & = & \int \psi_{st}(x_s, x_t) M_{ts}(x_t) \, \mathrm{d}x_t \\ B_s(x_s) & = & m_{ts}(x_s) M_{st}(x_s) \end{cases} . \tag{5.30}$$

The $B_s(x_s)$ are known as the *beliefs* and are proportional to the true marginals as we showed with the development above.

5.4.2 Belief propagation on a chain

A particular case of interest is the chain graph which is associated with Hidden Markov Models introduced at section 1.2 and studied in more details at chapter 3. On the leaf node, we have a prior $\pi_0(x_1)$ and an observation with likelihood $p(y_1 \mid x_1)$. Therefore, we have $M_{12}(x_1) = \pi_0(x_1) p(y_1 \mid x_1) \propto p(x_1 \mid y_1)$. Correspondingly, the message m_{12}

is given by

$$m_{12}(x_2) \quad \propto \quad \int p(x_2 \mid x_1) p(x_1 \mid y_1) \, \mathrm{d}x_1 \quad = \quad p(x_2 \mid y_1).$$

Forward messages

Let us assume that $m_{t-1,t} \propto p(x_t \mid y_{1:t-1})$ (prediction density). Then, the next pre-message is:[4]

$$M_{t,t+1}(x_t) \quad \propto \quad p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) \quad \propto \quad p(x_t \mid y_{1:t}). \tag{5.31}$$

Consequently, the next message can be computed thereby verifying our earlier assumption:

$$m_{t,t+1}(x_{t+1}) \quad \propto \quad \int p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t}) \, \mathrm{d}x_t \quad = \quad p(x_{t+1} \mid y_{1:t}). \tag{5.32}$$

Backward messages

In a similar way, we now start with a leaf node with no prior, i.e.: $M_{T,T-1}(x_T) = p(y_T \mid x_T)$ and hence

$$m_{T,T-1}(x_{T-1}) \quad = \quad \int p(x_T \mid x_{T-1}) p(y_T \mid x_T) \, \mathrm{d}x_T \quad \propto \quad p(y_T \mid x_{T-1}).$$

Let us assume that $M_{t+1,t}(x_{t+1}) \propto p(y_{t+1:T} \mid x_{t+1})$. Noting that we can write the joint distribution on $(x_t, x_{t+1}, y_{t+1:T})$ in two equivalent ways:

$$\begin{aligned} p(x_t, x_{t+1}, y_{t+1:T}) &= p(x_{t+1} \mid y_{t+1:T}, x_t) p(y_{t+1:T} \mid x_t) p(x_t) \\ &= p(y_{t+1:T} \mid x_{t+1}) p(x_{t+1} \mid x_t) p(x_t) \end{aligned}$$

[4] Using $p(x_t \mid y_{1:t}) = p(x_t, y_{1:t-1}, y_t)/p(y_{1:t}) \propto p(y_t \mid x_t) p(x_t \mid y_{1:t-1})$.

we have that $p(x_{t+1} \mid y_{t+1}, x_t)p(y_{t+1:T} \mid x_t) = p(y_{t+1:T} \mid x_{t+1})p(x_{t+1} \mid x_t)$. Using the update equation for a message and our earlier assumption about the form of the pre-messages, we have

$$\begin{aligned} m_{t+1,t}(x_t) &\propto \int p(x_{t+1} \mid x_t)p(y_{t+1:T} \mid x_{t+1}) \, \mathrm{d}x_{t+1} \\ &\propto \int p(x_{t+1} \mid y_{t+1:T}, x_t)p(y_{t+1:T} \mid x_t) \, \mathrm{d}x_{t+1} \\ &\propto p(y_{t+1:T} \mid x_t). \end{aligned} \tag{5.33}$$

We can then confirm our earlier assumption since

$$M_{t,t-1} \propto p(y_t \mid x_t)p(y_{t+1:T} \mid x_t) \propto p(y_{t:T} \mid x_t), \tag{5.34}$$

which shows that the backward messages on a HMM are $p(y_{t+1:T} \mid x_t)$. Summarising, on a HMM, we have that the forward messages (t to $t+1$) correspond to the predictive densities and the forward pre-messages correspond to the filtering densities whilst the backward pre-messages correspond to the backward information filters discussed in chapter 3.

Beliefs

Now that the messages have been defined explicitly, we can write the beliefs as

$$\begin{aligned} B_t(x_t) &= m_{t-1,t}(x_t)M_{t,t-1}(x_t) \\ &\propto p(x_t \mid y_{1:t-1})p(y_{t:T} \mid x_t) \end{aligned} \tag{5.35}$$

which is the two filter formula for the smoothing distribution $p(x_t \mid y_{1:T})$ which we had introduced at point 3.1.1.

5.4.3 Loopy Belief Propagation

In the presence of cycles in the graph (*loopy graph*), BP does not necessarily converge and there may be more than one fixed point because propagating messages through different orderings of the nodes can lead to inconsistent messages (see 5.3 for an illustration).

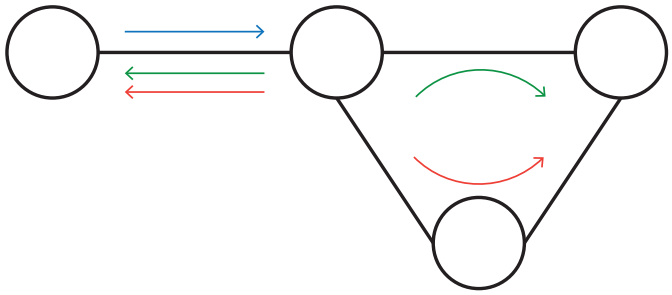


Figure 5.3: *A simple graph with a loop. Starting from the leaf node (blue arrow), the message can be propagated in the loop following a clock-wise ordering (green arrow) or a anti-clockwise ordering (orange arrow). The two backward messages coming back to the leaf node do not necessarily match. (This graph is best seen in colour.)*

A common workaround is to perform the BP algorithm multiple times on different spanning trees of the graphs corresponding to different ordering of the nodes and hope that the iterations converge to a fixed point. This is know as the *loopy belief propagation* (LBP) algorithm (Pearl, 1988; Yedidia et al., 2002):

$$\begin{cases} M_{st}^n(x_s) &= \psi_s(x_s) \prod_{r \in \partial s \setminus t} m_{rs}^{n-1}(x_s) \\ m_{ts}^n(x_s) &= \int \psi_{ts}(x_t, x_s) M_{ts}^n(x_t) \, \mathrm{d}x_t \\ B_s^n(x_s) &= m_{ts}^n(x_s) M_{st}^n(x_s) \end{cases} . \tag{5.36}$$

The initial messages are usually chosen among a class of simple distributions (e.g., the Normal distribution) potentially incorporating prior knowledge about the steady state messages. Provided the algorithm converges to a fixed point, the beliefs cannot

be guaranteed to be proportional to the true marginals unlike the tree case. However, the beliefs can provide good approximations to the marginals which makes the LBP a popular method for approximate Bayesian inference on MRFs (Sudderth et al., 2003). The fixed points of the LBP have been shown to correspond to extrema of the Bethe free energy (Yedidia et al., 2002) and there exist conditions under which the LBP algorithm provably converges (Heskes, 2004; Ihler et al., 2005).

Note that while it is known that the LBP algorithm can lead to good proxy to the marginals, the updates may still be intractable. Indeed, the computation of messages requires an integral which may be hard (or expensive) to compute accurately (e.g., if the dimension of the random variable on a node is high).

6 | Expectation Propagation for Distributed Bayesian Inference

In this chapter we discuss how the Expectation Propagation (EP) algorithm can be exploited when attempting to perform (approximate) Bayesian inference with distributed data. We focus on how different variants of the EP algorithm perform, how amenable they are to scaling and how robust they are to noisy estimations of the EP updates.

While very related to our paper (Hasenclever et al., 2017), this chapter differs in that a class of algorithms is defined and several algorithms are compared as opposed to one (the SNEP algorithm). Further, the fixed-point perspective and relation to mirror-descent is novel and the discussion of the limitation of the convergence claimed in the paper is also novel. Finally, we show in this chapter that a simpler method can lead to better performances than SNEP on simple models though more work would be needed on larger experiments to make this result definitive.

6.1 Distributed Bayesian Inference

In this section we consider the problem of performing Bayesian inference when the relevant data is distributed across different compute nodes. Let us assume that there are K such compute nodes each holding independent parts of the data y_i (with $i = 1, \ldots, K$), such that these parts form a partition of the complete data y. In a Bayesian setting, we are interested in the posterior distribution over a parameter of interest x given the entire data y. We can write this posterior as the product of K terms corresponding to the parts y_i:

$$p(x \mid y) \quad \propto \quad \pi_0(x) \prod_{i=1}^{K} p(y_i \mid x) \tag{6.1}$$

where π_0 is a prior distribution. Each of the likelihood terms itself factors over the individual data points of y_i, i.e.: $p(y_i \mid x) = \prod_{j=1}^{N_i} p(y_{ij} \mid x)$ with N_i the number of individual data points held by the ith compute node.

It is important to stress that, in the setup considered here, each compute node only has access to independent subsets of the data. In Hasenclever et al. (2017) we explain that this situation can occur in large scale learning but also in cases when the data cannot be shared directly (e.g.: for privacy reasons). This type of context is sometimes known as divide-and-conquer or consensus inference (Kleiner et al., 2014; Battey et al., 2015; Zhao et al., 2016).

Defining t_i to be nonnegative factors with $t_i(x) = p(y_i \mid x)$, the factorised form (6.1) reads $\pi_0(x) \prod_{i=1:K} t_i(x)$ which is the form (5.24) that we had used to motivate the introduction of the ADF and EP algorithms (cf. section 5.3). In the rest of this chapter, we show how different variants of the EP algorithm can be used to perform parallel variational inference in the presence of distributed data.

6.2 EP variants for Distributed Inference

The target distribution of interest has the form $p(x) \propto \pi_0 \prod_{i=1:K} t_i(x)$ where the evaluation of one of the t_i requires access to the compute node i. The aim is to construct an approximation $q \in \mathcal{F}_\phi$ parametrised by a vector θ with[1]

$$q_\theta(x) \quad = \quad \pi_0(x) \exp(\langle \theta, \phi(x) \rangle - A(\theta)) \quad = \quad \pi_0(x) \exp(-A(\theta)) \prod_{i=1}^{K} \tilde{t}_i(x) \tag{6.2}$$

where $\tilde{t}_i(x) := \exp \langle \omega_i, \phi(x) \rangle$ for parameters ω_i such that $\sum_{i=1:K} \omega_i = \theta$. The tilted distributions associated with each of the factors are given by

$$q_i(x) \quad = \quad \pi_0(x) t_i(x) \exp(\langle \lambda_i, \phi(x) \rangle - A_i(\lambda_i)) \tag{6.3}$$

where $\lambda_i = (\theta - \omega_i)$ and $A_i(\lambda_i)$ is the corresponding log-partition function. Much like the log-partition function of q, the A_i are convex and $\nabla A_i(\lambda_i) = \mathbb{E}_{q_i}[\phi(X)]$. Indeed, each A_i is simply the log-partition functions of the exponential-family $\mathcal{F}_\phi$ with respect to a modified base-measure including t_i. In the EP setting, we seek to determine the parameters $\omega_1, \ldots, \omega_K$ such that the LMMC (5.26) hold, i.e.: $\mathbb{E}_q[\phi(X)] = \mathbb{E}_{q_i}[\phi(X)]$ for each $i = 1, \ldots, K$. The general parallel computational framework iterates on the following steps:

1. the master node sends a global parameter θ to each compute node,
2. each compute node i attempts to find a new ω_i such that the corresponding LMMC is (approximately) met and sends the updated ω_i to the master node,

[1] we included a prior here unlike in (5.5), the log-partition function is therefore implicitly understood to consider the prior as well with

$$A(\theta) \quad = \quad \log \int \pi_0(x) \exp \langle \theta, \phi(x) \rangle \, \nu(\mathrm{d}x).$$

3. the master node integrates the new ω_i into θ.

This is illustrated in figure 6.1 drawn from our paper (Hasenclever et al., 2017).

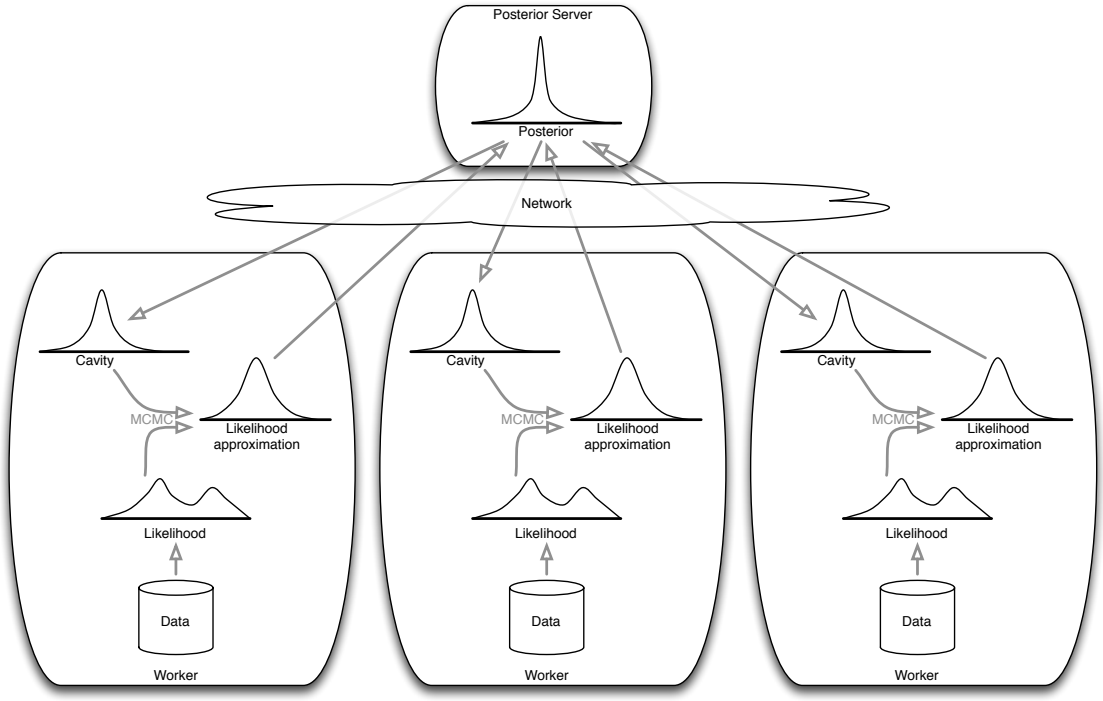


Figure 6.1: *Illustration of the use of EP as the backbone for distributed Bayesian learning. Each compute node or worker (bottom) has access to its data and can compute the likelihood of that data. Including the cavity retrieved from the master and projecting on the exponential family leads to an approximation of the global likelihood which can be sent back to the master and combined with other such approximations sent back by the other workers.*

As we will see, there are different ways to implement this general framework. We will show empirically that a determining element in the performance of a particular implementation is its robustness to Monte Carlo noise. Indeed, in our general setting we do not assume that computing $\mathbb{E}_{q_i}[\phi(X)]$ can be done exactly. Rather, we assume that it is approximated by a Monte Carlo estimator.

6.2.1 Damped updates in the natural parameter space

In their paper, Xu et al. (2014) suggest a method to synchronise Monte Carlo samplers running on independent compute nodes. They call it *Sequential Moment Sharing* or

SMS for short. It was originally proposed to scale up MCMC methods and as such it assumes that MCMC chains can be run for many iterations to convergence in between communications with the master. Effectively, SMS corresponds precisely to the case mentioned at the previous point consisting of running EP across multiple physical nodes while using noisy estimates of $\mathbb{E}_{q_i}[\phi(X)]$, the expected value of the sufficient statistics under the tilted distribution. Below, we introduce the algorithm as a damped fixed-point iteration in the natural parameter space. This will prove useful when comparing different EP-like algorithms.

We showed that the basic EP algorithm could be interpreted as a kind of fixed point iteration targeting the Local Moment Matching Conditions (5.26) (LMMC). At a given step of the algorithm, when considering the inclusion of the factor t_i, the global parameter should therefore be updated such as to verify $\mathbb{E}_{q_\theta}[\phi(X)] = \mathbb{E}_{q_i}[\phi(X)]$. Let us define the moment of the local tilted distribution as $\mu_i := \mathbb{E}_{q_i}[\phi(X)]$, the update then amounts to finding θ such that $\nabla A(\theta) = \mu_i$ or

$$\theta \quad \leftarrow \quad \nabla A^\star(\mu_i). \tag{6.4}$$

In the case where one is considering a Monte Carlo estimator for μ_i, these updates are inexact and it is crucial to mitigate the effect of the noise by considering *damped updates*. In fact, even when one uses the exact μ_i, it can be beneficial to the overall convergence of the algorithm to consider damped updates (Heskes and Zoeter, 2003). Damping (6.4) leads to the modified update:

$$\theta \quad \leftarrow \quad (1-\kappa)\theta + \kappa \nabla A^\star(\hat{\mu}_i), \tag{6.5}$$

where $\kappa \in (0, 1]$ is the damping parameter and $\hat{\mu}_i$ is an estimator for μ_i. As in the gradient descent algorithm, the damping parameter can be decreased over iterations

according to a schedule in order to further help convergence (in point 6.2.4 we expose in more details how EP can be understood in a gradient descent framework). The damped update (6.5) can also be expressed in terms of the ω_i (parameter of the factor approximation $\tilde{t}_i$) and the λ_i (parameter of the corresponding cavity):

$$\omega_i \leftarrow \omega_i + \kappa(\nabla A^\star(\hat{\mu}_i) - \lambda_i). \tag{6.6}$$

One can also swap the role of λ_i and ω_i and obtain another valid damped update mechanism. Both forms are amenable to parallelisation since they can be expressed solely in terms of local parameters attached to the ith compute node.

The basic SMS algorithm operates synchronous updates. At step k, each node receives the current global parameter θ and computes the parameter corresponding to the current tilted distribution q_i i.e.: $\lambda_i = (\theta - \omega_i)$. Each node then forms a Monte Carlo estimator $\hat{\mu}_i$ of $\mathbb{E}_{q_i}[\phi(X)]$, computes the damped step (6.6) and returns the updated local parameter ω_i to the master node. The master node then aggregates all updated local parameters and updates the global parameter:

$$\theta \leftarrow (1 - \kappa')\theta + \kappa' \sum_{i=1}^{K} \omega_i \tag{6.7}$$

where $\kappa' \in (0, 1]$ is a global damping parameter which may also help overall convergence. These steps are summarised at the algorithm 12 below.

It is easy to see how this algorithm can be altered to accommodate asynchronous updates. Indeed, instead of waiting for all nodes to complete their tasks, each node could asynchronously request the most recent global parameter θ available from the master node while the master node would continuously integrate information it receives from the compute nodes. In fact, each node would then simply send the difference $\delta_i = (\omega_i^{\text{new}} - \omega_i^{\text{old}})$ which can be integrated directly into an update of θ.

Algorithm 12 *Sequential Moment Sharing (synchronous)*

Initialise $\theta, \omega_1, \dots, \omega_K$ such that $\theta = \sum_{i=1}^K \omega_i \in \Omega$
for $k = 1 : N_{\text{EP}}$ **do**
  Send current global parameter θ to each node
  for node $i = 1 : K$ **do**
    Update the cavity parameter $\lambda_i \leftarrow \theta - \omega_i$
    Form an estimator $\hat{\mu}_i$ of $\mathbb{E}_{q_i}[\phi(X)]$
    Update the local parameter $\omega_i \leftarrow \omega_i + \kappa(\nabla A^\star(\hat{\mu}_i) - \lambda_i)$
    Send the updated ω_i to the master node
  end for
  Update global parameter $\theta \leftarrow (1 - \kappa')\theta + \kappa' \sum_{i=1}^K \omega_i$
end for
return θ

6.2.2 Damped updates in the mean parameter space

The main issue with the SMS algorithm which we will expose in more details in the experiments (see section 6.3) is that it performs the projection of the noisy estimator $\hat{\mu}_i$ *before* the damping step. Since this projection operator is in general non-linear and potentially numerically unstable (in the Gaussian case, for example, it requires a matrix inversion) this can hinder the performances of the algorithm. In the case of the Gaussian distribution for instance, the projection effectively requires the inversion of a noisy matrix (see (5.18)). It seems therefore sensible to suggest damping the estimator *before* it gets projected. This leads to going from (6.5) to

$$\nabla A(\theta) \quad \leftarrow \quad (1 - \kappa)\nabla A(\theta) + \kappa \hat{\mu}_i \tag{6.8}$$

or, equivalently, to

$$\theta \quad \leftarrow \quad \nabla A^\star(\nabla A(\theta) + \kappa(\hat{\mu}_i - \nabla A(\theta))). \tag{6.9}$$

Of course, if $\kappa = 1$ (no damping), both approaches are equivalent. Readers familiar with convex optimisation will also notice the similarity with the *mirror-descent algo-*

rithm (Nemirovski and Yudin, 1983; Beck and Teboulle, 2003). We will explore and discuss this similarity later in point 6.2.4. These updates can also be expressed in terms of updates of the local parameters:

$$\omega_i \leftarrow \nabla A^\star(\nabla A(\theta) + \kappa(\hat{\mu}_i - \nabla A(\theta))) - \lambda_i. \tag{6.10}$$

As in (6.6) the role of ω_i and λ_i can be swapped to obtain another valid damped update mechanism. The update (6.10) can be injected into the SMS algorithm 12 to obtain the mean-parameter variant which we will call *MP-EP* to refer to the fact that the updates are done in the Mean Parameter space by contrast to *NP-EP* (SMS) where the updates are done in the Natural Parameter space.

6.2.3 The EP energy perspective

In Minka (2001c), the author presents the LMMC (5.26) as a min-max problem of an objective function the author calls the *EP energy function*. One way to obtain this is to decompose the KL divergence between p and q. To simplify notations over the next few equations we omit the dependence of the functions in x which is obvious from the context; all sums and products are assumed to go over the range $i = 1, \ldots, K$ with K the number of factors. The KL divergence can then be written as

$$\mathrm{KL}(p, q) = \mathbb{E}_p \left[\log \left(Z_p^{-1} \pi_0 \prod_i t_i \right) - \log q \right]. \tag{6.11}$$

Consequently, the product t_i can be manipulated to make the q_i appear:

$$\prod_i t_i = \prod_i \frac{\pi_0 \exp \langle \lambda_i, \phi \rangle t_i}{\pi_0 \exp \langle \lambda_i, \phi \rangle} = \frac{\prod_i Z_i q_i}{\pi_0 (Z_q q)^{K-1},} \tag{6.12}$$

and using this in (6.11) leads to:[2]

$$\mathrm{KL}(p,q) + \log Z_p \;=\; (1-K)A(\theta) + \sum_i A_i(\lambda_i) + \mathbb{E}_p\left[\log q_i/q\right]. \tag{6.13}$$

The first part, denoted by $\mathcal{E}$ defines the *EP energy* with

$$\mathcal{E}(\theta, \lambda_1, \ldots, \lambda_K) \;=\; (1-K)A(\theta) + \sum_{i=1}^{K} A_i(\lambda_i) \tag{6.14}$$

under the constraint that $\theta = (K-1)^{-1}\sum_{i=1}^{K}\lambda_i$. Note that its gradient in λ_i is simply $\nabla_{\lambda_i}\mathcal{E} = (\mu_i - \nabla A(\theta))$. The stationary points of $\mathcal{E}$ in λ_i therefore correspond to the LMMC (5.26) that the EP algorithm attempts to satisfy. It is also interesting to briefly look at the remainder of the right-hand side of (6.11). Discarding the terms that do not depend on λ_i, we have

$$\begin{aligned}\nabla_{\lambda_i}\mathbb{E}_p[\log q_i/q] &= \nabla_{\lambda_i}\mathbb{E}_p\left[\langle\lambda_i,\phi\rangle - A_i(\lambda_i) - \left\langle (K-1)^{-1}\sum_i \lambda_i, \phi\right\rangle + A(\theta)\right] \\ &= \mathbb{E}_p[\phi] - \mu_i - \frac{1}{K-1}\mathbb{E}_p[\phi] + \frac{1}{K-1}\nabla A(\theta).\end{aligned} \tag{6.15}$$

Therefore the gradient of the complete right hand side of (6.11) (i.e. the gradient of $\mathrm{KL}(p,q)$) in λ_i is in fact proportional to $(\mathbb{E}_p[\phi] - \nabla A(\theta))$ leading to the same stationary points as the GMMC (5.21) as could have been expected.

In Heskes et al. (2005), the authors show that the energy (6.14) is in fact equivalent to their *Expectation Consistent free energy*. Further, they indicate that several stationary points can exist and that the framework does not provide a criterion to pick an optimal one. Lastly, they remind the reader that since the energy is neither convex nor

[2] Note that considering the reverse KL leads to a very similar decomposition:

$$\mathrm{KL}(q,p) - \log Z_p \;=\; -\mathcal{E} + \sum_i \mathrm{KL}(q,q_i).$$

concave with respect to the parameters λ_i, it does not tell whether stationary points are minimum, maximum or saddle points.[3]

6.2.4 Mirror Descent for the EP energy

Let us define $\mathcal{E}_i(\lambda_i)$ as the energy (6.14) with all λ_j fixed for $j \neq i$ i.e.:

$$\mathcal{E}_i(\lambda_i) \quad = \quad (1-K)A\left((K-1)^{-1}(\lambda_i + \sum_{j\neq i}\lambda_j)\right) + A_i(\lambda_i) + C_1 \tag{6.16}$$

where $C_1 = \sum_{j\neq i} A_j(\lambda_j)$ does not depend on λ_i. Letting $\eta := (K-1)^{-1}$ and $\theta_{-i} := \eta\sum_{j\neq i}\lambda_j$, we have $-\eta\mathcal{E}_i = A(\eta\lambda_i + \theta_{-i}) - \eta A_i(\lambda_i) + C_2$ with $C_2 = -\eta C_1$. On the corresponding compute node, the aim is then to determine a stationary point λ_i^{new} for $\mathcal{E}_i(\lambda_i)$ which is then equivalent to finding a stationary point for $\mathcal{G}_i(\lambda_i)$ with

$$\mathcal{G}_i(\lambda_i) \quad := \quad A(\theta_{-i} + \eta\lambda_i) - \eta A_i(\lambda_i). \tag{6.17}$$

A common approach to finding stationary points of a function is to consider Newton's method which revolves around a local quadratic approximation of that function. Let us denote λ_i the current parameter and λ_i^{new} the new one. We can consider the following development around λ_i:

$$\begin{aligned}\mathcal{G}_i(\lambda_i^{\text{new}}) \quad \approx \quad & \mathcal{G}_i(\lambda_i) + \eta\left\langle \lambda_i^{\text{new}} - \lambda_i, \nabla A(\theta) - \nabla A_i(\lambda_i)\right\rangle + \\ & \frac{\eta^2}{2}\left\langle \lambda_i^{\text{new}} - \lambda_i, \nabla^2\mathcal{G}_i(\lambda_i)(\lambda_i^{\text{new}} - \lambda_i)\right\rangle.\end{aligned} \tag{6.18}$$

Taking the gradient in λ_i^{new} and setting it to zero then leads to a Newton-like step. However, this requires the inversion of the Hessian of $\mathcal{G}_i$ which may be expensive and numerically unstable to compute. An alternative can be obtained by observing that, if

[3] So in fact, *energy* is a bit of an unfortunate misnomer here.

$\nabla^2\mathcal{G}_i(\lambda_i)$ is positive definite, then the quadratic term in the approximation is simply a measure of discrepancy between λ_i^{new} and λ_i. As in the mirror-descent algorithm (Nemirovski and Yudin, 1983; Beck and Teboulle, 2003) we could therefore try to replace it by a Bregman divergence.[4] In our case, it is convenient to consider the Bregman divergence induced by the log-partition function A itself;[5] this leads to the alternative approximation:

$$\mathcal{G}_i(\lambda_i^{\text{new}}) \quad \approx \quad \mathcal{G}_i(\lambda_i) + \eta \left\langle \lambda_i^{\text{new}} - \lambda_i, \nabla A(\theta) - \nabla A_i(\lambda_i) \right\rangle + \kappa\eta^2 B_A(\lambda_i^{\text{new}}, \lambda_i),$$

where $\kappa > 0$ is a scaling factor. Taking the gradient of that approximation in λ_i^{new} and setting it to zero then leads to a mirror-descent-like step:

$$\lambda_i^{\text{new}} \quad \leftarrow \quad \nabla A^\star \left[\nabla A(\lambda_i) + \kappa'(\hat{\mu}_i - \nabla A(\theta)) \right] \tag{6.19}$$

where $\kappa' = \eta/\kappa$ can be interpreted as a step-size and $\hat{\mu}_i$ is a noisy estimator of $\nabla A_i(\lambda_i)$. By analogy, we can also suggest the corresponding mechanism for an update in terms of the local parameters ω_i:

$$\omega_i^{\text{new}} \quad \leftarrow \quad \nabla A^\star \left[\nabla A(\omega_i) + \kappa''(\hat{\mu}_i - \nabla A(\theta)) \right] \tag{6.20}$$

which resembles the damped updates in the mean-parameter space (6.10) we had obtained earlier (we will show in the experiments section 6.3 that it underperforms compared to the updates (6.10)). More importantly, the updates (6.20) form the core of the *Stochastic Natural gradient Expectation Propagation* (SNEP) algorithm we suggested in Hasenclever et al. (2017). The update (6.20) can be injected into the SMS algorithm 12 to obtain the "SNEP" variant we use in the experiments.

[4] The Mirror Descent algorithm and its connection with the Natural Gradient algorithm is briefly covered in the appendix A.2).

[5] Recall that $B_A(\lambda_i^{\text{new}}, \lambda_i) := A(\lambda_i^{\text{new}}) - A(\lambda_i) - \langle \lambda_i^{\text{new}}, \nabla A(\lambda_i) \rangle$.

6.3 Experiments

In this section, we compare the three main versions of EP algorithms discussed in the chapter: NP-EP where the updates are done in the natural parameter space (the SMS algorithm with update (6.6)), MP-EP (6.10) and SNEP (6.20) where updates are done in the mean parameter space. For the purpose of the experiments, all algorithms are ran in a synchronous fashion.

6.3.1 Description of the experiments

The model considered is a simple Bayesian logistic regression model. Whilst this model is of moderate interest in itself, it has the nice property of being well behaved making the analysis of the behaviour of different inference algorithms easier. This is a similar setup than the one considered in the SMS paper (Xu et al., 2014), generating a dataset $\mathcal{D} = \{(z_c, y_c)\}_{c=1}^N$ with covariates $z_c \in \mathbb{R}^d$ and response $y_c \in \{0, 1\}$. The conditional distribution of y_c given z_c and weights $x \in \mathbb{R}^d$ is

$$p(y_c = 1|z_c, x) \quad = \quad \sigma(\langle z_c, x\rangle) \tag{6.21}$$

where σ denotes the logistic function: $\sigma(s) = (1 + \exp(-s))^{-1}$. A Gaussian prior $p_0(x) = \mathcal{N}(x; 0_d, 10\mathbb{I}_d)$ is set on x and the aim is to construct an approximation to the posterior $p(x|\mathcal{D})$. We generate $N = 5000$ data points with $d = 10$ using iid draws for the covariates, $z_c \sim \mathcal{N}(\mu, \Sigma)$ where the d components of μ are drawn iid from a Uniform $\mathcal{U}[0, 1]$ and $\Sigma = PP^T$ where the components of P are drawn from a Uniform $\mathcal{U}[-1, 1]$. The generating weight vector $x^\star$ is drawn from the prior p_0. The labels y_c are then sampled according to the model.

The data is sharded across five factors (corresponding to 5 compute nodes) each with one fifth of the data. To compute expected values against the tilted distribution,

an importance sampling estimator is constructed with a varying number of samples (5, 10, 50) as shown in figure 6.2. The proposal used is the last global approximation shared with the compute node. Note that more involved samplers can be used (as is done in Hasenclever et al. (2017) which uses a stochastic gradient MCMC method) but the purpose here is primarily to explore the robustness of the algorithms to Monte Carlo noise so that a simple sampler is adequate.

All algorithms use a fixed damping throughout the iterations and different dampings were used to exhibit the behaviour of the algorithm (see 6.2). All experiments were run 5 times and averaged over those runs to account for the inherent stochasticity of the algorithms.

As the base exponential family, we use a full-covariance Gaussian. We compare the predictive RMSE $\sqrt{\sum_c |\hat{y}_c - y_c|^2/N}$ obtained with $\hat{y}_c = \sigma(\langle z_c, \overline{x}\rangle)$ where $\overline{x}$ is the currently estimated posterior mean of the global approximation. As a reference point, we also show the RMSE corresponding to $x^\star$.

6.3.2 Results

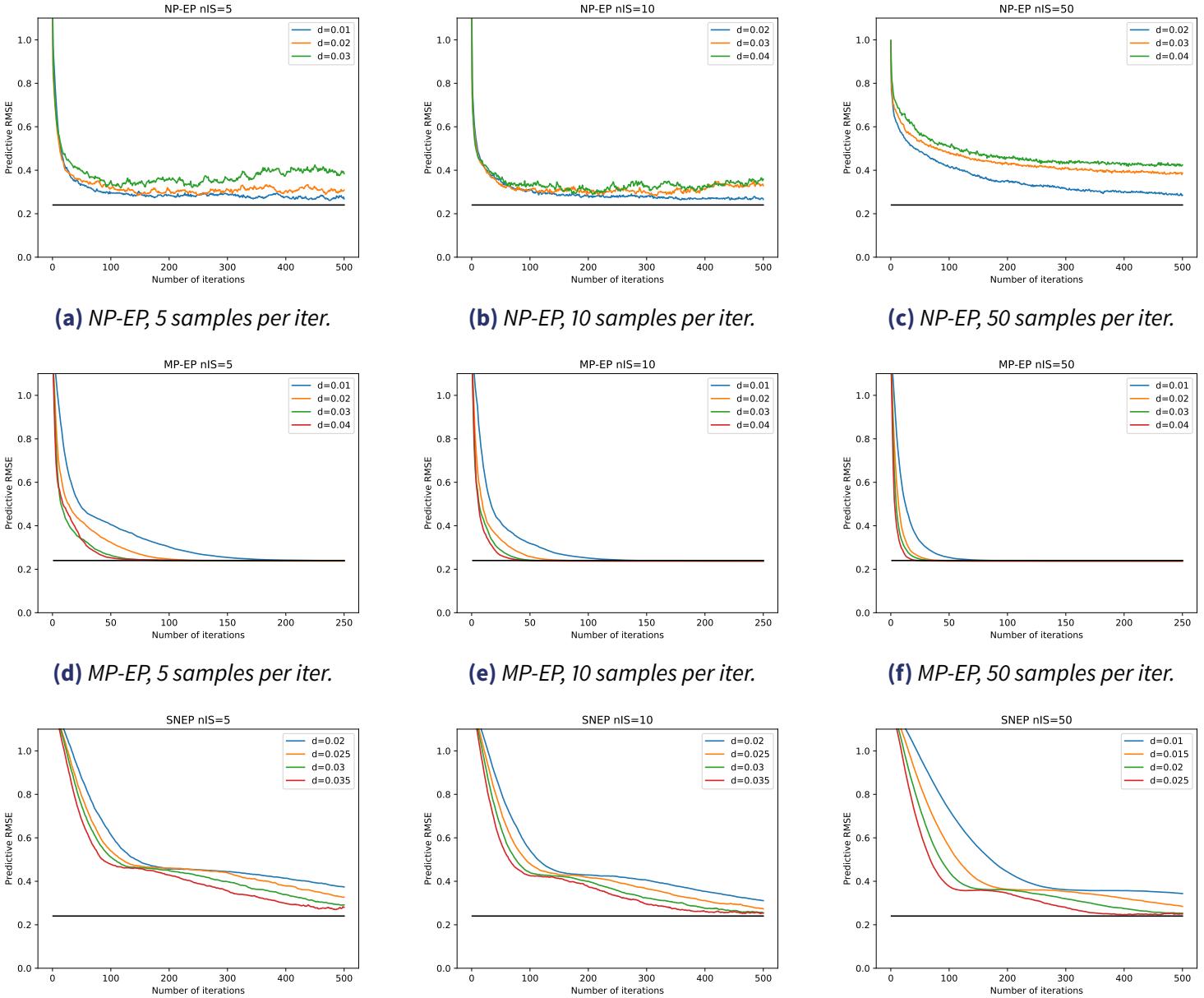
We show in figure 6.2 the evolution of the predictive RMSE with increasing number of iterations for the three algorithms considered when varying the number of iteration per samples (corresponding to the amount of Monte Carlo noise) and the damping parameter (indicated by d in the legends). Note that the computational complexity of the iteration for each of the three algorithm is very similar, the main difference residing in the ordering of operations.[6] As a baseline, the predictive RMSE corresponding to $x^\star$ (the generating set of weights) is drawn as a black line on each graph.

A range of damping parameters is selected to illustrate the behaviour of the algorithm. The range is always taken with the largest damping parameter being on the

[6] In all the runs, an iteration of MP-EP is around 5% faster than an iteration of NP-EP and around 20% faster than an iteration of SNEP but that is negligible compared to the factor of 2 speedup obtained by the much faster convergence of the algorithm.

verge of numerical instability for the algorithm. Indeed, when using a full-covariance Gaussian as base exponential family, going from the mean-parameter space to the natural parameter space or vice versa requires the inversion of a matrix. Doing the update in the mean parameter space helps the numerical stability of this step as does increasing the damping (reducing the step-size).

It is clear from the results that the best performing algorithm is the MP-EP algorithm which converges much faster than SNEP and is much more stable than NP-EP. It is also the algorithm that offers the most robust performance in the presence of Monte Carlo noise. Finally, this algorithm corresponds to a simpler mathematical development than SNEP by being simply a mirror-descent version of the NP-EP updates.



(a) *NP-EP, 5 samples per iter.* **(b)** *NP-EP, 10 samples per iter.* **(c)** *NP-EP, 50 samples per iter.*

(d) *MP-EP, 5 samples per iter.* **(e)** *MP-EP, 10 samples per iter.* **(f)** *MP-EP, 50 samples per iter.*

(g) *SNEP, 5 samples per iter.* **(h)** *SNEP, 10 samples per iter.* **(i)** *SNEP, 50 samples per iter.*

Figure 6.2: *Results of the EP experiments. Each curve corresponds to a different damping parameter indicated by d, taken in the range as close to numerical instability as possible. (Figures best seen in colour).*

6.4 Discussion

In this chapter we showed that Expectation Propagation could be used as the backbone of a distributed approximate Bayesian inference mechanism. We showed that different variant of the EP algorithm could be used and that, in our experiments, performing updates in the mean parameter space can outperform updates in the natural parameter space. In particular we showed that updates in the mean parameter space can be much more robust to noise introduced by following inexact EP updates when the moments of the tilted distribution are computed via Monte Carlo estimators. This is particularly important for complex models where the tilted distributions are complex and sampling is very expensive so that cheap Monte Carlo estimators based on a few sampling points are favoured (e.g.: graphical models (Heess et al., 2013; Eslami et al., 2014; Jitkrittum et al., 2015), hierarchical Bayesian models (Gelman et al., 2014) and approximate Bayesian computation (Barthelme and Chopin, 2011)).

Since Minka (2001a), there have been a substantial number of extensions and alternatives to EP proposed. Stochastic EP (Li et al., 2015) and averaged EP (Dehaene and Barthelme, 2015) assume that all factors can be well approximated by the *same* exponential family factor via parameter tying. This saves memory storage and was shown to work well when a lot of samples are available. This is an orthogonal idea to our work, and it is possible to apply it to SNEP and MP-EP as well although in the distributed learning setting considered, each worker must keep a copy of the parameters anyway which means it would not reduce memory requirements. Convergent EP (Heskes and Zoeter, 2003) is a similar algorithm with (idealised) convergence guarantees but it is extremely slow to run in practice.

A critique often raised against EP is that the algorithm does not offer convergence guarantees. In fact, Minka addresses this in his paper (Minka, 2001a) indicating that when EP does not converge it is usually a sign of a poor choice of approximating fam-

ily $\mathcal{F}_\phi$. Further to this, the convergence question may in fact be of little practical use since, in any case, we do not know how good an approximation the fixed point corresponding to the local moment matching conditions (LMMC) is to the fixed point corresponding to the global ones (GMMC) nor how good a proxy the latter is to the target distribution. In fact, a similar criticism can be raised for Mean Field Variational Inference: the iterations provably decrease an energy and converge but it is unclear how good a proxy it converges to. This situation is summarised in a conceptual manner in figure 6.3.

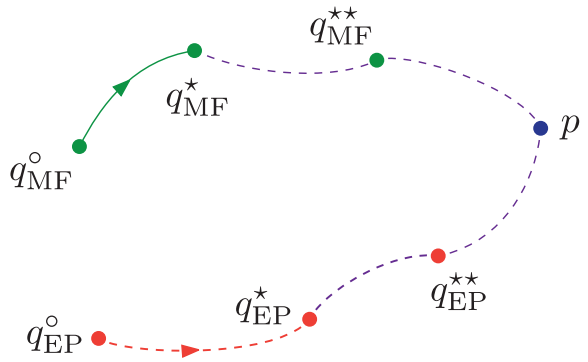


Figure 6.3: *Toy representation of the approximating distributions obtained through MFVI or EP when targeting a distribution p of interest. The starting points are denoted with a superscript $\circ$. MFVI provably converges to a point $q^{\star}_{MF}$ (solid green line) but we do not know how this point compares to a distribution $q^{\star\star}_{MF}$ that truly minimises the corresponding KL divergence.*
By contrast, EP is not guaranteed to converge to a fixed point $q^{\star}_{EP}$ (dashed red line) and we also do not know how it compares to $q^{\star\star}_{EP}$ that truly minimises the corresponding KL divergence.
In both cases, we do not really know, in general, how the minimisers obtained compare to the target distribution p (dashed purple lines). This graph is best viewed in colours.

Finally, we note that the entirety of the EP variants discussed in this section can be re-expressed in the general framework of "Power-EP" (Minka, 2004) where the discrepancy is more general than the basic Kullback-Leibler divergence (Hernandez-Lobato et al., 2015; Li and Turner, 2016). This, however, has little impact to the considerations discussed in the chapter in that the only element it changes is the definition of the projection operator $\mathbf{P}_\phi$. We focused on "classical" EP for this reason and also because, to

the best of our knowledge, the role of the exponent in Power-EP is poorly understood in general.

7 | Expectation Propagation for Particle Belief Propagation

In this chapter, we look at ways to implement the loopy belief propagation (LBP) algorithm on an arbitrary MRF with continuous state-spaces. We concentrate in particular on the representation of the messages and the computation of message updates in the LBP algorithm.

In this chapter, we consider the *nonparametric belief propagation* (NBP) algorithm of Sudderth et al. (2003) and the *particle belief propagation* algorithm of Ihler and McAllester (2009). We then discuss our *expectation particle belief propagation* (EPBP) algorithm (Lienart et al., 2015) and discuss how it can improve upon both.

7.1 Loopy Belief Propagation on Continuous State-Spaces

At point 5.4.3, we had shown that, at iteration n of the LBP algorithm, the messages are obtained by computing the following integral:

$$m_{ts}^n(x_s) \quad = \quad \int \psi_{st}(x_s, x_t) M_{ts}^n(x_t) \, \mathrm{d}x_t \tag{7.1}$$

where the pre-messages are given by

$$M_{st}^n(x_s) \quad = \quad \psi_s(x_s) \prod_{r \in \partial s \setminus t} m_{rs}^{n-1}(x_s). \tag{7.2}$$

Ultimately, we are interested in computing the beliefs obtained by multiplying message and pre-message: $B_s^n(x_s) = m_{ts}^n(x_s) M_{st}^n(x_s)$. There are a few main computational problems to tackle when attempting to run this algorithm in a continuous state space. First, the messages need to be represented in a tractable fashion. Second, the message updates need to be computable and the results easily expressible in the chosen representation system. Lastly, we need to be able to easily compute expected values with respect to the resulting estimators for the beliefs. We discuss below two existing methods attempting to tackle those issues.

7.1.1 Nonparametric Belief Propagation

In Sudderth et al. (2003), the authors suggest representing the messages in the LBP iterations as mixtures of Gaussians:

$$\widehat{m}_{ts}^{\mathsf{NBP}}(x_s) \quad := \quad \sum_{i=1}^{M} w_s^i \, \mathcal{N}(x_s; \mu_s^i, \Lambda_s). \tag{7.3}$$

They call their algorithm *nonparametric belief propagation* (NBP). In order to make computations more efficient, the authors restrict the covariance matrices to be diagonal. As indicated in the paper, the representation (7.3) makes sense only when the messages are finitely integrable. To guarantee this, the authors require that all potentials satisfy the following constraints:

$$\sup_{x_t} \int \psi_{st}(x_s, x_t) \, \mathrm{d}x_t \quad < \quad \infty, \quad \text{and} \quad \int \psi_s(x_s) \, \mathrm{d}x_s \quad < \quad \infty, \tag{7.4}$$

where the integrals are taken over the range of admissible values for the relevant variable. Provided these assumptions hold, the message updates are well defined and can be explicitly (and efficiently) computed. If these conditions are not met however the message approximations are not well defined and the iterations may explode. Computing the beliefs however, requires considering the product of mixtures of M terms leading to a complex representation and an explosion in computational cost. In order to alleviate this, the authors suggest an importance sampling approach targeting the beliefs and fitting mixtures of Gaussians to the resulting weighted particles. The computation of the message updates (7.1) is thereby always done over a constant number of terms.

Main issues

A key weakness of the NBP algorithm is that the conditions (7.4) do not hold in a number of important cases (the authors aknowledge this in a footnote of their paper). First, the node potentials ψ_u are usually proportional to likelihoods of the form $p(y_u \mid x_u)$ which need not be integrable in x_u. In fact, for most non-Gaussian potentials, this is the case. Then, there exist applications for which the first integrability condition on edge potentials does not hold. In imaging applications for example, the edge potential can encode a measure of similarity between pixels which need not verify the first integrability condition as in Nikolova (2000). Finally, by definition of NBP as an approximated representation with a fixed number of Gaussians, it will typically lead to biased estimators of the LBP messages.

7.1.2 Particle Belief Propagation

In Ihler and McAllester (2009), the authors suggest a way to overcome the shortcomings of NBP by considering importance sampling to tackle the update of the LBP messages instead of working with mixtures of Gaussians. For a chosen proposal distribu-

tion q_u on node u and a draw of N particles $X_u^{(i)} \sim_{\text{iid}} q_u$, the messages are represented via an importance sampling estimator of the corresponding integral:

$$\widehat{m}_{st}^{\text{PBP}}(x_t) := \sum_{i=1}^{N} w_{st}^{(i)} \psi_{st}(X_s^{(i)}, x_t), \quad \text{where} \tag{7.5}$$

$$w_{st}^{(i)} \propto \frac{\widehat{M}_{st}^{\text{PBP}}(X_s^{(i)})}{q_s(X_s^{(i)})}, \quad \text{and} \quad \widehat{M}_{st}^{\text{PBP}}(x_s) = \psi_s(x_s) \prod_{r \in \partial s \setminus t} \widehat{m}_{rs}^{\text{PBP}}(x_s).$$

They call their algorithm *particle belief propagation* (PBP). This algorithm has the advantage that it does not explicitly require the integrability conditions (7.4) to hold. In terms of choosing proposals, the authors suggest two potential choices: sampling from the local potential ψ_s, or sampling from the current belief estimate on the node.

Main issue

The weak point of the PBP algorithm is the choice of proposal distributions. A poor choice will lead to poor estimators of the messages which, over a few iterations of the LBP algorithm, can lead to poor representations of the beliefs.

The first suggestion by the authors to sample from the local potential is only valid if ψ_s is integrable which, as we have mentioned earlier, is not the case in general. The second suggestion implies sampling from a distribution of the form

$$\widehat{B}_s^{\text{PBP}}(x_s) \propto \psi_s(x_s) \prod_{r \in \partial s} \widehat{m}_{rs}^{\text{PBP}}(x_s) \tag{7.6}$$

which is a product of mixtures of N components. As in nonparametric BP, naive sampling of the proposal has complexity $\mathcal{O}(N^{|\partial s|})$ and is thus, in general, too expensive to consider.

Alternatively the authors suggest running a short MCMC simulation targeting it which reduces the complexity to order $\mathcal{O}(|\partial s| N^2)$. Indeed, each MCMC iteration requires evaluating $\widehat{B}_s^{\text{PBP}}$ point-wise which has complexity $\mathcal{O}(|\partial s| N)$, and we need at

least $\mathcal{O}(N)$ iterations of the MCMC simulation to produce the samples. Note that it is unclear how many more iterations are necessary to get N *good* samples. Running a short MCMC chain (to reduce computational cost) will therefore almost certainly lead to biased samples. In the code the authors shared with us, they run N parallel random walk Metropolis-Hastings chains (Robert and Casella, 2004, chapter 6) for a fixed number of steps. This choice may have been motivated by a desire to control the complexity of the algorithm more accurately but it is at the expense of the quality of the samples obtained.

7.2 Expectation Particle Belief Propagation

As for the PBP algorithm, we consider importance sampling to build particle representations of the messages. In our method, however, we consider a *sequence* of proposal distributions at each node from which one can cheaply sample particles at a given iteration of the LBP algorithm (Lienart et al., 2015).

The novelty of the approach is to propose a principled and automated way of designing a sequence of proposals in a tractable exponential family using the expectation propagation (EP) algorithm. The resulting method, which we call *Expectation Particle Belief Propagation* (EPBP), does not suffer from the restrictive integrability conditions of the NBP algorithm and sampling is done exactly unlike the PBP algorithm which implies that we obtain proper importance sampling estimators of the LBP messages with almost sure convergence guarantees.

Further, the development of our method also formally shows that considering proposals close to the beliefs, as suggested by Ihler and McAllester (2009), is a good idea. Our core observation is that since sampling from a proposal of the form (7.6) using MCMC simulation is very expensive, we should consider using a more tractable proposal distribution instead. However it is important that the proposal distribution is

constructed adaptively, taking into account evidence collected through the message passing itself. We propose to achieve this by using proposal distributions lying in a tractable exponential family, and adapted using EP.

7.2.1 Proposal selection for PBP

In this point we discuss the ideal choice of proposal distributions when considering the PBP algorithm. We consider two connected nodes s and t at a a given iteration and assume we have already constructed particle based representations of all incoming messages into both s and t apart from the messages from s to t and from t to s. We therefore have both pre-messages $\widehat{M}_{st}$ and $\widehat{M}_{ts}$ of the form:

$$\widehat{M}_{st}(x_s) \propto \psi_s(x_s) \prod_{r \in \partial s \backslash t} \widehat{m}_{rs}(x_s). \tag{7.7}$$

This is illustrated in figure 7.1.

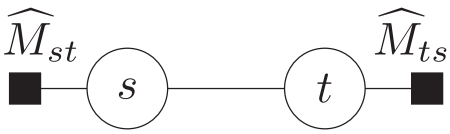


Figure 7.1: *Illustration of the situation for an edge (s,t): all messages coming into s and t have been approximated apart from the messages between s and t. Therefore approximated pre-messages (denoted $\widehat{M}$) are available on both nodes. We are considering the problem of determining the best proposals on s and t in order to approximate the messages between s and t given the situation.*

We would like to define q_s and q_t in a joint manner in such a way that all objects remain coherent with the LBP iterations. For that reason, we consider the joint belief B_{st} on s and t given the approximation to the pre-messages:

$$B_{st}(x_s, x_t) \propto \widehat{M}_{st}(x_s)\psi_{st}(x_s, x_t)\widehat{M}_{ts}(x_t). \tag{7.8}$$

The marginals of the joint belief are of the form

$$\begin{aligned} B_{st}(x_s) &\propto \widehat{M}_{ts}(x_s) \int \psi_{st}(x_s, x_t) \widehat{M}_{ts}(x_t) \, \mathrm{d}x_t \\ &\propto \widehat{M}_{st}(x_s) m_{ts}(x_s), \end{aligned} \tag{7.9}$$

where m_{ts} is the exact (but intractable) LBP message given the pre-message approximation $\widehat{M}_{ts}$. Following our point about defining q_s and q_t in a joint manner, let us consider $q_s q_t$ as a proposal for the joint belief B_{st}. We can then define an empirical distribution for it:

$$\widehat{B}_{st}(x_s, x_t) \propto \sum_{i,j=1}^{N} \frac{\widehat{M}_{st}(X_s^{(i)}) \psi_{st}(X_s^{(i)}, X_t^{(j)}) \widehat{M}_{ts}(X_t^{(j)})}{q_s(X_s^{(i)}) q_t(X_t^{(j)})} \delta_{(X_s^{(i)}, X_t^{(j)})}(x_s, x_t). \tag{7.10}$$

Marginalising this approximation over x_t leads to

$$\widehat{B}_{st}(x_s) \propto \sum_{i=1}^{N} \frac{\widehat{M}_{st}(X_s^{(i)}) \widehat{m}_{ts}(X_s^{(i)})}{q_s(X_s^{(i)})} \delta_{X_s^{(i)}}(x_s), \tag{7.11}$$

where $\widehat{m}_{ts}$ is the importance sampling estimator for m_{ts} using q_t as proposal. Of course, marginalising (7.10) over x_s leads to an expression analogous to (7.11). Focusing on node s, the expression (7.11) indicates that the proposals should be given by

$$q_s(x_s) \propto \widehat{M}_{st}(x_s) \widehat{m}_{ts}(x_s) = \psi_s(x_s) \prod_{r \in \partial s} \widehat{m}_{rs}(x_s) \tag{7.12}$$

where all $\widehat{m}_{rs}$ are importance sampling estimators built using the corresponding most recent q_r as a proposal. Further, this shows that the proposals on nodes should correspond to the most recent approximation of the belief at that node which aligns with the suggestion in Ihler and McAllester (2009).

7.2.2 The EPBP algorithm

As we showed in point 7.1.2, it is computationally expensive to use the particle approximated node belief as the proposal distribution. Our idea is therefore consider an approximation q_s drawn from a tractable exponential family $\mathcal{F}_\phi$ with a structure matching that of the belief ($B_s(x_s) = \psi_s(x_s) \prod_{r \in \partial s} m_{rs}(x_s)$):

$$q_s(x_s) \quad \propto \quad \eta_{\circ s}(x_s) \prod_{r \in \partial s} \eta_{rs}(x_s). \tag{7.13}$$

In the experiments we used a Gaussian family but the algorithm is – in theory – not limited to this choice. Using the framework of expectation propagation (EP) that we discussed in point 5.3.3, we can iteratively find good proxies to the node belief in the exponential family.

For each $r \in \partial s$, we update the η_{rs} following the EP framework. We start by forming the *cavity* $q_s^{\backslash r} = q_s / \eta_{rs}$ and the corresponding *tilted distribution* proportional to $\widehat{m}_{rs} q_s^{\backslash r}$. The updated distribution is then the projection of the tilted distribution onto the exponential family manifold:

$$q_s \quad \leftarrow \quad \mathbf{P}_\phi[\widehat{m}_{rs} q_s^{\backslash r}], \tag{7.14}$$

where $\mathbf{P}_\phi$ is the projection operator defined in section 5.20. Consequently, the factor η_{rs} is updated: $\eta_{rs} \leftarrow q_s / q_s^{\backslash r}$. Note that projection mechanisms that project with respect to divergences other than the Kullback-Leibler divergence can be considered as well.

In the algorithm as we have described it so far, the EP steps for each incoming message into s and for the node potential are performed first in order to fit the proposal to the current estimated belief at s. Then, this estimated belief is used to draw N particles which can be used to form the particle approximated messages from s to each

of its neighbours. Alternatively, once each particle approximated message $\widehat{m}_{st}(x_t)$ is formed, we can update their exponential family projections $\eta_{st}(x_t)$ immediately. This is the scheme described in algorithm 13.

Algorithm 13 *Node update in the EPBP algorithm*

1: sample $X_s^{(i)}$ from q_s for $i = 1, \dots, N$
2: evaluate the approximate representation of the belief:

$$\widehat{B}_s(X_s^{(i)}) = \psi_s(X_s^{(i)}) \prod_{t \in \partial s} \widehat{m}_{ts}(X_s^{(i)})$$

3: **for** $t \in \partial s$ **do**
4: evaluate the approximate representation of the pre-message:

$$\widehat{M}_{st}(X_s^{(i)}) := \widehat{B}_s(X_s^{(i)}) / \widehat{m}_{ts}(X_s^{(i)})$$

5: compute the corresponding normalised weights:

$$w_{st}^{(i)} \propto \widehat{M}_{st}(X_s^{(i)}) / q_s(X_s^{(i)})$$

6: update the estimator of the outgoing message:

$$\widehat{m}_{st}(x_t) = \sum_{i=1}^{N} w_{st}^{(i)} \psi_{st}(X_s^{(i)}, x_v)$$

7: update $\eta_{\circ t}$ to match the update $q_t \leftarrow \mathbf{P}_\phi[\psi_t q_t / \eta_{\circ t}]$
8: update η_{st} to match the update $q_t \leftarrow \mathbf{P}_\phi[\widehat{m}_{st} q_t / \eta_{st}]$
9: **end for**

To clarify the steps of the previous algorithm, we distinguish two phases.

Phase 1 of algorithm 13

In the first phase, new samples $X^{(i)}$ on node s are drawn from the current proposal q_s. Consequently, their importance weights must be obtained which requires computing the $\widehat{B}_s(X_s^{(i)})$ and $\widehat{B}_s$ is obtained by evaluating the product of the message estimators coming into s at those samples as illustrated on figure 7.3 below.

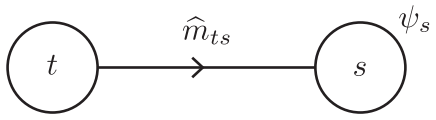


Figure 7.2: *Illustration of the first phase of the node update. All incoming message estimators are evaluated at the new samples $X_s^{(i)}$ and multiplied along with the value at ψ_s to form the $\widehat{B}_s(X_s^{(i)}$.*

The message estimators are mixtures corresponding to the current weighted population of samples along the edge $\{(X_t^{(j)}, w_{ts}^{(j)})\}_{j=1}^N$ i.e.: $\widehat{m}_{ts}(x_s) = \sum_{j=1}^N w_{ts}^{(j)} \psi_{ts}(X_t^{(j)}, x_s)$.

Phase 2 of algorithm 13

In the second phase, the approximations for the outgoing messages $\widehat{m}_{st}(x_t)$ for $t \in \partial s$ (resp. the prior) are formed as well as their EP representation η_{st} (resp $\eta_{\circ t}$). For this, the importance weights along each edge corresponding to the new population of samples need to be computed as per (7.5) Once the weights for the outgoing message

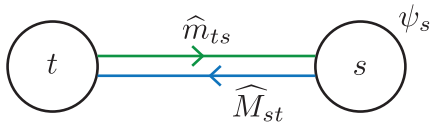


Figure 7.3: *Illustration of the first part of the second phase of the node update. All outgoing pre-message estimators are evaluated at the new samples $X_s^{(i)}$ and divided by the proposal q_s to get the importance weights.*

have been computed, a new approximation for the outgoing message is defined. This mixture can then be projected onto the exponential family $\mathcal{F}_\phi$ in order to get its new EP representation η_{st} which will be used to update the proposal on the neighbouring node t as per (7.13) (same for the prior).

Note that, in the EP step, if the projection leads to an invalid distribution (e.g., in the Gaussian case, if it leads to an element with negative variance), we simply revert the approximation to its previous value as in Minka (2001a).

7.2.3 Projection mechanisms

In the description above, we used the projection $\mathbf{P}_\phi$ from 5.20. Computing this projection requires computing the moments of the tilted distribution. Since this is usually intractable, we have to resort to numerical quadrature. In our scenario the moment computation can be performed crudely on a small number of integration points since it only concerns the updating of the importance sampling proposal. Since the dimensionality of the tilted distribution in the EPBP algorithm is typically low, a simple deterministic quadrature rule can be used (Davis and Rabinowitz, 1975).

The key desired element for the projection is that the resulting updated proposal distribution captures the support of the current estimator of the belief on the node. Since this is a rather general requirement, the KL-projection $\mathbf{P}_\phi$ does not necessarily lead to the best proposals and in fact may suffer from the fact that we have to compute the projection from the mean-parameter space to the natural parameter space $\nabla A^\star$ associated with the exponential family of interest which may be numerically unstable. This can be mitigated by damping as described at section 6.2 but alternative projection schemes can be tried as well. In our experiments for example, we tried considering the $q_s \in \mathcal{F}_\phi$ obtained by doing a maximum likelihood estimation of the natural parameters based on a small number of evaluation points of the tilted distribution. This proved to work well in practice and can be numerically more stable than the original KL-projection.

7.2.4 Computational complexity

The steps that dominate in terms of computational complexity are the evaluation of the approximate representation of the pre-message. Since the estimator for the belief is a product of $|\partial s|$ mixtures of N components, evaluating at N sampling points is $\mathcal{O}(|\partial s|N^2)$. Further, the evaluation of the pre-message requires evaluating the mes-

sage $\widehat{m}_{ts}$ at N sampling points. Since $\widehat{m}_{ts}$ is a mixture of N components, this also has quadratic complexity. The dominating complexity of the EPBP algorithm is therefore $\mathcal{O}(KEN^2)$ where K is the number of passes done over the graph, and E is the number of edges in the graph.

This inherent quadratic complexity can be reduced. Indeed, instead of we can follow a method presented in Briers et al. (2005) to sample from a product of mixtures where one draws, for each mixture $\widehat{m}_{st}$, a set of M indices $\{i_\ell^\star\}_{\ell=1}^M$ from a multinomial with weights $\{w_{st}^i\}_{i=1}^N$ and samples from a mixture of the corresponding M terms $\psi_{st}^{i_\ell^\star}$. This reduces the cost of the evaluation of the belief to $\mathcal{O}(|\partial s|MN)$ where M is taken to be sub-quadratic in N. We show in the next section that this version of the algorithm still compares favourably to the quadratic implementation with $M = \mathcal{O}(\log N)$. Finally, note that this simplification leads to a biased algorithm but, in our experiments, the bias seems smaller than the ambient noise making this still a practical heuristic to use.

7.3 Experiments

We investigate the performance of our method on two simple graphs. This allows us to compare the performance of EPBP to the performance of PBP in depth. We also illustrate the behaviour of the sub-quadratic version of EPBP. Finally we show that EPBP provides good results in a simple denoising application, an example on which the PBP implementation of Ihler and McAllester (2009) would struggle to give results in a reasonable time due to high dimensionality.

7.3.1 Grid and tree experiments

We start by comparing EPBP to PBP as implemented by Ihler and McAllester (2009) on a 3×3 grid (see figure 7.4 (left)) with random variables taking values on $\mathbb{R}$. The

node and edge potentials are selected such that the marginals are multimodal, non-Gaussian and skewed with

$$\begin{cases} \psi_u(x_u) & = & \alpha_1 \mathcal{N}(x_u - y_u; \mu_1, \sigma_1) + \alpha_2 \mathcal{G}(x_u - y_u; \mu_2, \beta) \\ \psi_{uv}(x_u, x_v) & = & \mathcal{L}(x_u - x_v; \mu_3, \sigma_3) \end{cases} \tag{7.15}$$

where y_u denotes the observation at node u, $\mathcal{N}(x; \mu, \sigma) \propto \exp(-x^2/2\sigma^2)$ (density of a Normal distribution), $\mathcal{G}(x; \mu, \beta) \propto \exp(-[(x-\mu)/\beta + \exp(-(x-\mu)/\beta)])$ (density of a Gumbel distribution) and $\mathcal{L}(x; \mu, \beta) \propto \exp(-|x-\mu|/\beta)$ (density of a Laplace distribution). The parameters were set (arbitrarily) to:

$$\alpha_1 = 0.6,\ \alpha_2 = 0.4,\ \mu_1 = -2,\ \mu_2 = 2,\ \mu_3 = 0,\ \sigma_1 = 1,\ \sigma_2 = 2,\ \beta = 1.3.$$

The choice of model was mainly driven by showing the performance on a controllable yet mildly multimodal and non-Gaussian setting. The model for the edge potential corresponds to a distribution around an ℓ_1 discrepancy between neighbouring nodes. While simple, the model already allows to clearly show that PBP underperforms as compared to EPBP.

We compare the two methods after 20 LBP iterations.[1] We then repeated the experiments on a tree with 8 nodes (figure 7.4 (right)) where we know that, at convergence, the beliefs computed using BP are proportional to the true marginals. Again, the node and edge potentials are picked such that the marginals are multimodal with this time

$$\begin{cases} \psi_u(x_u) & = & \alpha_1 \mathcal{N}(x_u - y_u; \mu_1, \sigma_1) + \alpha_2 \mathcal{N}(x_u - y_u; 1\mu_2, \sigma_2) \\ \psi_{uv}(x_u, x_v) & = & \mathcal{L}(x_u - x_v; \mu_3, \sigma_3) \end{cases} \tag{7.16}$$

[1] The scheduling used alternates between the classical orderings: top-down-left-right, left-right-top-down, down-up-right-left and right-left-down-up. One "LBP iteration" implies that all nodes have been updated once.

with the parameters also set in an arbitrary fashion to:

$$\alpha_1 = 0.3,\ \alpha_2 = 0.7,\ \mu_1 = -2,\ \mu_2 = 1,\ \mu_3 = 0,\ \sigma_1 = 1,\ \sigma_2 = 0.5,\ \sigma_3 = 1.$$

On this second example, we also considered "pure EP" with Gaussians to compare with EPBP and PBP.

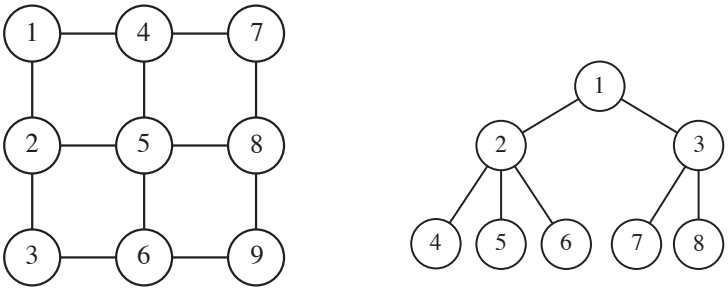


Figure 7.4: *Illustration of the grid (left) and tree (right) graphs used in the experiments.*

PBP as presented in Ihler and McAllester (2009) is implemented using the same parameters than those in an implementation code provided by the authors: the proposal on each node is the last estimated belief and sampled with a 20-step MCMC chain, the Metropolis Hastings proposal is a normal distribution. For EPBP, the approximation of the messages are Gaussians. The ground truth is approximated by running LBP on a deterministic equally spaced mesh with 200 points. All simulations were run with Julia on a Mac with 2.5 GHz Intel Core i5 processor, our code is available online.[2] All the experiments were run multiple times as shown on the figures to account for the inherent stochasticity of the methods. The mean trends are also drawn.

Figure 7.5 compares the performances of both methods. The error is computed as the mean absolute error (MAE) over all nodes between the estimated beliefs and the

[2] https://github.com/tlienart/EPBP.

ground truth using the same equally spaced mesh used for the ground truth i.e.:

$$\text{MAE} \quad = \quad |\mathcal{V}|^{-1} \sum_{i \in \mathcal{V}} \sum_{j=1}^{200} \left| \widehat{B}_i(x_j) - B_i^{\text{GT}}(x_j) \right| \tag{7.17}$$

where the x_j are the grid points.

One can observe that not only does PBP perform worse than EPBP but also that the error plateaus with increasing number of samples. This is because the sampling within PBP is done approximately and hence it does not correspond to a proper importance sampling estimator which means any convergence guarantee is lost. Note that for EPBP, one observes the expected $1/\sqrt{N}$ convergence of particle methods discussed in Ihler and McAllester (2009).

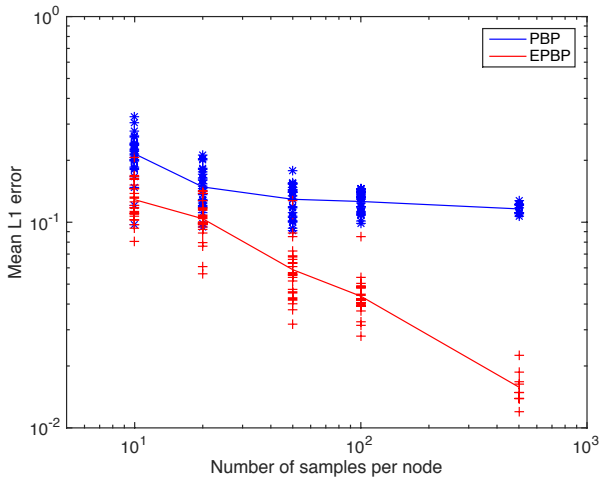


Figure 7.5: *Comparison of the mean L^1 error for PBP and EPBP for the 3×3 grid example. EPBP is more accurate for the same number of samples.*

Figure 7.6 compares the estimator of the beliefs obtained by the two methods for three representative nodes (node $1, 5$ and 9 as illustrated in figure 7.4 (left)). The figure also illustrates the last proposals constructed with our approach and one can notice that their supports match closely the support of the true beliefs.

The speed-up offered by EPBP is very substantial as can be seen in figure 7.7 (left). Hence, although it would be possible (and sensible) to use more MCMC iterations within

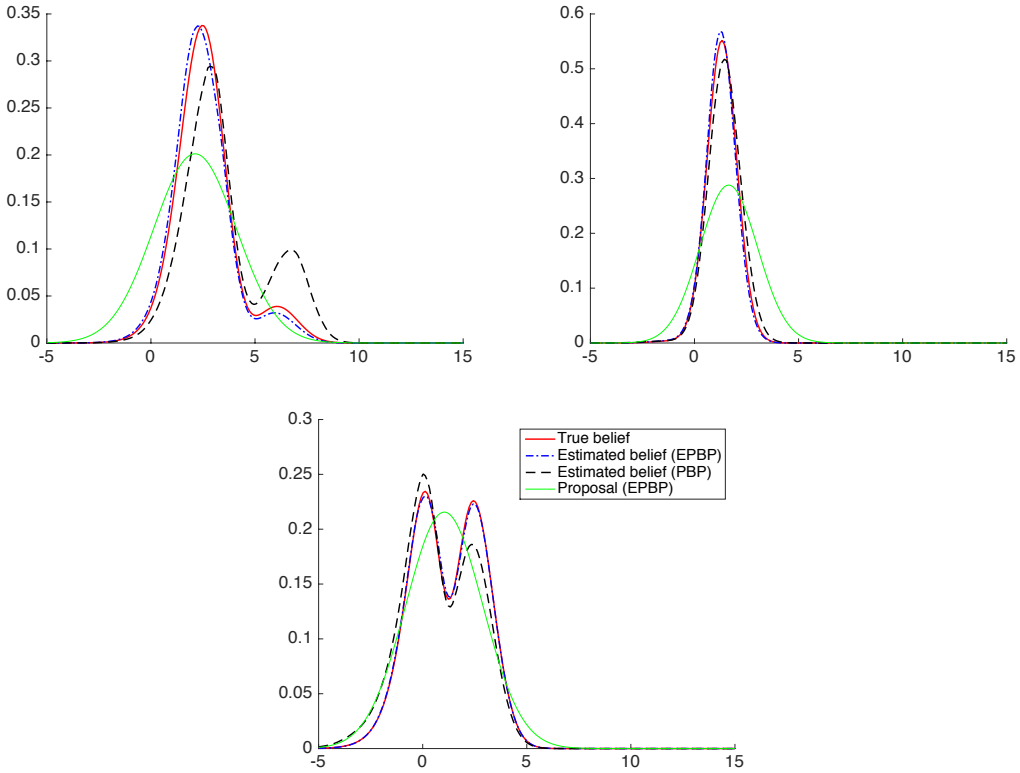


Figure 7.6: *Comparison of the beliefs on node* 1, 5 *and* 9 *(top left, top right, bottom) as obtained by evaluating LBP on a deterministic mesh (*true belief*), with PBP and with EPBP for the* 3×3 *grid example. All three plots share the same legend. The proposal used by EPBP in the last step is also illustrated. The results are obtained with* $N = 100$ *samples on each node and* 20 *BP iterations. One can observe visually that EPBP outperforms PBP.*

PBP to improve its accuracy, it would make the method prohibitively expensive to use compared to EPBP. Figure 7.7 (right) illustrates how the estimated beliefs converge as compared to the true beliefs with increasing number of loopy belief propagation iterations. One can observe that PBP converges more slowly and that the results display more variability which is likely also due to the MCMC runs being too short.

The same experiments were also run on the tree example with similar results. Additionally, we looked at how "pure EP" with normal distributions performs. We also tried using the distributions obtained with EP as proposals for PBP (referred to as "PBP after EP" in figures). All those methods underperform compared to EPBP. In particu-

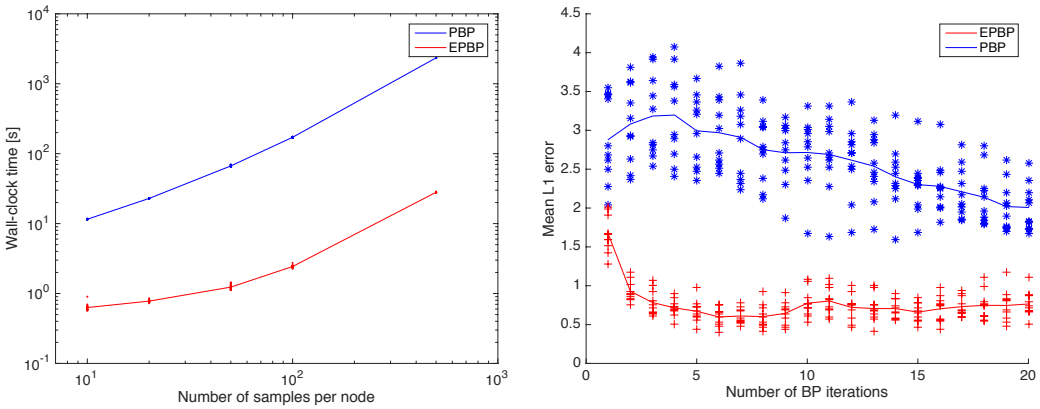


Figure 7.7: *(**left**) Comparison of the wall-clock time needed to perform PBP and EPBP on the 3×3 grid example. (**right**) Comparison of the convergence in L^1 error with increasing number of BP iterations for the 3×3 grid when using $N = 30$ particles.*

lar one can observe in figure 7.8 that "PBP after EP" converges slower than EPBP with increasing number of samples.

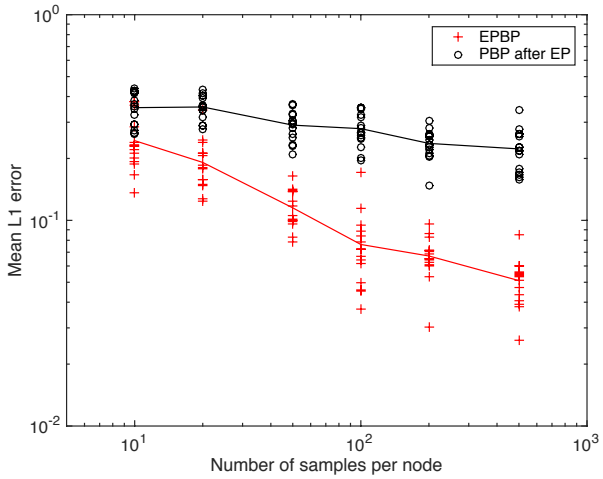


Figure 7.8: *Comparison of the mean L^1 error for EPBP and "PBP after EP" for the tree example. EPBP is more accurate and converges faster.*

Figure 7.9 compares the estimator of the beliefs obtained by all the methods on the tree example for three representative nodes (node $1, 3$ and 8 as illustrated in figure 7.4 (right)). As for the grid example, the figure illustrates how EPBP better recovers the true beliefs.

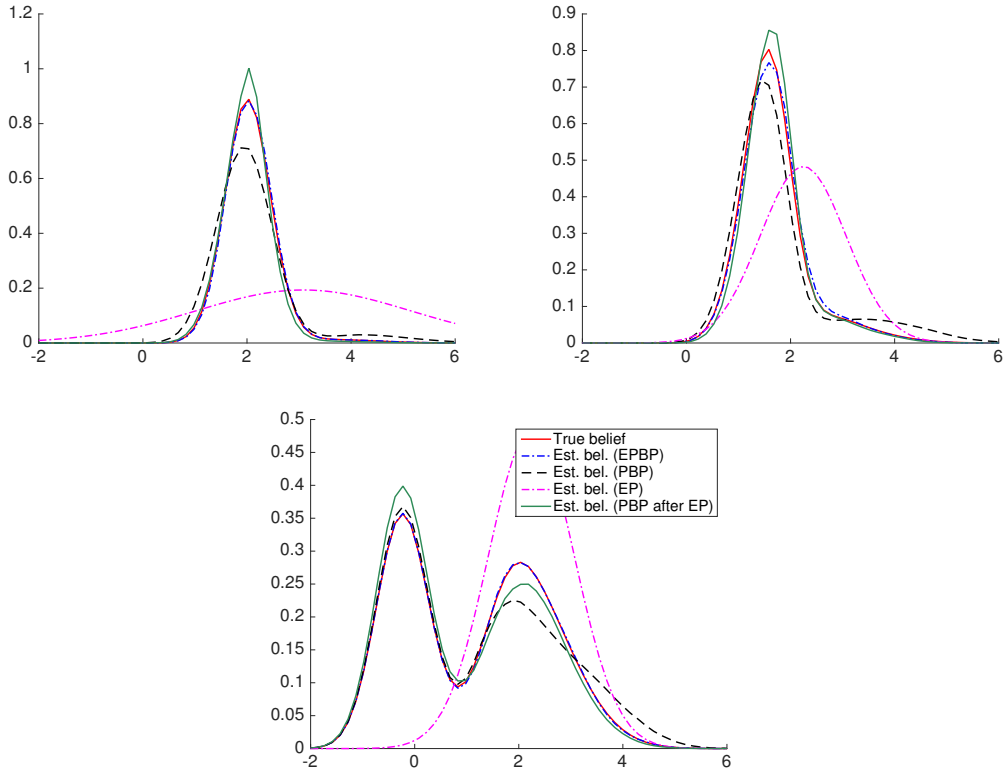


Figure 7.9: *Comparison of the beliefs on node* 1*,* 3 *and* 8 *(top left, top right, bottom) as obtained by evaluating LBP on a deterministic mesh, using EPBP, PBP, EP and PBP using the results of EP as proposals. All three plots share the same legend. This is for the tree example with* $N = 100$ *samples on each node and* 20 *LBP iterations. Again, one can observe that EPBP outperforms the other methods.*

7.3.2 Sub-quadratic implementation and denoising application

As outlined at point 7.2.4, the inherent quadratic complexity of the EPBP algorithm can be reduced to $\mathcal{O}(MN)$ where M is the number of mixture components effectively considered to represent the messages. We also consider the maximum likelihood-based projection mechanism as discussed in section 7.4 for the two experiments.

We apply this method to the 3×3 grid example in the case where $M = \mathcal{O}(\log(N))$: for $N = \{10, 20, 50, 100, 200, 500\}$, we pick $M = \{5, 6, 8, 10, 11, 13\}$. The results are illustrated in figure 7.10 where one can see that the $N \log N$ implementation compares very well to the original quadratic implementation at a much reduced cost.

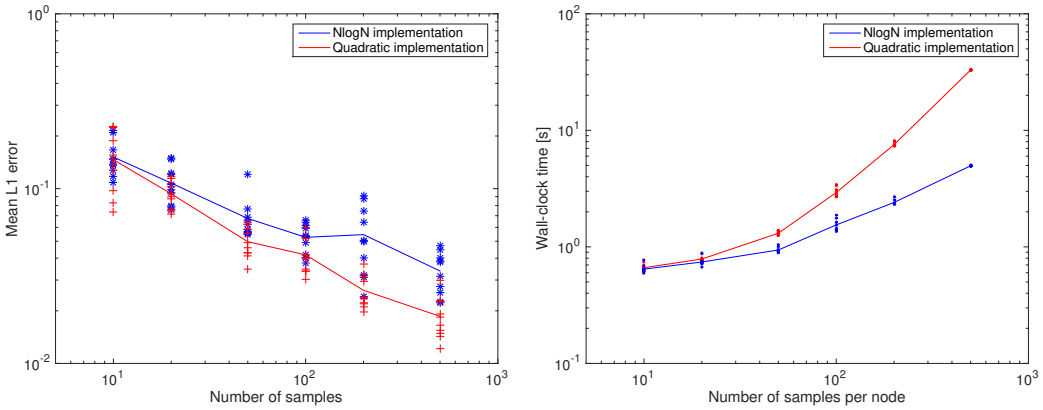


Figure 7.10: *Comparison of the quadratic and $\mathcal{O}(NlogN)$ implementations. (**left**) Comparison of the mean L1 error, (**right**) comparison of the wall-clock time. The sub-quadratic implementation performs almost as well as the original implementation and offers a significant speedup.*

We apply the same sub-quadratic method on a simple probabilistic model for an image denoising problem. The aim of this example is to show that the method can be applied to larger graphs and still provide good results. The model underlined is chosen to showcase the flexibility and applicability of our method in particular when the edge-potential is non-integrable (i.e. it would not meet the necessary condition (7.4) for NBP). It is not claimed to be an optimal approach to image denoising.[3]

The node and edge potentials are defined as follows:

$$\begin{cases} \psi_u(x_u) & = & \mathcal{N}(x_u - y_u; 0, 0.1) \\ \psi_{uv}(x_u, x_v) & = & \mathcal{L}^\lambda(x_u - x_v; 0, 0.03) \end{cases}, \tag{7.18}$$

where $\mathcal{L}^\lambda(x; \mu, \beta) = \mathcal{L}(x; \mu, \beta)$ if $|x| \leq \lambda$ and $\mathcal{L}(\lambda; \mu, \beta)$ otherwise. The node potential represents a simple Gaussian noise while the edge potential represents a clipped ℓ_1 similarity. In this example we set $\lambda = 0.2$. The value assigned to each pixel of the reconstruction is the estimated mean of the belief obtained over the corresponding node (figure 7.11). The image has size 50×50 (therefore corresponding to a grid graph

[3] In this case in particular, an optimisation-based method such as that of Rudin et al. (1992) is very likely to yield better results and much faster.

of 2500 nodes) and the simulation was run with $N = 30$ particles per nodes, $M = 5$ and 10 BP iterations taking under 2 minutes to complete with the setup described earlier. We compare it with the result obtained with EP on the same model. Running PBP or any other quadratic method on this example is prohibitively expensive due to the size of the underlying graph.

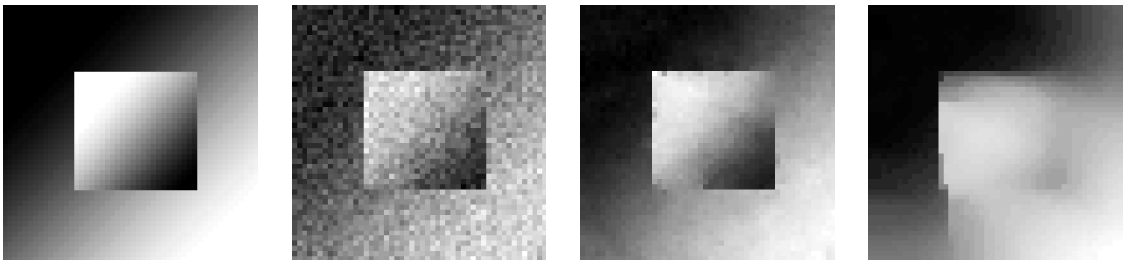


Figure 7.11: *From left to right: comparison of the original (first), noisy (second) and recovered image using the sub-quadratic implementation of EPBP (third) and with pure EP (fourth).*

7.4 Discussion

In this chapter, we presented an original way to design adaptively efficient and easy-to-sample-from proposals for a particle implementation of the LBP algorithm. The construction of proposals is done in the Expectation Propagation framework.

We have demonstrated empirically that the resulting algorithm is significantly faster and more accurate than an implementation of the PBP algorithm using the estimated beliefs as proposals and sampling from them using MCMC as proposed in Ihler and McAllester (2009). It is also more accurate than using plain EP due to the nonparametric nature of the messages and offers estimators of the LBP messages with almost sure convergence guarantee.

A sub-quadratic version of the method was also outlined and shown to perform almost as well as the original method on mildly multi-modal models, it was also applied successfully in a simple image denoising example illustrating that the method can be

applied on graphical models with thousands of nodes at a reasonable computational cost.

We believe that our method could be applied successfully to a wide range of applications requiring good approximation of the marginals of continuous MRFs such as smoothing for Hidden Markov Models Briers et al. (2010), tracking or computer vision Sudderth et al. (2004); Felzenszwalb and Huttenlocher (2004).

As discussed in point , other projection mechanisms than the KL projection can be considered. We could also look at other discrepancy measure than the KL such as α-divergences. This latter choice would suggest considering the power-EP framework (Minka, 2004) and could be justified if we seek to build representations of the messages with specific properties. It is unclear to us however how the choice of power in power-EP affects the quality of the proposals representing the node beliefs nor how it would affect the performances of the algorithm.

8 | Conclusion

In this chapter, we offer a critical look on the results obtained in this thesis and suggest further lines of work.

The purpose of this thesis was to leverage factorisation structures in probabilistic models in order to reduce the computational cost associated with traditional Bayesian inference methods. While a few interesting results were obtained, improving existing methods, it became clear during the analysis that some of the foundations of the methods themselves had issues. We discuss this superficially below before elaborating for each of the approaches we considered of the thesis.

A first simple criticism that can be leveraged towards performing inference on MRFs is that the inference step is, in some sense, a secondary one. The first and arguably most important one is to determine the model itself. In terms of the MRF this corresponds to determining the topology of the graph and determining the potentials on the cliques. This choice is potentially much more important than the inference algorithm used thereafter. Therefore, discussing accuracies obtained by different computational methods can seem inappropriate when the uncertainty around the model choice overwhelms the accuracy improvements obtained.

A second key criticism is that one of the justifications for attempting to perform

Bayesian inference is to model uncertainty. However it became clear during the work that led to this document that while the literature often mentions the need for uncertainty estimates (for example in a recent Nature paper by Ghahramani (2015)), it equally often omits to mention how exactly the quality of this uncertainty is assessed. Further, much of the literature in "Bayesian Machine Learning" discusses accuracy metrics such as the predictive RMSE or other metrics obtained with the posterior mean rather than, for example, the log-likelihood on a held out set. This is odd since optimisation methods relying on an appropriately regularised maximum-likelihood estimator will lead to similar (and likely better) metrics at a fraction of the computational cost using optimisation methods (see also Green et al. (2015)). Finally, using surrogate distributions as has been popularised with computationally-cheap methods such as *stochastic variational inference* (Hoffman et al., 2013) often leads to uncertainty estimates that are inadequate as discussed by Wang and Titterington (2005) which questions the reason for favouring such methods over regularised maximum likelihood estimation techniques.

8.1 On Sampling Methods

In the part on sampling methods, we started by considering the smoothing problem on a well defined non-linear Hidden Markov Model. Standard approaches such as the Forward Backward Smoothing Algorithm require a quadratic computational costs in the number of particles used to represent the marginals. We showed that these methods do not suffer significantly when colliding populations of samples are subsampled to reduce the total number of interactions that need to be considered. While this is good news in terms of computational complexity, we can not help but note that the overall performance of all particle smoothers we tested (quadratic or not) is rather poor at recovering a signal close to the ground-truth and, more importantly, does not

seem empirically to offer a significantly better approximation than simply looking at a Particle Filter. This point becomes even more obvious if the transition and observation densities are not known precisely. Of course, this observation is model-dependent and there may be specific models for which a particle smoother will significantly outperform the particle filter in the same way that the Kalman smoother can significantly outperform the Kalman filter.

We then looked at applying the Local Bouncy Particle Sampler on the Probabilistic Matrix Factorisation problem. While the application was interesting in its own right and thereby showed that the LBPS seems to be a viable option for inference on large graphical models, the reader surely noticed that the reported results – RMSE on the test set, the metric of choice of the corresponding literature – were blown out of the water by an adequate application of the Sparse SVD algorithm. This may reflect the data we considered (MovieLens 1M) since Salakhutdinov and Mnih (2008) show better performances of Bayesian PMF over SVD.[1] The LBPS is still in its early days of development and future work could look in more details at the selection of parameters such as the refreshment rate λ which seem to play an important role in the quality of the output trajectories. Exploring and interpreting results obtained on other large scale graphical models would also add to the understanding of this interesting method.

8.2 On Approximate Bayesian Inference

In the part on Approximate Bayesian Inference, we presented versions of the Expectation Propagation (EP) algorithm which are robust to Monte Carlo noise and can therefore be used as the backbone of a distributed Bayesian inference mechanism. We showed that a version of the EP algorithm inspired from the Mirror Descent algorithm performs much better than other versions considered. This is a positive result how-

[1] Though, as discussed earlier, it is important to note here that it is not specified *what* implementation of SVD they used nor whether they centred the data before applying the SVD.

ever it is unclear whether the uncertainty estimates obtained with the procedure are usable in complex models. Further, since (as per usual) it is the predictive RMSE that is reported, more work should be done to compare the results obtained from applying such a method as compared to a optimisation-based methods such as *downpour SGD* (Dean et al., 2012).[2]

Lastly, we looked at improving on the Particle Belief Propagation algorithm for arbitrary undirected graphical models using proposals on graph nodes that were adaptively constructed using EP. The method clearly outperformed PBP and is computationally not too expensive making it an attractive alternative. However, the method assumes that performing the Loopy Belief Propagation algorithm is an appropriate way of recovering sensible marginals on the graph nodes which is not guaranteed. The recovered beliefs may be good proxies for marginals but there are no guarantees as how they compare to the true marginals.

8.3 On scalable Bayesian Methods

Recently there has been much interest in applying Bayesian methods in the "Big Data" setting as well as in Machine Learning. We refer here to two insightful reviews on the topic: Green et al. (2015); Bardenet et al. (2017), both leading to a rather negative perspective on existing methods and approaches in the field.

Let us consider classical Machine Learning models based on a feature matrix of dimensions $n \times p$ where n is the number of instances and p the number of dimensions of the feature space (state space). In the case of low n (hundreds or less) and low p (half a dozen or less), Bayesian computations are known to be computationally practical and useful (Gelman et al., 2013). In the case of large n and $p \ll n$, also known as *tall data*, recent work has shown that Bayesian inference methods are not particularly useful

[2] Note that this was partially done in our paper (Hasenclever et al., 2017) but should be repeated in a simpler setting where all the moving parts are appropriately controlled.

and may in fact underperform significantly (Bardenet et al., 2017; Nagapetyan et al., 2017). However, the case of increasing interest has p large or very large. The question then becomes one of understanding the kind of conditional dependence structure that can be assumed on the features. Assuming a fully dependent model (dense graph) often leads to computationally intractable problems. Assuming a fully independent model (disconnected graph) or proxy like in Mean-Field Variational Inference or EP with a diagonal Gaussian exponential family, may be computationally cheap but may also be inappropriate and lead to grossly underestimated uncertainty estimates. Further, a better approach in the case where one is willing to consider a disconnected graph may simply be to consider a regularised maximum likelihood estimator.

We believe that the case of a sparse graphical model with a large number of nodes is the most promising one for Bayesian methods. Indeed, when a large number of nodes are present and the amount of data is limited in such a way that the posterior is not too concentrated, we believe that Bayesian inference can offer interesting perspectives. Few inference methods truly leverage such structures though the Local BPS seems to us to be a promising attempt and improvements are actively being researched upon.

Naesseth et al. (2015) have also proposed a SMC-like framework for graphical model which offers another interesting avenue for further exploration. We believe that this is an area of Bayesian computations where more work could lead to interesting and practical methods corresponding to modern applications.

A | Appendix

A.1 Fearnhead's Algorithm for a Linear-Gaussian Model

We describe here an algorithm corresponding to the description in (Fearnhead et al., 2010) with a simple linear and Gaussian underlying dynamic:

$$\begin{cases} \pi_0(x_1) &= \mathcal{N}(x_1; \mu_0, Q_0) \\ p(x_t \mid x_{t-1}) &= \mathcal{N}(x_t; A_t x_{t-1}, Q) \end{cases}$$

We start by discussing the form of the normalising densities following the choice suggested in (Briers et al., 2010) and considered in (Fearnhead et al., 2010) then discuss how the corresponding normalised backward information filter (BIF) can be targeted and finally how the particle filter and the normalised BIF can be coupled to yield estimators of the smoothing densities.

A.1.1 Normalising densities

The normalising density are defined with $\gamma_1 \equiv \pi_0$ and subsequently

$$\gamma_t(x_t) = \int p(x_t \,|\, x_{t-1})\gamma_{t-1}(x_{t-1})\,\mathrm{d}x_{t-1}, \quad \text{for } 2 \leq t \leq T \tag{A.1}$$

The integrand is a product of two Gaussian terms and so clearly γ_t will be a Gaussian itself. Let μ_{t-1} and Σ_{t-1} denote the mean and covariance matrix of γ_{t-1}. The quadratic term corresponding to the integrand is (ignoring the constant terms in x_t and x_{t-1}):

$$x_t^t Q^{-1} x_t - 2x_t^t A^t Q^{-1} x_{t-1} - 2x_{t-1}^t \Sigma_{t-1}^{-1}\mu_{t-1} + x_{t-1}^t A^t Q^{-1} A x_{t-1} + x_{t-1}^t \Sigma_{t-1}^{-1} x_{t-1} - 2\mu_{t-1}^t \Sigma_{t-1}^{-1} x_{t-1}. \tag{A.2}$$

In order for the integral (A.1) to simplify, we need to exhibit a Gaussian term in x_{t-1} so that the term integrates out easily. Assembling the terms in x_{t-1} appropriately we get

$$x_{t-1}^t (A^t Q^{-1} A + \Sigma_{t-1}^{-1}) x_{t-1} - 2x_{t-1}^t (A^t Q^{-1} x_t + \Sigma_{t-1}^{-1}\mu_{t-1}).$$

In order to form a complete Gaussian, we need to add a term corresponding to the mean. Let us call the mean and covariance matrix of this Gaussian $\mu_{t-1}^\bullet$ and $\Sigma_{t-1}^\bullet$ respectively. We have:

$$\begin{cases} (\Sigma_{t-1}^\bullet)^{-1} &= A^t Q^{-1} A + \Sigma_{t-1}^{-1} \\ (\Sigma_{t-1}^\bullet)^{-1}\mu_{t-1}^\bullet &= A^t Q^{-1} x_t + \Sigma_{t-1}^{-1}\mu_{t-1} \end{cases}.$$

The completion term is $\mu_{t-1}^\bullet (\Sigma_{t-1}^\bullet)^{-1}\mu_{t-1}^\bullet$ and must now be removed from the remaining terms in (A.2) to uncover the resulting Gaussian in x_t. We finally obtain the follow-

ing quadratic term in x_t

$$x_t^t \left(Q^{-1} - Q^{-1} A \Sigma_{t-1}^{\bullet} A^t Q^{-1}\right) x_t - 2 x_t \left(Q^{-1} A \Sigma_{t-1}^{\bullet} \Sigma_{t-1}^{-1} \mu_{t-1}\right).$$

This last expression gives the following recursion for the mean and the covariance matrices of the γ_t for $t = 2, \ldots, T$:

$$\begin{cases} \Sigma_t^{-1} &= Q^{-1} - Q^{-1} A \Sigma_{t-1}^{\bullet} A^t Q^{-1} \\ \mu_t &= \Sigma_t \left(Q^{-1} A \Sigma_{t-1}^{\bullet} \Sigma_{t-1}^{-1} \mu_{t-1}\right) \end{cases} \tag{A.3}$$

A.1.2 Targeting the normalised BIF

Now that we have defined the normalising densities, we can sample backwards to target the normalised BIF. Let us denote by $\tilde{X}_t^{(j)}$ and $\tilde{w}_t^{(j)}$ the particles and weights associated with the normalised BIF. The initial set of particles for step T can be sampled from the optimal proposal $\gamma_T(x_t) p(y_T \,|\, x_T)$ which is a Gaussian in x_T; using a similar approach than in the previous point, it is easy to see that the mean $\tilde{\mu}_T$ and covariance matrix $\tilde{\Sigma}_T$ are given by

$$\begin{cases} \tilde{\Sigma}_T^{-1} &= \Sigma_T^{-1} + B^t R^{-1} B \\ \tilde{\mu}_T &= \Sigma_T \left(\Sigma_T^{-1} \mu_T + B^t R^{-1} y_T\right) \end{cases}$$

The weights are uniform since the sampling is done from the optimal proposal. Subsequently, the optimal proposal for the following steps is derived from (3.9):

$$q(x_t \,|\, \tilde{X}_{t+1}^{(j)}) \;=\; \frac{\gamma_t(x_t) p(x_{t+1} \,|\, x_t) p(y_t \,|\, x_t)}{\gamma_{t+1}(\tilde{X}_{t+1}^{(j)})},$$

and the numerator is a Gaussian in x_t and, again, it is easy to obtain

$$\begin{cases} \tilde{\Sigma}_t^{-1} &= \Sigma_t^{-1} + B^t R^{-1} B + A^t Q^{-1} A \\ \tilde{\mu}_t &= \Sigma_t \left(\Sigma_t^{-1} \mu_t + B^t R^{-1} y_t + A^t Q^{-1} x_{t+1} \right) \end{cases}$$

This time, the weights need to be adapted to reflect the term $\gamma_{t+1}(\tilde{X}_{t+1}^{(j)})$ which is easily computed as it is the likelihood of a Gaussian with known mean and variance. Therefore, we have for $t = T-1, \ldots, 1$:

$$\begin{cases} \tilde{X}_t^{(j)} &\sim \mathcal{N}(x_t; \tilde{\mu}_t, \tilde{\Sigma}_t) \\ \tilde{w}_t^{(j)} &\propto \tilde{w}_{t+1}^{(j)} / \gamma(\tilde{X}_{t+1}^{(j)}) \end{cases} \quad \text{for } j = 1, \ldots, N.$$

A.1.3 Targeting the smoothing distributions

The last step of the algorithm is the combination of the representation of the predictive density relying on a particle representation of the filtering densities $\{X_t^{(i)}, w_t^{(i)}\}_{i,t=1}^{N,T}$ with that of the normalised BIF. For this, as suggested in Fearnhead et al. (2010), we follow the procedure below:

$$\begin{cases} i^\star &\sim \mathcal{M}(\{w_t^{(i)}\}_{i=1}^N) \\ j^\star &\sim \mathcal{M}(\{\tilde{w}_t^{(j)}\}_{i=1}^N) \\ \overline{X}_t^{(j)} &\sim \varphi_{i^\star, j^\star}(x_t) \end{cases} \quad \text{for } j = 1, \ldots, N, \text{ and } t = 1, \ldots, T$$

where $\varphi_{i^\star, j^\star}(x_t) = p(\tilde{X}_{t+1}^{(j^\star)} \mid x_t) p(y_t \mid x_t) p(x_t \mid X_{t-1}^{(i^\star)})$ and no reweighing step is needed since we can sample from it exactly (it is a Gaussian) and it is the optimal proposal. Using the same approach as in the previous two points, if we denote the parameters

of the optimal proposal $\varphi_{i^\star,j^\star}$ as $\overline{\mu}_t$ and $\overline{\Sigma}_t$ then:

$$\begin{cases} \overline{\Sigma}_t^{-1} &= Q^{-1} + A^t Q^{-1} A + B^t R^{-1} B \\ \overline{\mu}_t &= \overline{\Sigma}_t \left(A^t Q^{-1} \tilde{X}_{t+1}^{(j^\star)} + Q^{-1} A X_{t-1}^{(i^\star)} + B^t R^{-1} y_t \right) \end{cases}$$

A.2 Mirror Descent and Natural Gradient

In this part we show briefly the reasoning behind the mirror-descent algorithm focusing on the case of the KL geometry that is considered in the core document.

A.2.1 Classical gradient descent

We start from the generic problem of unconstrained minimisation of a convex function f. Additionally, we assume that the function has a computable gradient ∇f everywhere. The gradient descent algorithm considers the following iterative scheme:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

which, under some conditions on the sequence of step-sizes $(\alpha_k)_0^\infty$ will converge to a minimiser of the function i.e., an $x^\star$ such that $f(x^\star) \leq f(x)$ for all x. This scheme is equivalent to solving a sequence of optimisation problems:

$$x_{k+1} = \arg\min_x \left\{ \langle x, \nabla f(x_k) \rangle + \frac{1}{\alpha_k} \frac{\|x - x_k\|_2^2}{2} \right\}. \tag{A.4}$$

This re-formulation shows that the gradient-descent is explicitly linked to an isotropic Euclidean geometry via the distance $d(x,y) = \|x-y\|_2^2/2$. Other geometries can be considered by considering other distances or divergences. In particular, when a Bregman divergence is considered we get the mirror descent algorithm as shown below.

A.2.2 Bregman divergences

Consider a smooth, strictly convex function φ. By definition, this function is such that for any couple of points (x, z) in its domain with $x \neq z$,

$$\varphi(z) \quad > \quad \varphi(x) + \langle z - x, \nabla\varphi(x) \rangle \, .$$

We can therefore build a divergence by rearranging the previous inequality:

$$B_\varphi(z, x) \quad := \quad \varphi(z) - \varphi(x) - \langle z - x, \nabla\varphi(x) \rangle \, . \tag{A.5}$$

Note that $B_\varphi(z, x) > 0$ for all admissible $z \neq x$ and $B_\varphi(x, x) = 0$ for all admissible x. Apart from the fact that such a *Bregman divergence* is not necessarily symmetric, it has the same properties as a distance. A particular case is that of $\varphi(x) = \|x\|_2^2 / 2$ in which case the associated Bregman divergence is nothing but the Euclidean distance.

A.2.3 Mirror descent algorithm

Let us now consider the generalised gradient descent scheme with the Bregman divergence associated with some strongly convex, smooth function φ:

$$x_{k+1} \quad = \quad \arg\min_x \quad \left\{ \langle x, \nabla f(x_k) \rangle + \frac{1}{\alpha_k} B_\varphi(x, x_k) \right\} . \tag{A.6}$$

Multiplying the objective function by α_k, rearranging and taking the gradient in x to get the first order condition leads to:

$$\alpha_k \nabla f(x_k) + \nabla\varphi(x_{k+1}) - \nabla\varphi(x_k) \quad = \quad 0.$$

Rearranging and using $(\nabla\varphi)^{-1} \equiv \nabla\varphi^\star$ leads to:

$$x_{k+1} \quad = \quad \nabla\varphi^\star \left[\nabla\varphi(x_k) - \alpha_k \nabla f(x_k)\right]. \tag{A.7}$$

This algorithm is known as the *mirror descent algorithm* (Beck and Teboulle, 2003).

A.2.4 KL geometry and natural gradient descent

When considering an objective function $\mathcal{J}$ depending on a distribution $q_\theta \in \mathcal{F}_\phi$, one can consider the geometry of the natural parameters i.e., iterate from θ_k to θ_{k+1} using the classical gradient descent scheme. However, θ_k and θ_{k+1} will index two distributions q_{θ_k} and $q_{\theta_{k+1}}$ and it therefore makes more sense to consider a divergence between those distributions rather than between their natural parameters. We consider the KL divergence with for $q, q' \in \mathcal{F}_\phi$:

$$\mathrm{KL}(q', q) \quad = \quad \mathbb{E}_q[\log q - \log q'].$$

Since $q_\theta \equiv \exp(\langle\theta, \phi\rangle - A(\theta))$, this reduces to

$$\mathrm{KL}(q', q) \quad = \quad A(\theta') - A(\theta) - \langle\theta' - \theta, A(\theta)\rangle$$

which is the Bregman divergence associated with the log-partition function A. The corresponding mirror-descent algorithm reads:

$$\theta_{k+1} \quad = \quad \nabla A^\star \left[\nabla A(\theta) - \alpha_k \nabla \mathcal{J}(\theta_k)\right], \tag{A.8}$$

which is often known as the *natural gradient descent* algorithm in the machine learning community. For a deeper analysis of the reasoning behind using the natural gradient descent in learning, we refer to the seminal paper by Amari (1998).

References

S.-I. Amari. *Natural Gradient Works Efficiently in Learning*. Neur. Comp., 10:251–276, 1998.

B. Anderson and J. Moore. *Optimal Filtering*. Prentice Hall, 1979.

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*. IEEE Trans. Sig. Proc., 50(2):174–188, 2002.

R. Bardenet, A. Doucet, and C. Holmes. *On Markov Chain Monte Carlo Methods for Tall Data*. JMLR, 2017.

A. Barp, F.-X. Briol, A. D. Kennedy, and M. Girolami. *Geometry and Dynamics for Markov Chain Monte Carlo*. Ann. Rev. Stat. App., 5(1), 2018.

S. Barthelme and N. Chopin. *ABC-EP: Expectation Propagation for Likelihood-free Bayesian Computation*. ICML, 2011.

H. Battey, J. Fan, H. Liu, J. Lu, and Z. Zhu. *Distributed Estimation and Inference with Statistical Guarantees*. arXiv:1509.05457, 2015.

A. Beck and M. Teboulle. *Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization*. Operations Research Letters, 31:167–175, 2003.

R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

T. Bengtsson, P. Bickel, and B. Li. *Curse-of-dimensionality Revisited: Collapse of the Particle Filter in Very Large Scale Systems*. Inst. Math. Stats. Coll., pages 316–334, 2008.

E. Bernton, P. E. Jacob, M. Gerber, and C. P. Robert. *Inference in Generative Models using the Wasserstein Distance*. arXiv:1701.05146, 2017.

M. Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. arXiv:1701.02434, 2017.

J. Bierkens, P. Fearnhead, and G. Roberts. *The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data*. arXiv:1607.03188, 2016.

J. Bierkens, A. Bouchard-Côté, A. Doucet, A. B. Duncan, P. Fearnhead, T. Lienart, G. Roberts, and S. J. Vollmer. *Piecewise Deterministic Markov Processes for Scalable Monte Carlo on Restricted Domains*. arXiv:1701.04244, 2017.

A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. The MIT Press, 2011.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. *Variational Inference: A Review for Statisticians*. arXiv:1601.00670v4, 2016.

A. Bouchard-Côté, S. J. Vollmer, and A. Doucet. *The Bouncy Particle Sampler: A Non-Reversible Rejection-Free Markov Chain Monte Carlo Method*. arXiv:1510.02451, 2015.

Y. Bresler. *Two Filter Formulae for Discrete Time Nonlinear Bayesian Smoothing*. IJC, 43:629–641, 1986.

M. Briers, A. Doucet, and S. S. Singh. *Sequential Auxiliary Particle Belief Propagation*. ICIF, 1: 705–711, 2005.

M. Briers, A. Doucet, and S. Maskell. *Smoothing Algorithms for State-Space Models*. AISM, 62 (1):61–89, 2010.

L. D. Brown. *Fundamentals of Statistical Exponential Families*. Inst. Math. Stats., 1986.

R. Caflisch. *Monte Carlo and Quasi-Monte Carlo Methods*. Acta Numerica, pages 1–49, 1998.

N. Chopin. *Central Limit Theorem for Sequential Monte Carlo Methods and its Application to Bayesian Inference*. Ann. Stats., 32(6):2385–2411, 2004.

D. Crisan and A. Doucet. *A Survey of Convergence Results on Particle Filtering Methods for Practitioners*. IEEE Trans. Sig. Proc., 50(3):736–746, 2002.

P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. New York, Academic Press, 1975.

J. Dean, C. G. S., R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. *Large Scale Distributed Deep Networks*. NIPS, 2012.

G. Dehaene and S. Barthelme. *Expectation Propagation in the Large-data Limit*. arXiv:1503.08060, 2015.

P. Del Moral, A. Doucet, and A. Jasra. *Sequential Monte Carlo Samplers*. JRSSB, (68):411–436, 2006.

A. Doucet and A. Johansen. *A Tutorial on Particle Filtering and Smoothing: Fifteen years later*. Tutorial, 2011.

A. Doucet, S. Godsill, and C. Andrieu. *On Sequential Monte Carlo Sampling Methods for Bayesian Filtering*. Stats. and Comp., 2000.

S. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. *Just-In-Time Learning for Fast and Flexible Inference*. NIPS, 2014.

P. Fearnhead, D. Wyncoll, and J. Tawn. *A Sequential Smoothing Algorithm with Linear Computational Cost*. Biometrika, 97(2):447–464, June 2010.

P. F. Felzenszwalb and D. P. Huttenlocher. *Efficient Graph-Based Image Segmentation*. IJCV, 59 (2), 2004.

M. Gales and S. Young. *The Application of Hidden Markov Models in Speech Recognition*. Found. and Tr. in Sig. Proc., 1(3):195–304, 2007.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis, 3d ed.* Chapman & Hall, 2013.

A. Gelman, A. Vehtari, P. Jylänki, C. Robert, N. Chopin, and J. P. Cunningham. *Expectation Propagation as a Way of Life*. arXiv:1412.4869, 2014.

C. Geyer. *Markov Chain Monte Carlo Lecture Notes*. Lecture Notes, 2005.

Z. Ghahramani. *An Introduction to Hidden Markov Models and Bayesian Networks*. IJPRAI, 15 (1):9–42, 2001.

Z. Ghahramani. *Probabilistic Machine Learning and Artificial Intelligence*. Nature, 2015.

S. Godsill, A. Doucet, and M. West. *Monte Carlo Smoothing for Nonlinear Time Series*. JASA, 2004.

P. J. Green, K. Latuszynski, M. Pereyra, and C. P. Robert. *Bayesian Computation: a Summary of the Current State, and Samples Backwards and Forwards*. Stat. Comput., 2015.

L. Hasenclever, S. Webb, T. Lienart, S. Vollmer, B. Lakshminarayanan, C. Blundell, and Y. W. Teh. *Distributed Bayesian Learning with Stochastic Natural Gradient Expectation Propagation and the Posterior Server*. JMLR, (18):1–37, 2017.

N. Heess, D. Tarlow, and J. Winn. *Learning to Pass Expectation Propagation Messages*. NIPS, 2013.

R. Herbrich. *On Gaussian Expectation Propagation*. MSR TR, 2005.

R. Herbrich, T. Minka, and T. Graepel. *TrueSkill: A Bayesian Skill Rating System*. NIPS, 2006.

D. Hernandez-Lobato, J. M. Hernandez-Lobato, and P. Dupont. *Generalized Spike-and-Slab Priors for Bayesian Group Feature Selection Using Expectation Propagation*. JMLR, 14, 2013.

J. M. Hernandez-Lobato, Y. Li, M. Rowland, D. Hernandez-Lobato, T. Bui, and R. E. Turner. *Black-box α-divergence Minimization*. arXiv:1511.03243, 2015.

T. Heskes. *On the Uniqueness of Loopy Belief Propagation Fixed Points*. Neural Comp., 16:2379–2413, 2004.

T. Heskes and O. Zoeter. *Extended Version of "Expectation Propagation for Approximate Inference in Dynamic Bayesian Networks"*. UAI, 2003.

T. Heskes, M. Opper, W. Wiegerink, and O. Winther. *Approximate Inference Techniques with Expectation Constraints*. JSM, (11), 2005.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. *Stochastic Variational Inference*. JMLR, 14: 1303–1347, 2013.

J. D. Hol, T. B. Schön, and F. Gustafsson. *On Resampling Algorithms for Particle Filters*. IEEE Proc. NSSP, pages 79–82, 2006.

M. Hürzeler and H. R. Künsch. *Monte Carlo Approximations for General State-Space Models*. JCGS, 7(2):175–193, 1998.

A. T. Ihler and D. A. McAllester. *Particle Belief Propagation*. AISTATS, pages 256–263, 2009.

A. T. Ihler, J. W. Fisher, and A. S. Willsky. *Loopy Belief Propagation: Convergence and Effects of Message Errors*. JMLR, (6):905–936, 2005.

W. Jitkrittum, A. Gretton, N. Heess, S. A. Eslami, B. Lakshminarayanan, D. Sejdinovic, and Z. Szabo. *Kernel-Based Just-In-Time Learning for Passing Expectation Propagation Messages*. NIPS, 2015.

G. Kitagawa. *Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models*. JCGS, 1996.

A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. *A Scalable Bootstrap for Massive Data*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(4):795–816, 2014.

A. Kong. *A Note on Importance Sampling using Standardized Weights*. U. of Chicago TR, 348, 1992.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. *Automatic Differentiation Variational Inference*. arXiv:1603.00788, 2016.

S. Kullback and R. A. Leibler. *On Information and Sufficiency*. Ann. Math. Stat., 22(1):79–86, 1951.

M. Kuss and C. E. Rasmussen. *Assessing Approximate Inference for Binary Gaussian Process Classification*. JMLR, 6:1679–1704, 2005.

Y. Li and R. E. Turner. *Renyi Divergence Variational Inference*. arXiv:1602.02311, 2016.

Y. Li, J. M. Hernandez-Lobato, and R. E. Turner. *Stochastic Expectation Propagation*. NIPS, 2015.

T. Lienart, Y. W. Teh, and A. Doucet. *Expectation Particle Belief Propagation*. NIPS, 2015.

J. Miguez and K. Eugenia. *Particle filtering with transformed weights*. IEEE CAMSAP, pages 364–367, 2013.

T. P. Minka. *Expectation Propagation for Approximate Bayesian Inference*. UAI, pages 362–369, 2001a.

T. P. Minka. *A Family of Algorithms for Approximate Bayesian inference*. MIT PhD thesis, 2001b.

T. P. Minka. *The EP Energy Function and Minimization Schemes*. MSR TR, 2001c.

T. P. Minka. *Power EP*. MSR TR, 2004.

C. Naesseth, F. Lindsten, and T. Schön. *Sequential Monte Carlo for Graphical Models*. NIPS, 2015.

T. Nagapetyan, A. B. Duncan, L. Hasenclever, S. J. Vollmer, L. Szpruch, and K. Zygalakis. *The True Cost of Stochastic Gradient Langevin Dynamics*. arXiv:1706.02692, 2017.

R. M. Neal. *MCMC using Hamiltonian Dynamics*. Chapman & Hall / CRC Press, 2011.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

M. Nikolova. *Thresholding Implied by Truncated Quadratic Regularization*. IEEE Trans. Sig. Proc., 48(12):3437–3450, 2000.

M. Opper. *A Bayesian Approach to Online Learning*. Aston U. TR, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, 1988.

E. A. J. F. Peters and G. de With. *Rejection-free Monte Carlo Sampling for General Potentials*. PRE, 85:026703, 2012.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

R. T. Rockafellar. *Convex Analysis*. Princeton Mathematical Series, 1970.

L. I. Rudin, S. Osher, and E. Fatemi. *Nonlinear Total Variation Based Noise Removal Algorithms*. Physica D, 60(1):259–268, 1992.

R. Salakhutdinov and A. Mnih. *Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo*. ICML, 2008.

M. Seeger. *Expectation Propagation for Exponential Families*. UC Berkeley TR, 2007.

D. C. Sorensen. *Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations*. NASA TR, 1996.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. *Maximum-Margin Matrix Factorization*. NIPS, 2004.

E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. *Nonparametric Belief Propagation*. CVPR, 1:605–612, 2003.

E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. *Visual Hand tracking Using Nonparametric Belief Propagation*. In *CVPR*, 2004.

E. Taghavi. *A Study of Linear Complexity Particle Filter Smoothers*. Chalmers U. thesis, 2012.

P. Vanetti, A. Bouchard-Côté, G. Deligiannidis, and A. Doucet. *Piecewise Deterministic Markov Chain Monte Carlo*. arXiv:1707.05296, 2017.

M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Found. and Tr. in Mach. Learn., 1(1–2):1–305, 2008.

B. Wang and D. M. Titterington. *Inadequacy of Interval Estimates Corresponding to Variational Bayesian Approximations*. AISTATS, 2005.

Y. Weiss and W. T. Freeman. *On the Optimality of Solutions of the Max-Product Belief Propagation Algorithm*. IEEE Trans. Inf. Th., 2000.

C. Wu and C. Robert. *Generalized Bouncy Particle Sampler*. arXiv:1706.04781, 2017.

M. Xu, B. Lakshminarayanan, Y. W. Teh, J. Zhu, and B. Zhang. *Distributed Bayesian Posterior Sampling via Moment Sharing*. NIPS, pages 3356–3364, 2014.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Bethe Free Energy, Kikuchi Approximations and Belief Propagation Algorithms*. MERL TR, 2001.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms*. MERL TR, 2002.

B. M. Yu, K. V. Shenoy, and M. Sahani. *Expectation Propagation for Inference in Non-Linear Dynamical Models with Poisson Observations*. NSSP, 2006.

T. Zhao, G. Cheng, and H. Liu. *A Partially Linear Framework for Massive Heterogeneous Data*. Ann. Statist., 44(4):1400–1437, 2016.

W. Zucchini, I. L. MacDonald, and R. Langrock. *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman and Hall, 2016.