Introspective Classification for Robot Perception and Decision Making

Hugo Grimmett

New College



Supervisor:

Ingmar Posner

Mobile Robotics Group

Department of Engineering Science

University of Oxford

August 2016

Hugo Grimmett
New College

Doctor of Philosophy
August 2016

Introspective Classification for Robot Perception and Decision Making

Abstract

In robotics, a classifier is often a core component of the decision-making framework. *Precision* and *recall* have been widely adopted as canonical metrics to quantify the performance of a classifier, but for applications involving mission-critical decision making, good performance in relation to these metrics is insufficient. The use of a classification framework which produces scores with inappropriate confidences will ultimately lead to the robot making bad decisions, thereby compromising robot or user safety. In order to select a classifier which will make decisions reflecting the nature of the costs, we should pay careful attention to the ways in which it generates scores. We introduce and motivate the importance of a classifier's *introspective* capacity: the ability to give an appropriate assessment of confidence with any test case. Classification made confidently must be correct, and mistakes should be made with high uncertainty. A classifier's capacity to do so must remain consistent despite unusual or surprising test cases. We propose that a key ingredient for introspection is a classifier's potential to increase its uncertainty with the distance between a test datum and its training data.

We define the ideal introspective behaviour, and derive idealised classifiers which serve to benchmark a number of commonly used classification frameworks in a variety of decision-making tasks. We show that classifiers that offer predictive variance at test-time are more cautious and less over-confident than those which consider a single hypothesis or discriminant. However, in high-cost (or high-risk) decision making, none of the classifiers evaluated in this thesis are sufficiently introspective to prevent all potential catastrophic mistakes. We show that in sequential decision-making, when the mapping from score to class is explicitly stated, a classifier's ability to behave consistently despite non-stationary test data is of primary importance.

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Hugo Grimmett, New College

Funding

The work described in this thesis was funded under the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916 (V-CHARGE).

Acknowledgements

First and foremost I would like to thank my supervisor, Professor Ingmar Posner. His support, direction, and collaboration know no bounds, and I am immensely grateful for having had the opportunity to work together. I would also like to thank Professor Paul Newman, who inspired me to begin this journey and made me feel at home at MRG.

Secondly, I would like to thank my thesis examiners Professors Mike Osbourne and Tom Duckett for their insightful questions and feedback during and after the viva examination. Their thorough treatment of my work has brought to light interesting aspects that I had not previously considered, and for that I have great appreciation.

I give great thanks to my friends and coauthors, Drs Rohan Paul, Rudi Triebel, Lina Paz, and Pedro Pinies. The late nights writing papers together are my fondest and most formative memories from the past years.

I owe a great deal to countless enlightening discussions with Dr Chi Tong, Geoff Hester, and Dr Dominic Wang. Their generosity of their immense collective knowledge empowers all of those around them.

The funding from the V-Charge project allowed me to pursue this work, and collaborate and find friends in Paul Furgale, Wojciech Derendarz, Ulrich Schwesinger, and Matthias Buerki to name a few. It has been a pleasure to work with you all, and to contribute to the success that is the V-Charge autonomous valet car.

My thanks to all of MRG, who have supported and inspired me throughout the years. To name but a few: Colin McManus, Terry Scott, Dan Withers, Peter Ondruska, Corina Gurau, Matt Gadd, Winston Churchill, Julie Dequaire, Chris Prahacs, Tom Wilcox, and Anita Hancox. Also thank you to those who have since left the group for your immeasurable wisdom, Alastair Harrison, Ashley Napier, Bonolo Mathibela, Ben Davis, and Ian Baldwin. My heartfelt gratitude also goes to my friend Amaury Dame, whose perspectives on life guide me always.

I am indebted to those outside robotics who supported me throughout my time in Oxford, most notably but not limited to Philippa Harris, Leila Denniston, Michael West, Alex Nevitte, Jonny Sadler, Christine Moore, Jon Daly, Joanna Hamer, Isaac Black, and Rachel James. Thank you for believing in me.

Finally I would like to thank my parents, Geoffrey and Rosine, for their overwhelming and unconditional love and support. Their selflessness and trust are both my inspiration and aspiration.

Hugo Grimmett

August 11, 2016

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Introspection 2

1.3 Contributions 3

1.4 Thesis Structure 4

1.5 Publications 5

2 Semantic Mapping: A Case Study **7**

2.1 The V-Charge Vehicle 9

2.2 Related Works 10

2.3 Semantic Mapping 11

2.3.1 The Road Network 12

2.3.2 Parking Space Locations 17

2.3.3 Recommended Driving Speed 22

2.4 Integrating Metric and Semantic Layers 23

2.5 Conclusions 29

3 Data Sets, Features, and Performance Metrics **31**

3.1 Traffic Lights Recognition 32

3.2 GTSRB 34

3.3 Daimler Pedestrian 34

3.4 KITTI . . . 35
3.5 Synthetic Data . . . 35
3.6 Features . . . 37
3.6.1 Template Features . . . 37
3.6.2 Histogram of Oriented Gradients (HOG) . . . 40
3.7 Performance Metrics . . . 40
3.7.1 Precision, Recall, Accuracy, and F-measure . . . 40

4 Introspection **43**
4.1 The Ideal Classifier . . . 44
4.2 Related Works . . . 48
4.3 Idealised Classifiers . . . 51
4.3.1 Determining the Density Functions . . . 55
4.4 Summary . . . 60

5 Introspection in Practice **61**
5.1 Notation . . . 63
5.2 Measures of Uncertainty . . . 63
5.3 A Distance-Based View on Introspection . . . 65
5.4 Related Works . . . 67
5.5 Commonly-Used Classification Frameworks . . . 69
5.5.1 Gaussian Processes Classification . . . 70
5.5.2 Support Vector Machine . . . 74
5.5.3 LogitBoost . . . 76
5.5.4 Random Forests . . . 77
5.5.5 Kernels . . . 79
5.6 Analysis of Non-Stationary Data . . . 81
5.6.1 Synthetic Data . . . 83

5.6.2 Real Data . . . 88
5.6.3 Discussion . . . 94
5.7 Uncertainty in Detection . . . 95
5.8 Conclusions . . . 106

6 Introspection in Decision Making 109
6.1 Making Errors with Uncertainty . . . 110
6.2 Active Learning . . . 113
6.2.1 Related Works . . . 115
6.2.2 The Cross-Over Experiment . . . 116
6.2.3 Discussion . . . 117
6.3 Decision Making with Costs . . . 121
6.3.1 Bayesian Decision Theory . . . 122
6.3.2 Relating Costs to ϵ-bounds . . . 125
6.3.3 Experiments . . . 126
6.4 Conclusions . . . 129

7 Introspection in Sequential Decision Making 133
7.1 The Observation Probability Function as a Classifier . . . 135
7.2 Related Works . . . 137
7.3 Test Scenarios . . . 139
7.3.1 Grid World . . . 139
7.3.2 Wumpus World . . . 142
7.4 Entropy . . . 144
7.5 Experiments . . . 147
7.5.1 The Ill Effects of Non-stationarity . . . 149
7.5.2 Consistent and Appropriate Sensors . . . 155
7.5.3 Changing the Size of the World . . . 161

7.5.4 Changing the Number of Sensors . . . 163
7.6 Conclusions . . . 165

8 Conclusions and Future Work **167**
8.1 Conclusions . . . 167
8.2 Future Work . . . 170
8.2.1 Introspection . . . 170
8.2.2 Semantic Mapping . . . 175
8.2.3 Active Learning . . . 175
8.2.4 Classical Decision Making . . . 176
8.2.5 Sequential Decision Making . . . 177

Appendices **180**

A Road Graph Generation Algorithms **181**

B Classifier Probability Contours **184**

C Idealised Classifier Error Functions **187**

D Empirical Probability Density Functions of Real Classifiers **189**

E MDPs and POMDPs **192**
E.1 Markov Decision Process (MDP) . . . 192
E.2 Partially Observable Markov Decision Processes (POMDP) . . . 197

Bibliography **202**

Chapter 1

Introduction

1.1 Motivation

An autonomous robot operating in an environment such as a city street will see so many varied inputs that it is impossible to have prepared it by labelling each one. In practice we attempt to forewarn it by selecting training exemplars and generating a model, hoping that together they will be sufficient to generalise well to its future experiences. However, these exemplars will almost never fully characterise the problem. This is either because they are too few to cover the domain or because the domain is non-stationary and shifts unexpectedly over time. Nevertheless, this is how we typically perform classification, and the classification scores are used for making potentially safety-critical decisions. We hope that when our robot encounters a test which is not represented in its training set, the model it learned will allow it to generalise and correctly classify the object. However, we will inevitably encounter a test datum which is perplexing – one that is not similar to the things we have seen before. This could be due to lighting, perspective change, inter-class variation, and many other factors. What should we expect our classifier to do with these, and what will be its resulting decision?

For example, if a robot is tasked with safely crossing the road, we need it to wait cautiously when it sees something unexpected, rather than confidently and erroneously determine that the way is clear.

1.2 Introspection

In this thesis we evaluate the abilities of a number of commonly-used classification frameworks to make appropriate decisions in robotics. Specifically, we are concerned with the situation where a classifier makes a decision about a test datum which is unlike anything it has seen during training. We argue that this situation is commonplace, and that we can associate the appropriateness of the decision with that classifier's treatment of *distance* in feature space. Data which are far away from the training set should be regarded with suspicion, or high uncertainty. Familiar examples, on the other hand, those more similar to the training examples, should be perceived with greater confidence, because indeed their classification is likely to be correct.

We propose that a classifier's incorrect decisions should be made with greater suspicion or uncertainty, while the ones made with confidence should also tend to be correct. This tendency we term, '*introspection*'. A classifier with a great introspective capacity will correlate[1] confidence to correctness. This behaviour is not captured by precision, recall, or accuracy, and yet can critically affect any resulting decisions.

We will define the ideal introspective behaviour, and compare a number of popular classification frameworks to this ideal. We will examine the methods by which they determine classification confidence, and relate them to distances between data

[1] in using the word correlation in this context, we mean to say that as confidence increases, so correctness tends to increase, and vice-versa for decreases. This is more general than the strictly mathematical meaning of correlation. Some might prefer the term 'positive association'.

or models. Having evaluated their introspective capacity, we investigate the effects within various domains of robotics:

1. object classification and detection,
2. active learning,
3. high-cost decision making, and finally
4. sequential decision making.

1.3 Contributions

The following contributions are made in this thesis:

- (Chapter 2) We motivate the importance of introspection via a case-study of a semantic mapping system. Specifically, we build semantic maps of car parks for an autonomous car to perform valet parking.
- (Chapter 4) The concept of introspection: that it is useful for a classifier to make its mistakes only with high uncertainty, while simultaneously making correct classifications with confidence, in order to allow the perception system to predict when it might be making a mistake. That prediction can then be used to take an appropriate action.
- (Chapter 4) The ideal behaviour for an introspective classifier, and a number of idealised example classifiers displaying this behaviour to varying degrees.
- (Chapter 5) The proposition that this introspective tendency is linked to distances, and that a consistent treatment of distance leads to better introspection in classification tasks. We determine the introspective capacities of a number of real classifiers.

- (Chapter 6) We evaluate the importance of introspection in active learning, where truly introspective classifiers should have a great advantage. We show that the more introspective classifier does indeed benefit over the other; the differences are small but significant to the 99% level.

- (Chapter 6) An investigation of the introspective capacities of the real classifiers in classical decision-making, particularly in the presence of expensive outcomes. We show that none of our real classifiers are introspective enough to avoid high-confidence errors across every data set.

- (Chapter 7) We show that introspection is important in sequential decision making, and that the ability to differentiate between true and false classifications leads to achieving the goal faster. In this particular setting we place value on the information content of the classifiers rather than their correlation of confidence with correctness. We analyse the effects of poorly modelling classifier behaviour, and the detriment to the quality of the resulting decisions.

1.4 Thesis Structure

We motivate the need for introspection in Chapter 2 with our work on semantic mapping in the context of an autonomous valet service.

In Chapter 3 we discuss the three data sets used throughout this thesis. This chapter can be skipped on first reading, and used as a reference in the later chapters.

In Chapter 4 we define and further motivate the ideal introspective behaviour, and design a number of idealised classifiers which display varying degrees of introspection.

We then develop the idea of the distance-based view of introspection in Chapter 5, proposing that the manner in which various commonly-used frameworks perform classification has an important effect on their uncertainty when faced with new,

unseen data. We apply the classifiers to synthetic data, showing how their uncertainties vary with data quantity, dimensionality, and distance between training and test data. We then apply them to real data, showing that these effects are persistent when the test data are dissimilar to the training data in classification tasks. Finally, we apply them to real data in object detection tasks.

In Chapter 6 we consider the two extremes of the uncertainty space. First, we apply the classifiers to active learning, examining high-uncertainty decisions. The classifiers choose high-uncertainty test data for labelling by a human oracle, and we investigating the relative performance increases based on particular choices. Secondly, we investigate the effect of introspection when we associate costs with classification tasks, specifically analysing low-uncertainty errors.

In Chapter 7 we apply the idealised classifiers from Chapter 4 to sequential decision making, showing the effects of introspection in agent-navigation problems under uncertainty. We consider the Partially Observable Markov Decision Process (POMDP) framework, and explore how the use of more and less introspective sensors affects the speed with which a robot achieves its goal. We draw a link between the information content of a measurement and the agent's speed of solution.

We conclude with Chapter 8. We discuss the implications of the thesis, summarise our contributions, and propose further avenues for exploration.

1.5 Publications

Some of the work presented in Chapter 2 was presented as "Integrating Metric and Semantic Maps for Vision-Only Automated Parking" at the International Conference on Robotics and Automation in Seattle, USA, May 2015 [Grimmett et al., 2015a]. A further publication featuring our contributions is Furgale et al. [2013].

Part of Chapter 5 is published as "Knowing What We Don't Know: Introspective

Classification for Mission-Critical Decision Making" at the International Conference on Robotics and Automation in Karlsruhe, Germany, May 2013 [Grimmett et al., 2013].

There are two publications relevant to the material in Chapter 6. The work on active learning is motivated by the contents of "Driven Learning for Driving: How Introspection Improves Semantic Mapping", presented at International Symposium on Robotics Research in Singapore, December 2013 [Triebel et al., 2013], with further data sets and conclusions. The work on classical decision making is related to "Introspective Classification for Robot Perception" in the International Journal of Robotics Research in 2015 [Grimmett et al., 2015b].

Chapter 2

Semantic Mapping: A Case Study

In this chapter we present a case study of semantic mapping, which serves to motivate the need for introspection in perception systems to be used for autonomous operation. The case study presented is part of the V-Charge project [vch, 2015][1].

The goal of the V-Charge project is to showcase the feasibility of a system which provides an autonomous valet service. This system must only use close-to-market sensors, and require only a wireless network and a server in terms of external infrastructure. At the end of the project in the Summer of 2015, the consortium successfully demonstrated the following scenario: a user manually drives to their place of work and exits the car at the entrance to the building. On their smart phone they open the V-Charge app, and press the *park* button. They now start their day at the office. Meanwhile, the vehicle communicates with a server, allocating it a charging bay in the car park. The car drives autonomously using the lanes, stopping for other vehicles (autonomous and non-autonomous alike) where required, and docks with the charging bay. When charging is complete, the vehicle relays this information to the server, and is allocated an ordinary parking space in the car park. Once more it navigates autonomously to its intended parking space

[1]The majority of this chapter is published in Grimmett et al. [2015a]

and performs a parking manoeuvre. If that space is in fact recently taken, it can search for another. At the end of the day, the user summons the vehicle using the app, at which time it leaves the parking space and drives once more to the front of the building for collection.

The project completed on time with a number of demonstrations to the press and public, and also led to a number of scientific publications by the consortium, which comprised ETH Zurich, The University of Parma, Braunschweig University, Volkswagen, and Bosch, as well as Oxford University.

Our contribution to the project is entirely within the offline semantic mapping stage, allowing the vehicle to navigate and drive naturally around the car park. We present a system which takes as input (a) the raw data from the vehicle and (b) a geometric map of the car park produced by ETH Zurich, and outputs a semantic map of the area, an example of which is shown in Figure 2.1. The semantic map contains

- the road network of the car park,
- the locations of all the parking and charging bays, and
- the recommended driving speed at any point,

and there are aspects of the map that we would like to detect and label automatically using classification techniques. The semantic map is then used for planning routes around the car park which obey the rules of the road. The author's role in this work is to design and implement the strategy for how to solve this task, particularly for the road network and the recommended driving speed, as well as the active learning components of the parking-space detection.

We distinguish between two types of semantic labels: *static* semantics which represent more permanent features of the environment such as fixed obstacles or points of interaction, and *dynamic* semantics which represent the way in which the

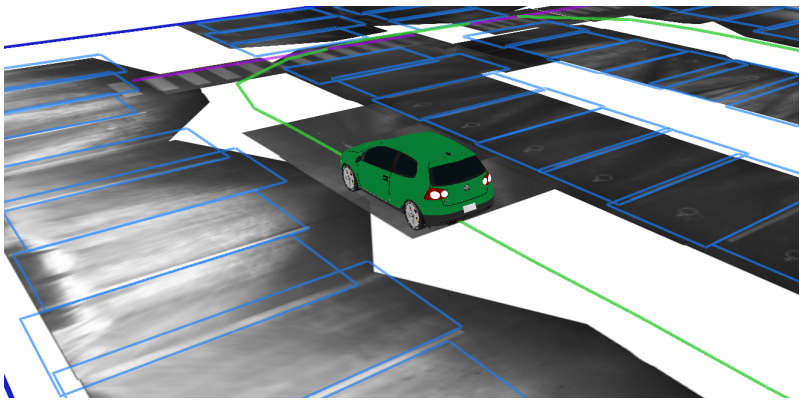


Figure 2.1: The semantic information placed relative to the metric map. The driving lanes are shown in green, and the parking spaces in blue.

environment is shared with other users. The static map represents things like road infrastructure, and is unlikely to change between revisits. The accuracy of this static map is paramount to the functioning of the vehicle, and so a human operator is used to verify it. The dynamic map contains estimates of the likelihood of moving obstacles (in our case, pedestrians) being in any particular location of the robot's environment, allowing the robot to drive at an appropriately slower speed in busy regions of the car park. Dynamic semantics can be recomputed at each revisit in an unsupervised manner, that is, without requiring further human labelling effort.

After a description of the vehicle in Section 2.1, we present a summary of the literature on semantic mapping in Section 2.2. In Section 2.3, we detail a semantic mapping pipeline which satisfies the requirements. Next we present a relative framework for integrating the positions of semantic labels in metric maps in Section 2.4. We finish by discussing the strengths and limitations of our approach, motivating introspective behaviour in our classification systems in Section 2.5.

2.1 The V-Charge Vehicle

The primary goal of the V-Charge project is to demonstrate that an autonomous valet service is possible without using expensive sensors. For this project that means

using cameras and ultrasound sensors costing in the low hundreds of pounds, rather than laser scanners which typically cost in the order of thousands of pounds. The majority of the vehicle's capacity for vision comes from four wide-angle, rolling-shutter cameras, with one on each side of the vehicle. The side cameras are mounted in the wing-mirrors, and the front and back cameras within the Volkswagen emblems, one on the nose of the car and one at the back of the boot compartment. Long-distance forward views are given by a stereo camera mounted on the inside of the windscreen. There are also ultrasound sensors along the forward and rear bumpers.

2.2 Related Works

Although research into autonomous driving predates the DARPA Grand Challenges in 2004 and 2005, they motivated the first leaps and bounds in the field. They were followed by the Urban Challenge in 2007 [Blasch et al., 2006]. There, teams from around the world competed to complete navigation tasks, first off-road and then in simulated town environments. The vehicles had to map (at least locally) and navigate around complex and often dynamic environments in order to reach their objectives, and all of the winning strategies made use of expensive and experimental sensors. These included nodding LIDAR range finders, RADAR, and colour cameras [Urmson et al., 2007, Kammel et al., 2008, Urmson et al., 2008]. Most of the vehicles made use of an Inertial Navigation System (INS) which comprises a GPS antenna and Inertial Measurement Unit (IMU) to inform their localisation system, and two more GPS antennas to estimate the vehicle's absolute heading. Although the tasks in the Urban Challenge included many similarities to the functional requirements of the V-Charge car, such as on-lane driving, appropriate behaviour around other manned vehicles and parking manoeuvres, the sensors they used are still prohibitively expensive for commercial series models.

The PReVENT project [Schulze et al., 2005] makes a step towards autonomous production cars, making use of cheaper sensors like cameras and RADAR, and focussing on the communication of hazards to other vehicles. More recently, Furda and Vlacic [2011] detail a system for making decisions in a two-stage process: the first selects safe and feasible trajectories, and the second aims to maximise comfort and efficiency. They test the system in simulations of city driving. Ibisch et al. [2014] focus on autonomous driving in parking garages, with a collision avoidance system which relies on cameras embedded within the environment. We wish to presume the least amount of infrastructure possible for each car park. The City-Mobil2 project aims to deliver autonomous shuttle pods to several European cities for use on road and around pedestrianised areas, using LIDAR sensors for collision avoidance [Alessandrini et al., 2014]. The current AdaptiVe project aims to produce an autonomous car which helps the user with common driving manoeuvres, such as parking assistance, a city chauffeur that handles roundabouts and intersections, and a co-operative merging function for highway entry, exit, and lane changes [ada, 2016]. The project is still in its infancy and so details such as sensor types are not yet available.

All these cases serve to highlight the V-Charge project as a contribution towards fully autonomous driving and parking in dynamic urban environments using close-to-market sensors.

2.3 Semantic Mapping

Initially we expected that mapping the semantics of car parks would be significantly easier than mapping the open road due to their consistent structure, when in reality they have the same if not greater variety in appearance. Lanes and parking spaces are demarcated in many ways (or not at all), making the job of creating a lane

classifier which works across car parks very difficult. Over the course of the project we mapped three car parks, all of which have very distinct appearances, as shown in Figure 2.2. As a result, developing a system which inherits knowledge from previously experienced car parks is of limited use.

The three components of the semantic map, the road network, parking bay locations, and recommended driving speed are all generated separately, and thus there is a subsection per component.

2.3.1 The Road Network

The requirement of the road network is to allow the vehicle to plan and subsequently drive around the car park which would be predictable for other road users, resulting in a smooth, efficient journey without breaking the rules of the road. The subsequent behaviour of the vehicle is outside of our remit, but this layer of the semantic map contains the information required for the vehicle to be capable of driving appropriately, and thus at least requires knowledge of lanes and intersections.

We experimented with methods for lane marker extraction similar to Aly [2008], but found that due to the varying appearances of lanes in car parks (and some total absences of lane markings), generalising across car parks was impossible. Therefore, we would have needed a hand-tailored solution per car park. This would have not been a good general solution. Therefore, we propose a novel method requiring the survey driver to drive along the centre of each lane at least once, such that the trace of the vehicle centre would map out each lane.

ETH CVG (Computer Vision Group) perform a 3D bundle adjustment over both the visual landmarks and the vehicle positions to produce a consistent *metric map* (a map with distances in some unit, in this case, metres) with loop-closures. Because this required a survey drive through the car park, driving each lane for the road network generation came at virtually no overhead. An example of a this metric map



(a) Stuttgart



(b) Zurich



(c) Wolfsburg

Figure 2.2: Here are pictures of each of the three car parks we mapped as part of the V-Charge project. Note the variety in appearance from one to the next.

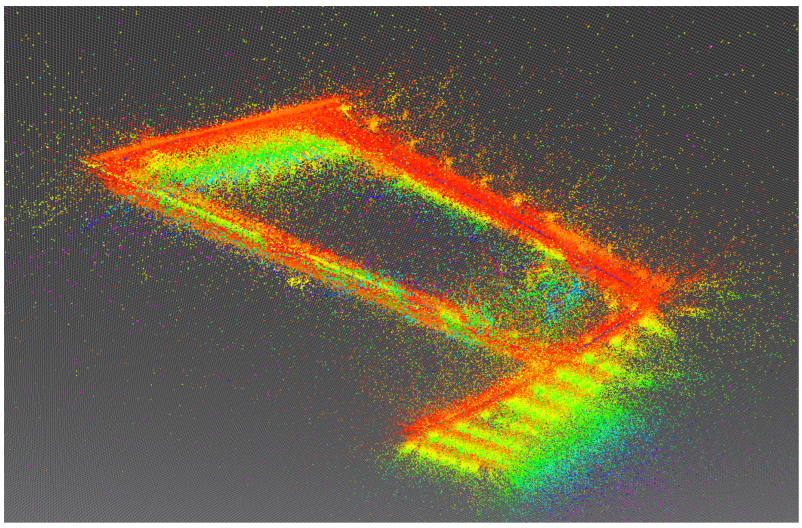
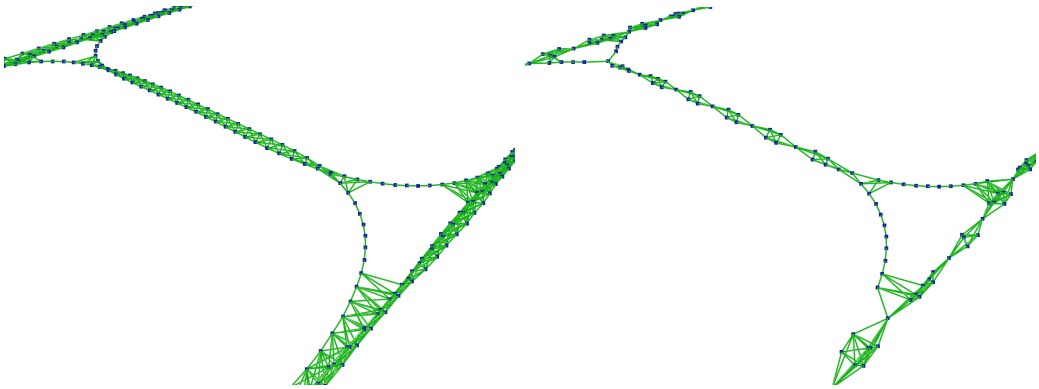


Figure 2.3: The metric map produced by ETH Zurich. This follows a full bundle-adjustment over the vehicle positions and visual landmarks with loop-closure.

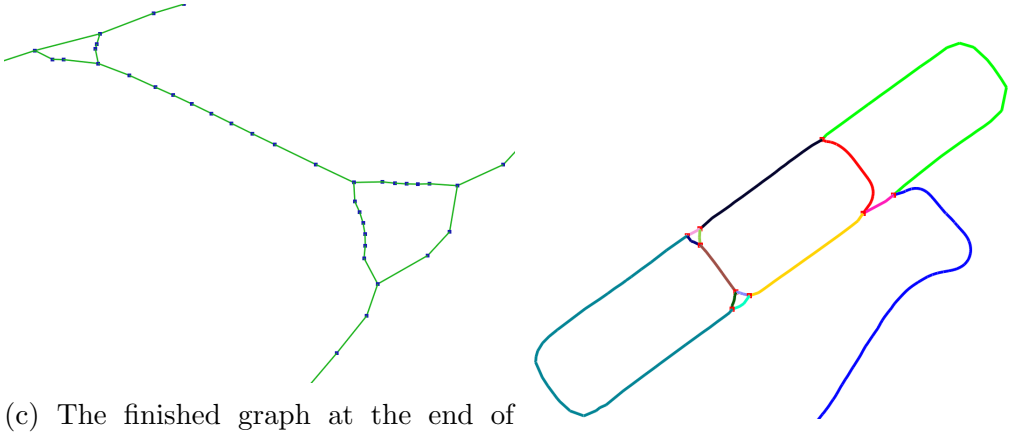
is shown in Figure 2.3. After the survey drive and the optimisation of the vehicle poses, we have a trace that follows the vehicle through the car park. The next challenge is that some of the lanes are likely to have been driven more than once, and due to various sources of noise, those traces will not lie perfectly on top of each other. We need a way of simplifying the overlapping traces to a single skeleton of the underlying lane and junction structure.

We have developed an algorithm for automatically generating a road network, whose only requirement is for the vehicle to have been driven through each lane at least once. To the best of our knowledge, this is the first published system that performs this task. The algorithm takes as input the three-dimensional positions of the vehicle at regular time intervals, and outputs both lanes and intersections. The method is presented in Algorithms 1 and 2 in Appendix A. In summary, we consider the vehicle positions as nodes in a graph, and connect them to nearby nodes in order to simplify lanes which are driven several times with slight displacement each time (see Figure 2.4a). We then repeatedly prune this graph by replacing maximal cliques by their centre point (or rather, an existing node nearest to their centre



(a) The fully connected graph (Algorithm 1, lines 1 to 12).

(b) The graph after one pruning loop (Alg. 1, lines 13 to 25).



(c) The finished graph at the end of Alg. 1.

(d) The lanes as calculated by Alg. 2.

Figure 2.4: The process of calculating the road network applied to the Stuttgart car park.

point, see Figure 2.4b) until we have a skeleton graph (see Figure 2.4c). The final step involves iteratively exploring the pruned graph to find the distinct lanes and intersection points, as shown in Figure 2.4d.

Evaluation

In Figure 2.4 we show some of the intermediary steps and the final result of the road network generation algorithm on the Stuttgart car park. This road structure has been used for autonomous driving. However, because this particular car park is in the ground floor of a 7-story parking lot, we cannot use the INS system to collect ground-truth (combining accelerometer (IMU) and GPS positional information) and

evaluate the accuracy of the road network in the semantic map, and we lack the infrastructure to measure driving errors with a greater accuracy than the metric map. In order to appropriately evaluate the accuracy of the system, we require an outdoor test site, such as the one used for the final demonstrations in Wolfsburg at the VW MobileLifeCampus.

We used the presented algorithm to generate a road network for the car park using a simple mapping route, but as a result of a change in requirement for the demonstrations, large two-way lanes needed to be modelled as two, narrower, one-way lanes side-by-side. Therefore, we used a manual annotation tool to alter the road graph. Further adjustments are made to allow the vehicle to plot a smooth trajectory through narrow barriers. As a result, the final semantic map used in the demonstration is different from the one generated automatically. Nevertheless, we use an INS system to evaluate the accuracy of the final road graph. To generate ground truth, a driver manually drives around each lane as smoothly as possible, recording vehicle position via the INS system. We then compare the trace of that ground truth path with that of the semantic map. The displacements between them can be considered the error, and are shown in Figure 2.5.

The errors are generally very small, with notable exceptions in the North-East corner, the Western corner, and the junction between the two loops. The first is a length of a single wide lane, terminating with a very tight turn around a narrow barrier. The ground-truth driver decided to take the corner as wide as possible, drifting into the adjacent lane going the opposite way as he turned. This is acceptable with no traffic, but not the ideal trajectory for the map. It should be noted that this is a difficult corner even for an experienced driver. The second notable area of error, in the Western corner, is a long stretch of free-driving in a very wide lane. Both the trajectory of the ground-truth driver and the semantic map are equally valid. In the third area of error, the junction between the two loops, the ground-truth driver

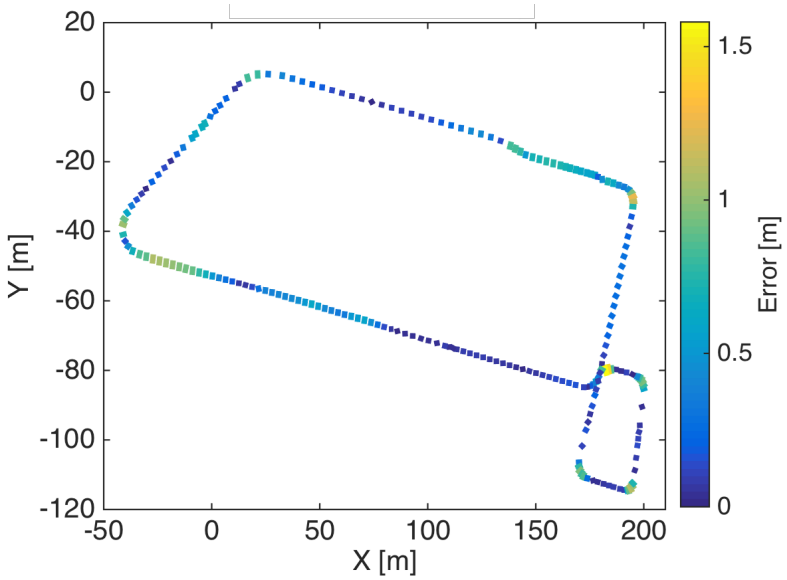


Figure 2.5: The error between the lanes in the semantic map and the ground truth, measured using an INS system in the Wolfsburg car park.

passes very close to another vehicle and staying longer than necessary in the lane going the opposite way. We propose that the trajectory proposed by the semantic map is in fact preferable in the absence of further information about whether the opposite lane is occupied by another vehicle. We recognise that there is no single canonical driving behaviour appropriate for all situations, and so we seek a road network that describes how a manual driver would behave in the most restrictive situation, namely, there are numerous other drivers using the car park. We conclude that despite the worrying nature of the threat of a 1.5m error in the road graph, the routes in the semantic map were are at least as good as those taken by the ground-truth driver, if not better in places. This demonstrates a flaw in the ground-truth data.

2.3.2 Parking Space Locations

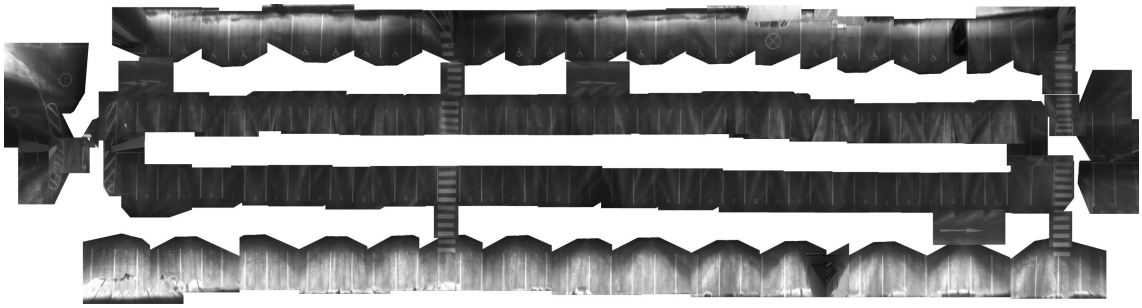
The appearance of parking spaces also varies greatly from one car park to the next. For instance, some parking spaces in Wolfsburg have no painted demarcations at

all, as shown in Figure 2.2c. However, we seek a different solution as for the road network due to the great expense in time that it would incur. We experimented with methods inspired by Seo et al. [2009] and found them to be brittle and unreliable, largely due to the assumptions regarding the structure of the car park, and which line marking would be visible in the overhead image. Therefore, we learn what parking spaces look like in each car park individually. First we use the metric map and the fisheye images from the monocular cameras to produce a synthetic overhead image, and then perform active learning with a human in the loop to detect the parking spaces. In this active learning system, a human manually labels some examples, and trains a classifier which returns the most confident parking-space hypotheses. The human can then accept or reject those hypotheses, and the classifier is re-learned.

The synthetic overhead image

The synthetic overhead image is made by first rectifying the fisheye images, then using the vehicle poses from the metric map to project the image pixels onto a virtual ground plane. When projections from multiple images lie on the same pixel in the ground plane, the running mean of the values is used. The synthetic overhead images we created are shown in Figure 2.6. Note that the quality of the overhead image is determined by the lighting and material properties of the car park, and the quality of the vehicle poses in the metric map.

Note that we use high-resolution aerial photography as an overhead image for the Wolfsburg car park in Figure 2.6c, but this photograph has a much greater resolution than is available in the free online repositories of satellite photography. The latter would not be sufficient for detailed semantic mapping, and so the creation of synthetic overhead images would be required in the absence of high-resolution images.



(a) Stuttgart



(b) Zurich



(c) Wolfsburg

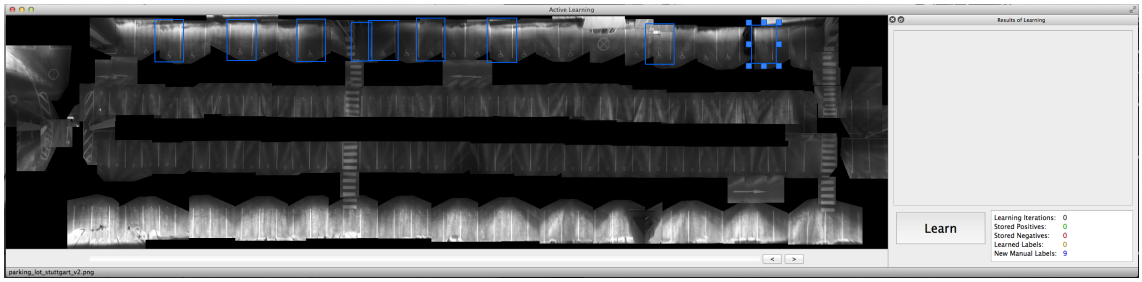
Figure 2.6: The synthetic overhead image from the surveyed car parks. The differences in appearance and lighting conditions of the car parks directly affect the quality and clarity of the synthetic overhead image. No synthetic overhead image of the Wolfsburg car park is required due to the availability of high-resolution aerial photography.

Finding parking spaces

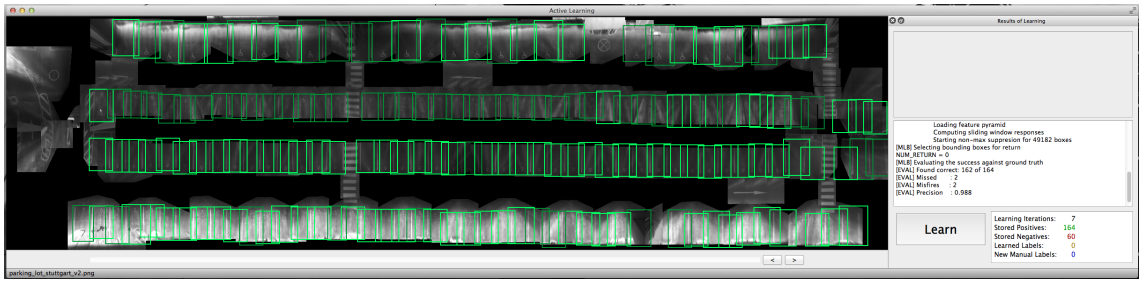
In order to find the locations of the parking spaces, we take the synthetic overhead image (e.g. Figure 2.6) and allow the human annotator to draw rectangles around a few exemplar parking spaces. Then we start the classifier learning process shown in Figure 2.7. 500 negative examples are randomly sampled from the image at various scales. HOG feature descriptors are used [Dalal and Triggs, 2005], and a linear Gaussian process classifier (see Section 5.5.1) is trained on the examples. Passing a sliding window across the image, we extract many test patches and apply the classifier to them. The most confident results are returned to the user as parking spot hypotheses. The size of the sliding window is learned from the hand-drawn labels. The user can now mark them as 'correct' or 'incorrect', and resize them if desired. This operation is faster than drawing new training examples by hand. The process is then repeated until all of the parking spaces have been detected. We performed this in the Stuttgart car park and found that the labelling was very fast, with the user only manually drawing 14 out of 164 parking spaces over 6 rounds of active learning to achieve a precision of over 0.93, as shown in Figure 2.8. If we had to map the remaining floors of this car park, we could re-use this classifier to do so.

Note that the overhead image for the Zurich car park in Figure 2.6b is much more distorted and blurry than the one for the Stuttgart car park. This reduces the effectiveness of active learning, and generally the human will have to do more manual labelling if this is the case.

The Wolfsburg car park (Figure 2.6c) contains large rows of parking areas with no clear demarcations; vehicles are expected to park orthogonally to the lane in a considerate manner. This makes classification of individual parking bays very difficult (there are concrete blocks on the edge of the parking areas to indicate a possible alignment and spacing for vehicles as shown in Figure 2.2c, but this is



(a) Hand-labelling a number of example parking spaces.



(b) The final output of the system, showing all the parking spaces.

Figure 2.7: Shown are two stages of the graphical interface used to perform active learning on the synthetic overhead image. In (a) the human operator labels a few parking spaces, and (b) shows the final output after 6 training rounds. The number of drawn parking spaces is shown in Figure 2.8.

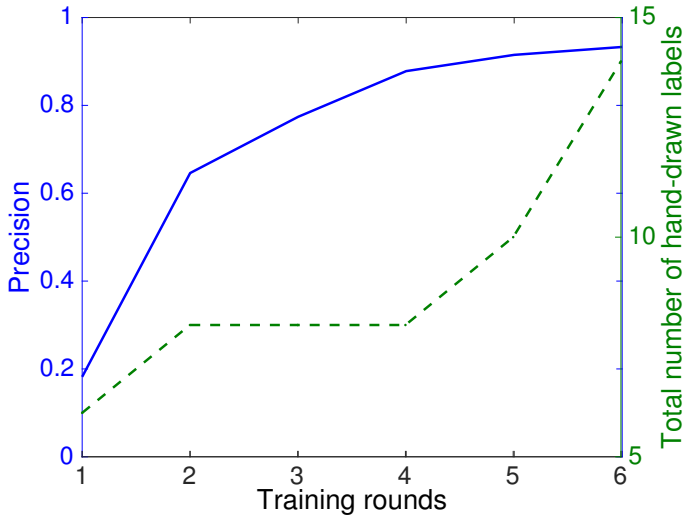


Figure 2.8: The precision of the parking space detections (solid blue) across active learning loops for the Stuttgart car park (using the graphical interface shown in Figure 2.7). The cumulative number of hand-drawn labels is shown as the dashed green line. Note that at each loop, the user labels some of the classifier hypotheses as correct and incorrect, the quantities of which are not shown as it is very quick to label them relative to the effort involved in drawing new labels from scratch.

optional).

2.3.3 Recommended Driving Speed

The third component of the semantic map is the recommended driving speed at any point in the car park. Typically the speed limit is signed, but the most appropriate speed is rarely the maximum. In areas with increased pedestrian activity it would be sensible for the vehicle to drive more slowly. We propose a system which observes where pedestrian activity takes place, and varies the driving speed accordingly.

In summary, we detect pedestrians in the fisheye images, and given the camera pose, project their positions into a metric dynamic map. We maintain a function of pedestrian density over the map, and use it to linearly interpolate between a minimum and maximum driving speed. The pedestrian detection is performed using the same system as the parking space detection shown in Figure 2.7, testing over all the images collected by the vehicle rather than a single overhead image. HOG features are used, as before, and we use ten detection scales, learned from the labelled positive examples.

The problem is represented by a graphical model whose nodes are the set of discrete map locations (or pixels $\mathbf{x} \in \mathbb{N}^2$). A prior probability $p(\mathbf{x}_0)$ on a node is assigned by considering the static map at that pixel location $\mathbf{x}_0 \in \mathbb{N}^2$. If the static map indicates a pedestrian crossing or pavement, the prior on pedestrian density is high, and if it's a driving lane, it is lowest. Given observations z of pedestrians, the maximum a posteriori estimate of pedestrian location is calculated by solving

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \Big(-\log \overbrace{p(\mathbf{z}|\mathbf{x})}^{\text{likelihood}}\, p(\mathbf{x}|\mathbf{x}_0)\, \underbrace{p(\mathbf{x}_0)}_{\text{prior}} \Big), \tag{2.1}$$

where the prior distribution $p(\mathbf{x}_0)$ and likelihood $p(\mathbf{z}|\mathbf{x})$ terms are modelled as normal distributions over the map, making the optimisation linear. To account for the effect

of a node x_i on its neighbours $N(i)$, we impose regularisation by adding linear binary constraints (the second sum in Equation 2.2).

$$\log\ p(\mathbf{x}|\mathbf{x}_0) \quad \propto \quad -\sum_i \parallel x_i - x_{i0} \parallel_2^2 - \sum_i \sum_{j \in N(i)} \parallel x_i - x_j \parallel_2^2 \tag{2.2}$$

$$\log\ p(\mathbf{z}|\mathbf{x}) \quad \propto \quad -\sum_i \parallel z_i - x_i \parallel_2^2 . \tag{2.3}$$

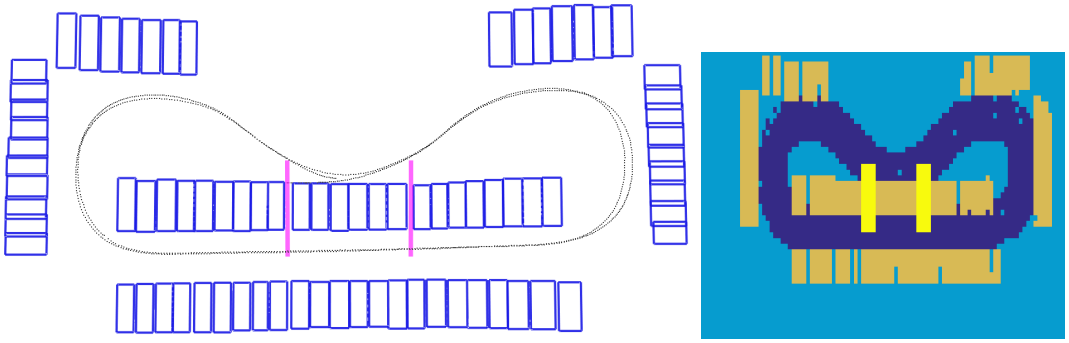
Due to the linear nature of the problem, an exact solution is achieved after a single batch iteration.

Defining minimum and maximum driving speeds (influenced by the signed speed limit), we use the probability to interpolate linearly between the two. A high likelihood of a pedestrian lowers the recommended speed at that location. This produces a map such as those on the right hand side of Figure 2.9.

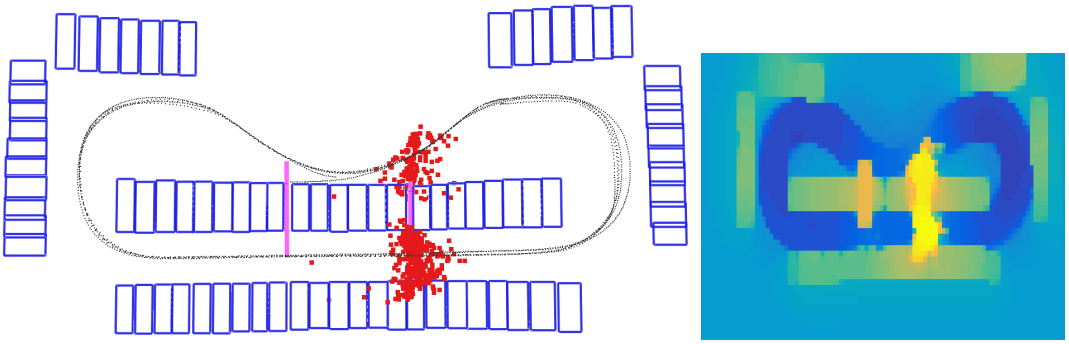
More observations of pedestrians can only improve the quality of the speed map, and thus we show its gradual improvements over six revisits of the car park. During the initial drive there are no pedestrians in the car park, but during each revisit there are pedestrians walking along the pedestrian crossings. Using the same active learning framework used for detecting parking spaces applied to the raw camera images, we detect these pedestrians and then use those as observations to update the graphical model. In Figure 2.9 we show the positions of the pedestrians as red points on the left, and the evolution of p(pedestrian) on the right. It is also convenient that as we see more pedestrians, the classifier becomes increasingly good at detecting them, thus reducing the human labelling effort as time goes on.

2.4 Integrating Metric and Semantic Layers

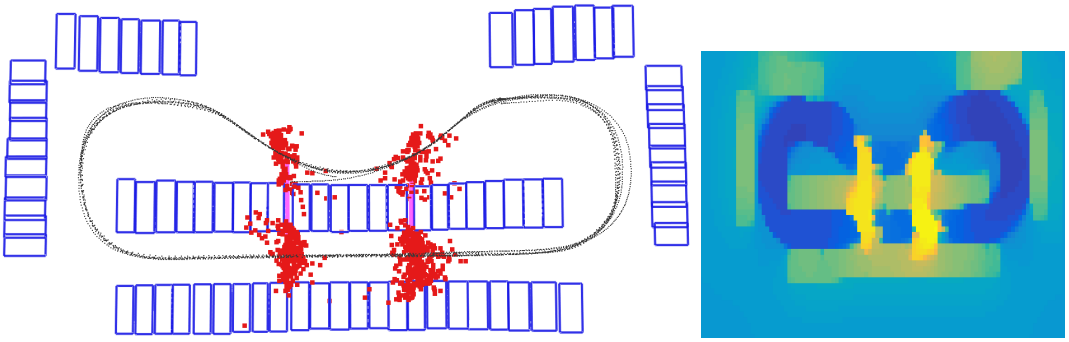
The environments in which our robots operate often contain some features which are continually evolving, along with some more permanent features. Alongside these



(a) Initial drive and speed map prior.



(b) The maps using data up to and including the 3$^{\text{rd}}$ revisit.



(c) The maps using data up to and including the 6$^{\text{th}}$ revisit.

Figure 2.9: On the left we show the red dots represent projections of the detected pedestrians into the static map. On the right we show the evolution of the dynamic map as more pedestrians are detected. In the dynamic map, blue represents lower danger (higher speed) and yellow represents more danger (lower speed). The top dynamic map is calculated using only the prior over pedestrians given the static map.

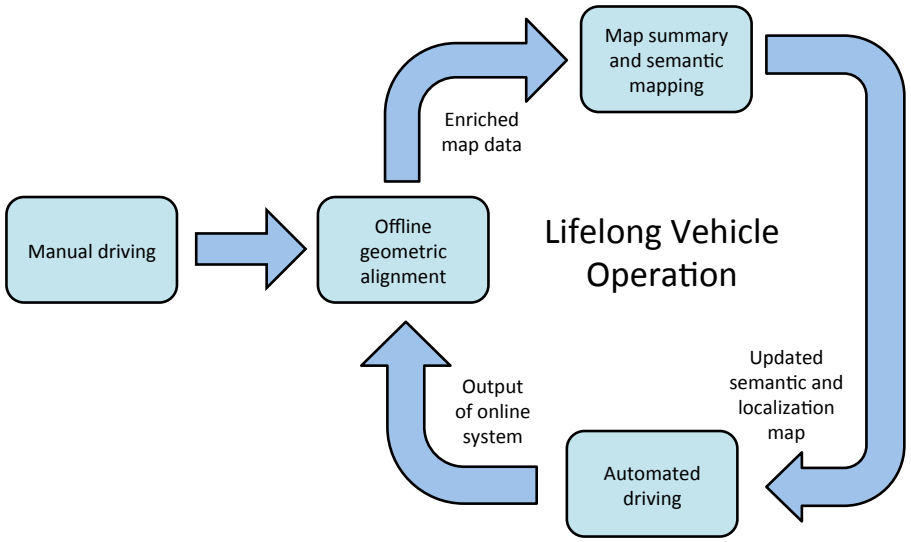
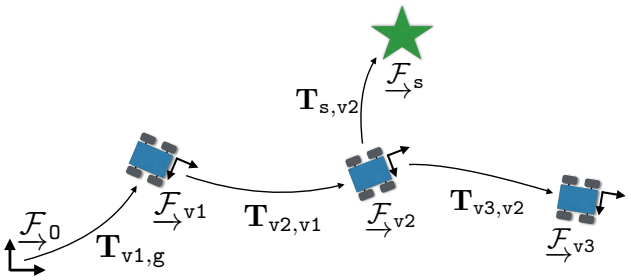


Figure 2.10: The cycle of improvement for both metric and semantic maps as a vehicle autonomously revisits a place.

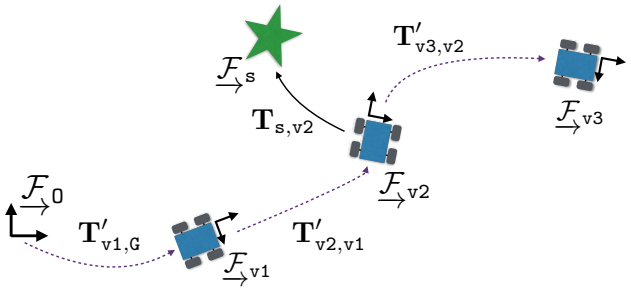
varying scales of change, certain tasks required for autonomous operation can be carried out in an unsupervised manner, making them cheap, while others require significant human involvement due to either the difficulty of the task or the need for accuracy guarantees. Here we propose a framework which strives for the best of both worlds: we manage the reprocessing of tasks based on how often they require updating, and we streamline tasks which require human involvement while maintaining the accuracies required for safety-critical automated driving.

We envisage a system by which our metric and semantic layers improve as our robots revisit previously-explored areas, as shown in Figure 2.10. The data collected during a revisit can be used to refine the metric layer from the previous visit, and that in turn can be used to refine aspects of the semantics. For instance, better loop closures improve the metric layer, and those changes propagate through to the positions of the semantic labels.

We do this by performing the initial labelling in a frame of reference local to the sensor in which the object is visible. This is such that when the metric map is recomputed, the new position of the semantic label is an unmodified local trans-



(a) Initial visit



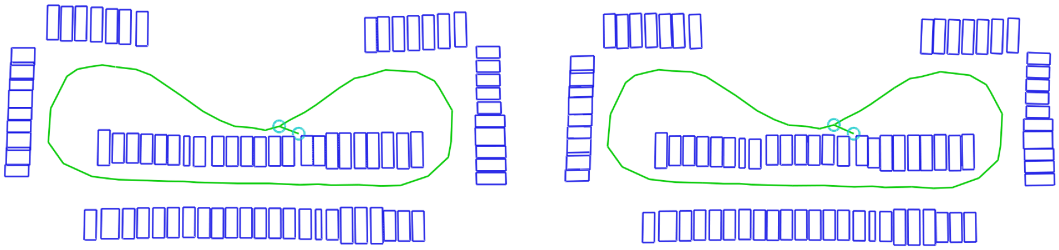
(b) Revisiting the same place

Figure 2.11: While the vehicle positions [v1, v2, v3] may be recomputed between (a) the initial visit and (b) the revisit, the position of the semantic object s relative to v2 remains unchanged. This allows us to update the metric map without having to recreate the semantic labels. $\mathcal{F}_0$ represents the global origin coordinate frame, and $\boldsymbol{T}_{a,b}$ represents a 6 d.o.f. transformation from b to a.

formation of the newly computed vehicle frame. Figure 2.11 demonstrates this principle. Structuring localisation problems in this relative sense is an important principle in SLAM (Simultaneous Localisation and Mapping), e.g. Durrant-Whyte and Bailey [2006], Sibley et al. [2009]. It is used in a similar manner in this context.

For the evaluation of the improvement cycle and automated parking we use an underground car park in Zurich. There are 80 parking spaces which vary in size depending on their position relative to the edges and corners. The parking spaces are delineated by yellow markers as shown in Figure 2.2b.

The data are organised as follows: the car is driven twice around the car park as an initial visit, and then there are six subsequent 'revisit' loops. During each revisit



(a) Initial visit (b) First revisit

Figure 2.12: The evolution of the semantic map through the original drive, and the first revisit. The parking spaces are the blue rectangles, lanes are in green and the intersections are cyan circles. Notice that the top and middle rows of parking spaces shift from (a) to (b) (see, for instance, the top row of parking spaces) as the changes in the metric map propagate through to the positions of the semantic labels.

loop, pedestrians walk between the pedestrian crossings; in revisit loops 1 to 3, they walk along the right-hand pedestrian crossing, and then in loops 4 to 6 they walk along the left-hand pedestrian crossings, as shown in the left hand side of Figure 2.9.

We then use these loops to emulate the process in Figure 2.10: the base map (comprising both metric and semantic components) is created using the initial run, then as we iterate the evolution cycle, the next map comprises the initial loop plus the 1st revisit. The third map comprises the initial run, the 1st and 2nd revisits, and so on. In total we have seven maps, each more informed than the previous one.

Because we want to minimise the human labelling effort, we only label the parking spaces and pedestrian crossings once, in the base map. The synthetic overhead image used for creating those labels is shown in Figure 2.6b. However, because the labels are associated with the vehicle positions from which they were visible in the images, the changes in the metric map shifts the global positions of those semantic labels as the maps evolve. This evolution process is shown in Figure 2.12.

Next we use the first four maps to test the repeatability of the autonomous parking system. ETH Zurich performed the following actions:

1. The car is driven to a point 20m away from the desired parking location.

2. The car localises itself in the metric map and uses the semantic map to drive autonomously along the lane and park in the parking space.

3. Next, the car is manually driven out of the parking space to the same starting location 20m away, and the process is repeated for a total of five times per map, resulting in a total of 20 parking manoeuvres.

Calculating the ground truth of localisation systems for mobile platforms is an open problem, so we have done this by estimating the position of the vehicle relative to a chequerboard at a fixed position in the car park which is visible in the camera images. The camera's intrinsic and extrinsic (relative to the car) parameters are known, and the size of the chequerboard is known. Having registered the chequerboard in the image, it is straightforward to calculate the vehicle pose relative to the chequerboard. The parking positions are shown in Figure 2.13. This shows that despite the fact that the metric maps are recalculated in an unsupervised manner and the positions of the semantic labels shift accordingly, the parking accuracy is very consistent, with a significant cluster of points within 0.13m and the few outliers never varying by more than 0.3m.

We attribute these outliers to errors in the odometry, and the fact that the parking planner plotted a bad course for the vehicle during one of the parking manoeuvres during the first revisit. Ideally we might expect the parking accuracy to improve over successive revisits. We believe that the accuracy of the base map is so high that already the parking accuracy is limited by the planning module and controller noise, rather than by a lack of geometric information.

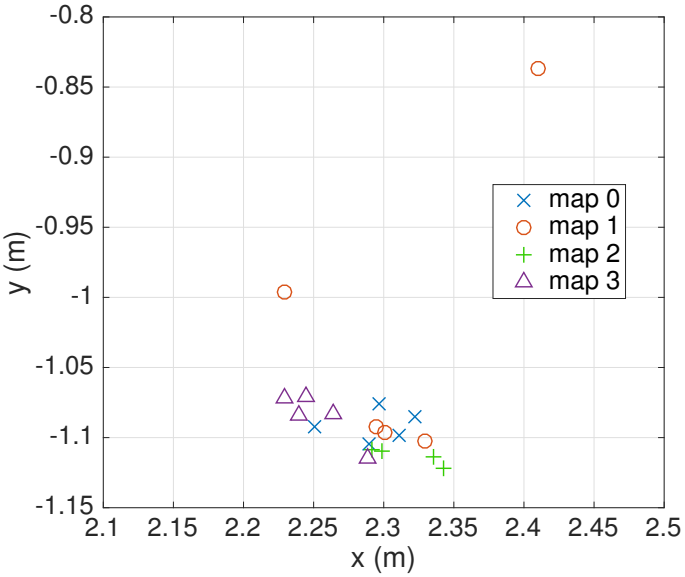


Figure 2.13: The positions of the five autonomous parking manoeuvres of the vehicle in map for maps 0 to 3. Map 0 comprises the initial drive, then map 1 includes both the initial drive and the 1st revisit, then map 2 includes those for map 1 plus the 2nd revisit, and so on. The displacement between the parking centres and the position of the parking spot (at the origin) is due to the difference between the vehicle centre and the centre of the parking spot.

2.5 Conclusions

We have explained and demonstrated a system which creates semantic maps of car parks, by making use of a survey run of the lanes and the metric map generated from that survey. As a result of the need for extremely accurate labels in an environment where humans and robots operate together, we have made use of a human supervisor to verify, and in some cases create these labels. This is the status quo of semantic mapping. The system is a success because it leads to a map with the requisite accuracy, with supervised automation of the individual components. In our system, the limitations involve the degree to which human input is required. That degree depends upon the complexity of a particular car park, but ultimately every label is verified by a human at some stage. This mistrust in the output of our classifiers is what drives us towards the need for introspection. These semantic mapping systems must become fully autonomous, and for that we require a better treatment

of classification confidence.

If we temporarily accept a need for human input, we should strive for a system that minimises the amount of human effort before the maps can be considered sufficiently accurate. We propose that the first step to full autonomy is the accuracy self-evaluation of a particular semantic map. For example, the most questionable labels should be checked first, and the most confident sections need not be checked at all. This motivates the need for a classifier to provide feedback on any particular classification: how likely is it to be correct? Many classification frameworks have a method of scoring a classification, and often that score is used to calculate a 'probability of class'. However, for us to be able to trust the most confident classifications without human intervention while checking the most uncertain ones requires the probability associated with a classification to be correlated with its correctness.

More generally, creating labels in a semantic map is akin to taking decisions, and indeed our conclusions are general to decision-making in robotics. We would like decisions which are likely to be wrong to be made with high uncertainty, and the low-uncertainty decisions to be correct. A classifier which achieves this would have an appropriate sense of its own knowledge, and could be considered *introspective*.

If this correlation between correctness and confidence is present, as classifiers improve their confidence will increase, requiring progressively less human intervention. This will lead us, ultimately, to full automation in semantic mapping.

In Chapter 4 we develop this idea of introspection, and later go on to demonstrate its importance in a variety of decision-making tasks.

Chapter 3

Data Sets, Features, and Performance Metrics

In order to investigate the consistency of the introspective capacities of the various frameworks, we evaluate our experiments on commonly-used data sets which encompass several domains of robotics. Those data sets are described here, along with the features we use to describe them, and the performance metrics typically employed to compare classifiers.

In Sections 3.1, 3.2, 3.3, and 3.4 we outline the characteristics of the four 'real' data sets used in this thesis. In Section 3.5 we describe the synthetic data set used in Chapter 5 to highlight the indicators for introspective capacity. In Section 3.6 we describe the feature representations used to encode images from the real data sets as vectors for classification. Finally, in Section 3.7 we define the standard metrics of success applied to classification tasks in machine learning and robotics.

Parameter	TLR	GTSRB	DP	KITTI
Cropped image height	30	32	96	26
Cropped image width	12	32	48	32
HOG cell size	n/a	n/a	10	10
N. of orientations	n/a	n/a	5	6
Final feature dimension	200	200	950	198

Table 3.1: The parameters for the features for the TLR and GTSRB (using template features) and the DP and KITTI data sets (using HOG features).

3.1 Traffic Lights Recognition

The Traffic Lights Recognition (TLR) dataset [of Mines ParisTech, 2010] is a sequence of images taken by a monocular camera from a car driving through central Paris, an example of which is shown in Figure 3.1a. The TLR dataset comprises just over 11,000 frames, in which most of the traffic lights have been labelled with bounding boxes and further metadata such as the colour of the light or whether a particular label is ambiguous (e.g. the image suffers from motion blur, the scale is inappropriate, or the viewing angle is oblique). A few traffic lights have been omitted altogether. As recommended by the authors, we exclude from our experiments any labels of class *ambiguous* or *yellow signal* and any instances which are partially occluded. We split the dataset into two parts (at frame 7,200 of 11,178), with an approximately equal number of remaining labels in each part and with no physical traffic lights in common. Positive data are extracted as labelled. Negative *background* data are extracted by sampling patches of random size and position from scenes in the dataset while ensuring that the patches do not overlap with positive instances. The images are resized according to the parameters in Table 3.1.



(a) TLR



(b) DP



(c) KITTI

Figure 3.1: Example images from the various data sets.

			
roadworks ahead (1500)	*right ahead* (688)	*stop* (780)	*keep left* (298)
			
lorries prohibited (420)	*speed limit* (1980)	*yield* (2159)	

Table 3.2: The seven classes of the German Traffic Sign Recognition Benchmark (GTSRB) dataset considered in our work. The numbers in brackets indicate the number of data available per class.

3.2 GTSRB

The German Traffic Sign Recognition Benchmark dataset [Stallkamp et al., 2012] comprises over 50,000 loosely-cropped images of 42 classes of road signs, with associated bounding boxes and class labels. From this dataset we specifically focus on the seven classes shown in Table 3.2, chosen arbitrarily. The images are resized according to the parameters in Table 3.1.

3.3 Daimler Pedestrian

For pedestrian detection, we use the non-occluded monocular intensity images from the Daimler multi-cue occluded Pedestrian data set (DP) [Enzweiler et al., 2010], examples of which are shown in Figure 3.1b. There are over 52,000 positive and 32,000 negative examples split into training and test sets. The images are resized according to the parameters in Table 3.1.

3.4 KITTI

The KITTI data set [Geiger et al., 2012] comprises over 7,400 non-sequential colour images from a camera pointing out from the front of a car driving through a German city, an example of which is shown in Figure 3.1c. In this data set we are detecting vehicles, with 3-5 in a typical frame. The images are cropped and resized according to the parameters in Table 3.1.

3.5 Synthetic Data

The data in the data sets already described in this section are derived from real images of complex real scenes captured by noisy cameras and described using various feature descriptors. As a result, the data sets are difficult to characterise and visualise, and thus it is problematic to make fundamental conclusions regarding the nature of the classification frameworks relating to distances in feature space. In order to carry out an in-depth investigation of the effects of introspection we need a greater degree of control over the data.

The fundamental effects on classifier uncertainty we want to explore are those which are apparent as we test the classifiers on data which do not closely resemble those seen in the training data, as motivated by Section 2.5. We start by creating a training set from the training distribution

$$X^{+}_{\text{train}} \sim \mathcal{N}_d(\mu^{+}_{\text{train}}, \sigma^{+}_{\text{train}}), \text{ and} \tag{3.1}$$

$$X^{-}_{\text{train}} \sim \mathcal{N}_d(\mu^{-}_{\text{train}}, \sigma^{-}_{\text{train}}) \tag{3.2}$$

where $\{+, -\}$ denote the positive and negative classes, d is the dimensionality of

the data, and

$$\mu^{+}_{\text{train}} = 0.8 \cdot u_1, \tag{3.3}$$

$$\sigma^{+}_{\text{train}} = 0.3 \cdot u_2, \tag{3.4}$$

$$\mu^{-}_{\text{train}} = -0.8 \cdot u_3, \tag{3.5}$$

$$\sigma^{-}_{\text{train}} = 0.8 \cdot u_4, \tag{3.6}$$

$$\text{where } u_i \ \sim \mathcal{U}_d[0, 1]. \tag{3.7}$$

These distributions are chosen to allow randomness between runs, and to keep the training data with a mean close to 0 and a standard deviation under 1. This is recommended by the SVM training manual [Hsu et al., 2010], for no further data normalisation takes place after they are drawn from the distributions. These recommendations originate from the implication that the optimisation in the SVM may be poorly conditioned if the data are distributed very differently from these values.

Note that generally, $\sigma^{+}_{\text{train}} < \sigma^{-}_{\text{train}}$. This is because we will largely be tackling detection tasks, in which the negative background class is inherently more varied than the positive class (for more information see Section 5.7). We then perturb the means and standard deviations of the training distributions to create the test distributions:

$$\mu^{+}_{\text{test}} = \mu^{+}_{\text{train}} + M^{+}_{\mu} \ \cdot \ n_1, \tag{3.8}$$

$$\sigma^{+}_{\text{test}} = \sigma^{+}_{\text{train}} + |M^{+}_{\sigma} \ \cdot \ n_2|, \tag{3.9}$$

$$\mu^{-}_{\text{test}} = \mu^{-}_{\text{train}} + M^{-}_{\mu} \ \cdot \ n_3, \tag{3.10}$$

$$\sigma^{-}_{\text{test}} = \sigma^{-}_{\text{train}} + |M^{-}_{\sigma} \ \cdot \ n_4|, \tag{3.11}$$

$$\text{where } n_i \ \sim \mathcal{N}_d(0, 1). \tag{3.12}$$

where M^{+}_{μ}, M^{+}_{σ}, M^{-}_{μ}, and M^{-}_{σ} are scalars and denote the magnitude of the pertur-

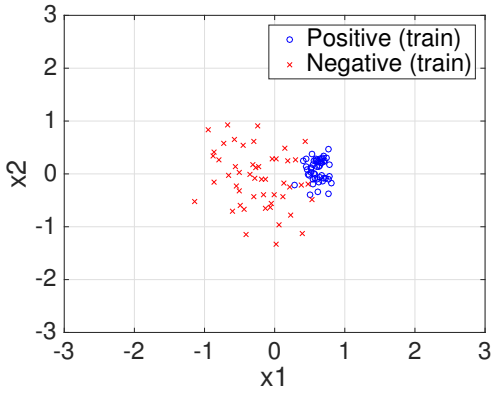
bation. Choosing a large value for M could indeed move the test data outside the recommended bounds for SVM training, but when it comes to real data we cannot always predict a normalisation of the data which will guarantee that every test datum will also lie in the recommended bounds, so we consider this to be acceptable. Figure 3.2 shows example of training and test distributions with varying parameters.

3.6 Features

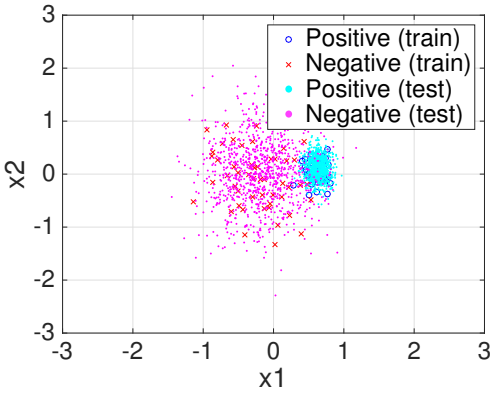
In the previous section we describe the data sets used throughout this thesis, and here we describe which feature descriptors we use for those data sets. For those with consistency in appearance for the positive class, the TLR (traffic lights) and GTSRB (road signs) data sets, we use template features, leveraging the consistent cropping and aspect ratio of the image windows. For those with a great variety in the positive class, the KITTI (cars) and DP (pedestrians) data sets, we use the Histogram of Oriented Gradients (HOG) features which have been shown to work well in these domains [Dalal and Triggs, 2005, Ziegler et al., 2014]. These are described in more detail in this section. Note that neither of the features considered here are rotationally invariant. It is common practice to include many permutations of each labelled example in the training set by adding small rotations, scalings, exposures, and blur. However, these techniques are not used in this thesis.

3.6.1 Template Features

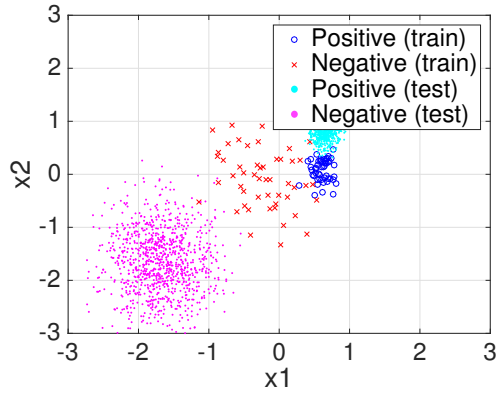
A rich body of work on the detection and classification of road signs and traffic lights has established a successful track record of template-based features for this purpose. Specifically, we leverage the approach proposed by Torralba et al. [2007] in which a dictionary of partial templates is constructed, against which test instances are matched, as described in Figure 3.3a. A single feature consists of an image



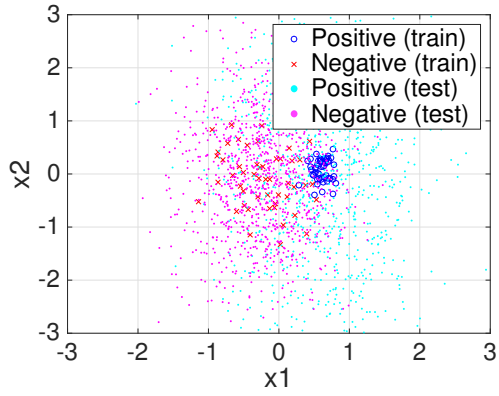
(a) Training data only



(b) $M_\mu^+ = M_\sigma^+ = M_\mu^- = M_\sigma^- = 0$



(c) $M_\mu^+ = M_\mu^- = 1$, $M_\sigma^+ = M_\sigma^- = 0$



(d) $M_\mu^+ = M_\mu^- = 0$, $M_\sigma^+ = M_\sigma^- = 1$

Figure 3.2: We demonstrate the effects of changing the values for M: the magnitudes of the differences between the training and test distributions. In this case each datum is two dimensional, $x_i = [x1_i, x2_i]$. (a) shows the training data, which are also superimposed onto (b), (c), and (d). These last three show some test data for three sets of values of M: in (b) the training and test distributions are the same, in (c) we perturb the means, and in (d) we perturb the standard deviations. Note that the training data are all generated from the same process as described in Section 3.5.

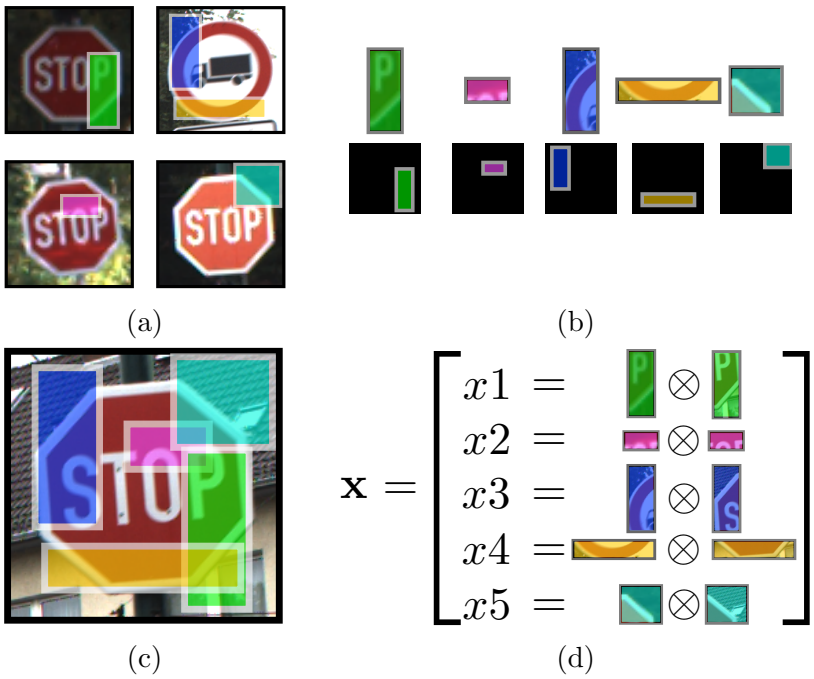


Figure 3.3: Generating template features for an image.
(a) First, sample patches from a collection of images.
(b) Their pixel values and positions within the original image form the dictionary.
(c) To calculate the feature for an image, first extract the pixel values at the appropriate positions in the image to match the patches in the dictionary, and
(d) evaluate the normalised cross correlation between the patches in the image and those in the dictionary. The resultant feature vector has one element for each patch in the dictionary.

patch (ranging in size from 8×8 to 14×14 pixels) and its location within the object as indicated by a binary mask ($h \times w$ pixels according to Table 3.1). For any given test instance, the normalised cross-correlation is computed for each feature in the dictionary and hence the resulting dimensionality of the feature. Therefore, we obtain a feature vector of length d per test instance, where d is the size of the dictionary. We found empirically that $d > 200$ leads to negligible performance increase in classification. Throughout our experiments we therefore set $d = 200$.

	Training data		Test data	
Data set	Positives	Negatives	Positives	Negatives
TLR	250	500	1000	8000
DP	250	500	2000	16000
KITTI	250	500	1000	10000

Table 3.3: The number of training and test data of each class used for the detection experiments in Section 5.7. The quantities of data from the GTSRB data set for the classification experiments are detailed in Section 5.6.2.

3.6.2 Histogram of Oriented Gradients (HOG)

For the Daimler Pedestrian and KITTI data sets, we have chosen to use Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] as features. These have been shown to perform well for pedestrian detection [ibid]. We use the implementation in *vlfeat* [Vedaldi and Fulkerson, 2010] and use parameters as detailed in Table 3.1.

3.7 Performance Metrics

In this section we outline some of the standard classification performance metrics used in machine learning and robotics, particularly the ones we refer to in this thesis. For further details, see Powers [2011].

The metrics described here use a vector of classification scores and the ground truth information to measure some aspect of performance. In the context of this thesis, these scores all lie in the range $[0, 1]$ and the ground truth is binary.

3.7.1 Precision, Recall, Accuracy, and F-measure

Given the scores, the ground truth, and a decision boundary, we can calculate the number of the following outcomes: true positives, true negatives, false positives, and false negatives. These totals are denoted by tp, tn, fp, and fn, respectively. They

are used to calculate precision (P), recall (R), and accuracy (A):

$$\mathrm{P} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}} \tag{3.13}$$

$$\mathrm{R} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}} \tag{3.14}$$

$$\mathrm{A} = \frac{\mathrm{tp} + \mathrm{tn}}{\mathrm{tp} + \mathrm{tn} + \mathrm{fp} + \mathrm{fn}}. \tag{3.15}$$

Precision answers the question: "out of all the examples which test positive, what proportion of them are actually positive?". *Recall* answers the slightly different question: "out of all the examples which are actually positive, what proportion of them tested positive?".

Accuracy is the proportion of the tests which were correct. For this reason, in very biased data sets a classifier can report very good accuracy despite always misclassifying the rarer class. Given that many of the data sets used in this paper are biased, we prefer to use F-measure, which is a combination of precision and recall. F-measure takes an extra parameter $\beta \in \mathbb{R}^+$, which may be interpreted as the relative importance of precision to recall. F_β is defined as the weighted harmonic mean of precision and recall given by

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2}\Big(\frac{1}{P} + \frac{\beta^2}{R}\Big). \tag{3.16}$$

We rearrange this to give the standard form

$$F_\beta = (1+\beta^2)\frac{\mathrm{P} \times \mathrm{R}}{\beta^2 \times \mathrm{P} + \mathrm{R}}. \tag{3.17}$$

Throughout this thesis we weight precision and recall equally, setting β to 1 and

therefore reducing the equation to

$$F_1 = 2 \cdot \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}. \tag{3.18}$$

All of these metrics require a decision threshold on the classification scores in order to count the number of each potential outcome. A common approach is to calculate the desired metrics for many values of the decision threshold from $-\infty$ (where all tests are positive, so precision is 0 and recall is 1) to $+\infty$ (where all tests are negative, and so the precision is 1 and recall is 0). The values for precision and recall for each decision threshold are plotted on a graph (called a precision-recall or P-R curve). The decision boundary is then chosen by the user for an appropriate trade-off of precision and recall suitable for the task at hand.

Where we report these metrics by a single value rather than a P-R curve, the decision threshold has been set to 0.5.

Chapter 4

Introspection

In Chapter 2 we motivated the importance of knowing the confidence in a semantic label, on the basis that we could prioritise the human oversight of the labels which are least likely to be correct. More generally, all decisions made by robots should made with a sense of confidence in mind. In this chapter we consider the requirements for a classifier to make appropriate decisions.

In Section 4.1 we present a novel basis for the ideal classifier. We model a classifier as a set of probability distribution functions, one per class. This model is used to define the ideal behaviour of an imperfect classifier. After an examination of the literature in Section 4.2, we derive a number of these idealised classifiers which satisfy that ideality to varying degrees in Section 4.3. Despite their varying levels of idealness, they will all have the same accuracy, and thus perform equivalently in terms of traditional metrics. It is not the overall number of errors which differs between them, but rather the confidence of those errors. These will be used as a baseline to examine the behaviours of a number of commonly-used classification frameworks introduced in Chapter 5.

4.1 The Ideal Classifier

The ideal classifier is one which, given some vector $\mathbf{x}$ representing a test datum, determines to which class in the set $\mathcal{C} = \{C_1, \ldots, C_c\}$ the datum belongs, while never making an error. This is a lofty goal in the context of object detection in images, where for instance the top rated algorithm detecting cars in the KITTI vision benchmark [Geiger et al., 2012] does not exceed 90% average precision (in the 'moderate' difficulty category) at the time of writing.

A classifier with limited knowledge placed within an open-world situation cannot make perfect decisions. Therefore, rather than demanding that a classifier be perfect, we might demand that it be perfect some of the time, and the rest of the time it will say, 'I don't know'. This would allow a robot to take some alternative, safe action when it encounters an object it cannot classify, thereby guaranteeing safe operation without requiring perfection. The key advantage of this behaviour would be that the user or decision-making system can determine which classifications are correct and which may be wrong.

We restrict ourselves henceforth to binary classification frameworks, which determine inputs to be on some scale with extremities indicating the respective classes C_1 and C_2. This may be done via scores in the interval $[0, 1]$, in which scores near 0 are more indicative of C_1 and scores near 1 are more indicative of C_2.

Consider a situation in which a robot wants to cross the road. On board the robot is a classifier, which informs it about whether the road is clear, or a car is approaching. The robot has two possible actions to choose from: it can *go* or it can *wait*. The ideal behaviour of the robot is to *go* when the way is clear, or *wait* when a car is approaching. Based upon the output from the classifier, the robot decides which action to execute.

The state of the road can be in either of two classes, C_1 denoting that the road

is clear, and C_2 denoting that a car is approaching. The *go* action is denoted by a_1, and the *wait* action is denoted by a_2. In the moment of the decision, the classifier provides the robot with a score (or *measurement*) $z \in [0, 1]$ which gives information about the state of the road (as above). Let z be a draw (or *sample*) from a random variable Z. Crucially, let us define a classifier to be a pair of functions $f_1(z)$ and $f_2(z)$, where $f_i(z)$ is the probability density function of the measurement associated with class C_i, or $f_i(z) = f(z \,|\, C_i)$.

Note that this is not a parallel mechanism to the real classifiers considered later on in this thesis. Those real classifiers take as input a feature of an image patch, x, and can be modelled as $p(z \mid x)$. That is to say, given the feature x, the classifier gives the following measurement or output. The idealised classifiers considered here are *simulations* of classification output given the *ground truth*, and should not be thought of as mechanisms to classify data. With the idealised classifiers, at no point in the process is there a feature x. As a result of this, they are less data set dependent than real classifiers; to simulate their output on a data set, you only need know the ratio of positives to negatives.

Consider a decision making process in which there is a threshold T such that: action a_1 (*go*) is taken if $z < T$ and action a_2 (*wait*) is taken if $z > T$.

For a perfect classifier, the probability density functions $f_1(z)$ and $f_2(z)$ might resemble those in Figure 4.1. In this case the optimal threshold is given by $T = 0.5$, perfectly separating the two density functions. If the true class is C_1 (the way is clear), the resulting value of z is always in the range $[0, 0.5)$, so the robot will always choose to *go*, which is the correct decision. Conversely, if the class is C_2 (a car is approaching), the resulting measurement z will lie in the range $(0.5, 1]$, and so the robot will *wait*, which again is the correct decision. If we were to plot the probability of an error given some measurement z, it would be zero over the entire range $[0, 1]$.

Drawing inspiration from probability theory, we might interpret $z = 0$ to mean

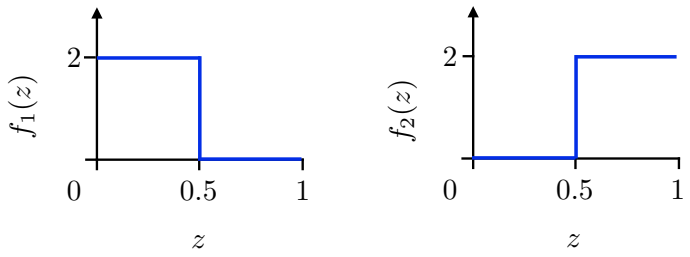
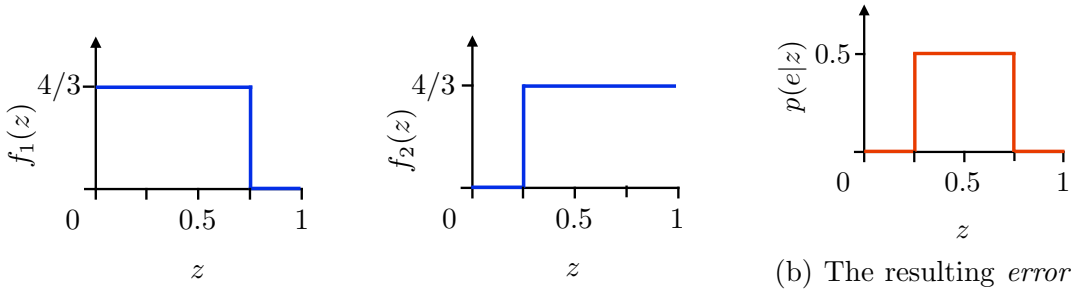


Figure 4.1: The probability density functions for a perfect classifier in the binary case. A 'classifier' here returns a measurement z given a test datum. $f_i(z)$ is the probability density function of the random variable Z when the test data are from class C_i. A test datum from class C_1 will always return a measurement $z < 0.5$, and conversely a test datum from class C_2 will always yield a measurement $z > 0.5$. With a decision threshold T of 0.5, the chosen decision will be perfect.



(a) The probability density functions for a classifier which can make errors between $0.25 < z < 0.75$, but is perfect elsewhere.

(b) The resulting *error function* given the probability density functions in (a). Let e be the event of a classification error.

Figure 4.2

the classifier is very confident that the way is clear (C_1), and $z = 1$ to mean it is very confident that there is a car approaching (C_2). In the absence of further information, measurements within the region around $z = 0.5$ would be the most ambiguous. For example, if we consider the functions $f_i(z)$ in Figure 4.2a, the interval $0.25 \leq z \leq 0.75$ is the region within which mistakes can be made. This is illustrated by showing the probability of error given the measurement, $p(\text{error} \mid z)$ in Figure 4.2b, assuming equal likelihoods of C_1 and C_2. Henceforth in this thesis we refer to $p(\text{error} \mid z)$ as an *error function*.

For this scenario and imperfect classifier, we might decide that the robot should

only cross the road if there is no chance of a car approaching, thus always avoiding a collision. Therefore, if there is a strictly positive probability of a false negative error (choosing the *go* action if there is a car approaching, C_2), the robot should *wait*. For this 'top hat' classifier (the error function resembles a top hat), the optimal decision boundary would be $T = 0.25$, such that half of all opportunities to cross the road are taken, and whenever there is a chance of error it chooses to *wait*. Thus, it satisfies the requirements as outlined earlier in this section.

If we modify this top hat classifier by changing $f_2(z)$ (the right hand side of Figure 4.2a) such that there is a small but non-zero probability of z lying in the range $[0, 0.25]$, the optimal decision boundary with the requirement of never making a false negative error will be $T = 0$, resulting in the robot always *waiting* and never *going*. This is safe, but even if the likelihood of an error is extremely small, the robot will never cross. Desiring a behaviour which reflects the likelihood of an error, we might choose to relax the constraint, instead choosing some acceptable likelihood of a false negative error, which we call ϵ. The choice of ϵ together with the characteristics of the classifier ($f_1(z)$ and $f_2(z)$) allow us to optimise

$$\min_T p(e) \tag{4.1}$$

$$\text{subject to } \; p(e \,|\, C_i) \leq \epsilon \tag{4.2}$$

$$\text{for some predetermined } i \text{ in } \{1, 2, ...\}. \tag{4.3}$$

A classifier with some $p(e)$ subject to (4.3) will make more appropriate decisions than another with a larger $p(e)$. One way to reduce $p(e)$ is to find a classifier whose $f_1(z)$ and $f_2(z)$ overlap as little as possible, allowing an apt choice of the threshold T to effectively separate true and potentially erroneous classifications. Thus, the classifier from Figure 4.2 is very effective in decision making. Intuitively, the classifications near $z = 0$ and $z = 1$ are confident, and those near $z = 0.5$ are

uncertain. Crucially, it is the ability to make correct decisions with confidence while making mistakes with uncertainty which causes this separation, and therefore allows the classifier to make good decisions. We call this behaviour *introspection*, because the classifier is *introspective* if it is able to make an appropriate assessment of how qualified it is to make a particular classification. Good introspection means a strong correlation between confidence and correctness.

In order to apply this principle to a real classifier, we can derive the empirical distribution function using a classifier's measurements from a labelled test set, and calculate T from a pre-determined ϵ by (4.3). However, how can we say that this bound will not be violated during future testing? Crucially, this relies on the classifier being able to generalise consistently over all future data. In the next chapter we propose that non-stationarity between training and testing is commonplace, and that this consistency over unseen data is a vital component of introspection.

We therefore seek a classifier which:

- Makes few errors overall (scoring highly by conventional metrics),
- Makes errors with high uncertainty, and is correct when confident, and where
- These properties are consistent despite surprising and unseen test data.

After summarising the literature on classification uncertainty in decision making, we expand our example in Figure 4.2 by introducing a number of idealised classifiers with varying degrees of introspective ability. These will be used in subsequent chapters for comparison against real classification frameworks.

4.2 Related Works

Using classifiers as a source of information for making decisions is extremely common. Successful applications are as diverse as the detection of ground traversability

(e.g. Thrun et al. [2006]), the detection of lanes for autonomous driving (e.g. Huang and Teller [2010]), the consideration of classifier output to guide trajectory planning and exploration (see, for example, Meger et al. [2008], Velez et al. [2011]). These works typically use classification output on a model-trust basis; systems are optimised with respect to precision and recall, and egregious misclassifications are accepted as par for the course. However, the suitability of the classification framework employed with respect to its introspective capacity has not previously been considered in robotics. Thus, we consider motivating, defining, and investigating introspection in a robotics context to be the primary contribution of our work.

In the related field of reinforcement learning, the authors of Li et al. [2008] present a general framework which determines whether enough labelled data have been provided to constrain certain problems. If the learner's space of solutions is insufficiently constrained such that its output cannot be guaranteed to be within ϵ of the true solution with probability $1 - \delta$, it asks for more labelled data. This decision considers both false positive and false negative errors equally, and thus the framework is not appropriate for situations in which costs associated with those errors are unbalanced. In the context of autonomous systems, the costs are commonly unbalanced.

Sayedi et al. [2010] extend Li et al. [2008]'s paper by considering the trade-off between allowing no mistakes (as Li et al. [2008] do) and frequently saying *don't know*, and allowing mistakes but never saying *don't know*. The authors present an algorithm which minimises the number of *don't know* responses for a user-set tolerance on the number of errors. This is very similar to setting a value for ϵ, as we propose, except that we are defining the tolerance of the number of type I errors (false positives) and allowing the system to choose the optimal threshold which minimises the number of type II errors (false negatives). An ϵ value of 0 would be analogous to the original paper.

Rather than using classifier uncertainty as a measure of correctness, the computer vision community have used a separate learner, mapping input to classifier reliability, alongside a standard detector [Zhang et al., 2014]. This concept has been applied to quality estimation of fingerprints [Tabassi et al., 2004, Grother and Tabassi, 2007] (where a neural network was used) and face detection [Abhishek et al., 2015], where image quality features such as face pose and lighting are used as input. In a sense, this out-sources the introspection requirement to another learner. But will that separate learner itself be introspective? We suggest that while this may improve decision making, it is still valuable to evaluate the confidences offered by our detectors.

Boshra and Bhanu [2000] also use a secondary learner to estimate performance, but also seek to answer *why* prediction might be poor, which could be a result of noise, similarity between classes, and distance between training and test examples. We share this opinion that distance is important in estimating performance.

Marsland et al. [2005] use a neural network to detect unseen test data in a *novelty detection* system. Their network uses habituation synapses, which decrease in strength as their end nodes fire. In this way, they desensitise the network to the training data.

In PAC learning [Valiant, 1984, Angluin, 1988] we try to learn, with high probability, a hypothesis which is a good approximation of the target hypothesis. Hence, *probably approximately correct*. This can be used to determine the number of training data to guarantee the predetermined error bounds for a learner. However, the bounds will be violated if the data set is non-stationary after training.

Torralba and Efros [2011] and Khosla et al. [2012] discuss the issue of *data set bias*, recognising the non-stationarity between labelled data sets commonly used for benchmarking perception systems and the real world in which our robots operate. We share the opinion that this is detrimental to the subsequent performance of our

robots.

Introspection is closely related to *calibration* [Dawid, 1982, DeGroot and Fienberg, 1983], which is the degree to which a forecaster's probabilistic predictions hold true in the limit (the original context of calibration is weather forecasting). That is, if of "those events to which [the forecaster] assigns a probability 30 percent, the long-run proportion that actually occurs turns out to be 30 percent" [Dawid, 1982]. This is used in Platt [1999], where the *empirical calibration curve* is plotted for a hold-out data set, and each classifier is compared to the diagonal, indicating good calibration. However, this analysis ignores the question of whether there is stationarity between training and test sets, which is a clear case when a classifier will go out of calibration. Calibration in theory should lead to good introspection, but in practice classifiers are calibrated by learning a constant mapping from their output to more appropriate probabilities given some hold-out set. This implicitly assumes stationarity between the training data and future test data, an assumption we argue is dangerous in robotics.

The density distributions $f_i(z)$ can be thought of as second order probabilities [Pearl, 1987]. These were developed to distinguish, "between uncertainty about truths from uncertainty about probabilistic assessments" [ibid], which is exactly what we are evaluating in this thesis. We use these second order probabilities to indicate whether the probabilistic assessments are likely to lead to truths, and to characterise them when they do not.

4.3 Idealised Classifiers

In Section 4.1 we introduced the concept of *introspection*, which is a classifier's capacity to associate, with any classification, a degree of confidence which correlates to the likelihood that it is wrong. We used the illustration of the 'top hat' classifier

from Figure 4.2, which draws a very clear boundary in uncertainty between classifications which are correct, and those which could be erroneous. In this section we design a number of idealised classifiers, similar to the top hat classifier, with varying levels of introspective capacity. Despite their varying levels of introspection, they will all have the same overall accuracy. Thus, their differences highlight the importance of appropriate confidence levels rather than performance according to traditional metrics such as F-measure. They will then be used as benchmarks for real classifiers used commonly in robotics.

By using idealised classifiers we are free to explore the effects of changing the error functions (the probability of error given some value of z, or $p(e \mid z)$) without exposure to confounding factors such as data set peculiarities or classifier implementation differences. In choosing the density functions $f_1(z)$ and $f_2(z)$ we can generate measurements from a classifier which are consistent in a way that real classifiers are not. However, the conclusions we eventually draw will result solely from the subtle differences between the error functions, and thus allow us to examine the importance of where errors are made along in the range of z. We are not primarily concerned with the total frequency of errors, but with where they occur in z.

There are infinitely many possible idealised classifiers, so we choose to restrict ourselves to those which adhere to some reasonable expectations of real-world classifiers. We have motivated the *top hat* classifier from Figure 4.2 as a desirable optimum (given some level of total error) and show it in Figure 4.3. Next, we consider a classifier which has no introspective quality because there is zero correlation between confidence and correctness. It is shown in continuous blue and is called the *uniform* error function, because the probability of error is uniform across the range of z.

Intuitively we expect real classifiers to lie somewhere between the *top hat* and *uniform* error functions. We expect that there is some positive correlation between

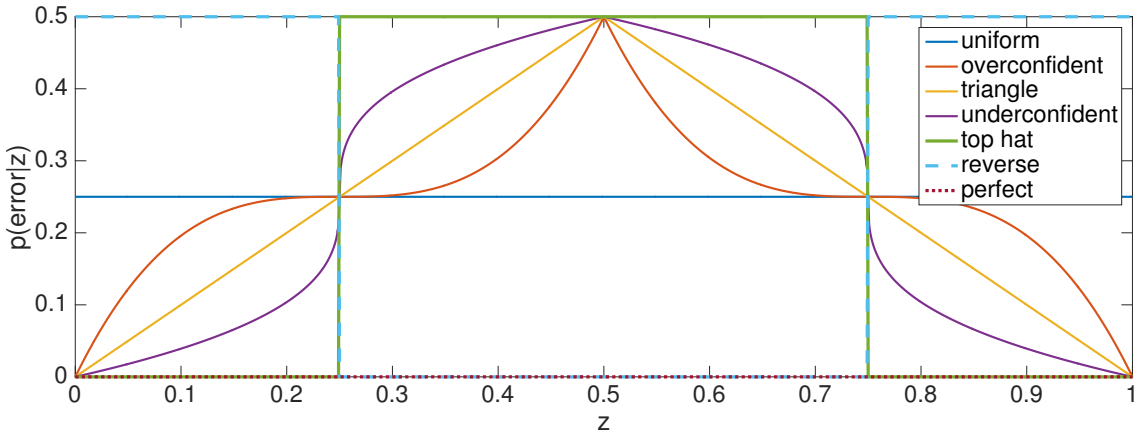


Figure 4.3: Here we see the likelihood of making an error given some probability measurement z (or *error function*) for various idealised classifiers given by Equations (C.1) to (C.7). We choose these examples of classifiers because the first four increasingly make their mistakes around $z = 0.5$, making them more introspective. The error function for the *top hat* classifier is the extreme case of making zero mistakes in the range $0 \leq z < 0.25$ and $0.75 < z \leq 1$, and the error function for the *reverse* classifier is the exact opposite. Note that the *perfect* classifier makes no mistakes, so $p(e \mid z) = 0 \; \forall z$, and is used as a baseline. (Best viewed in colour.)

correctness and confidence. Therefore, we define three error functions which lie between the two. These are called the *overconfident*, *triangle*, and *under-confident* error functions, and are shown in red, yellow, and purple, respectively. The *triangle* error function is perhaps the most intuitive, with a gradual increase in error rate as the measurement becomes more uncertain. However, we will show that it does not perform as well as the *top hat* classifier. The *overconfident* error function is closest to the uninformative *uniform* error function, and the *under-confident* error function is closest to the *top hat* error function. These shapes are chosen such that the area underneath them is the same, which helps keep the total error rate $p(e)$ the same for each. The calculation of the error rate is discussed in detail later in Section 4.3.1.

Lastly, we show two more interesting error functions, included as reference points to contextualise the others. The first is the *reverse* error function, which negatively correlates confidence with correctness, and is thus the reverse profile of the *top hat* classifier. We would expect this classifier to perform very badly in classical decision-

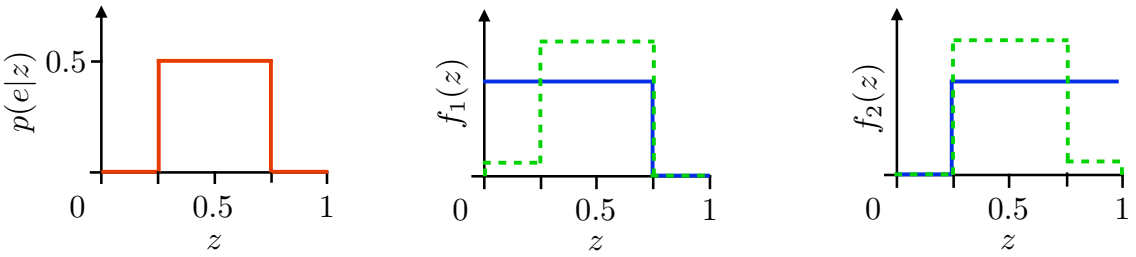


Figure 4.4: We demonstrate that the red error function on the left is not sufficient to define a classifier: either of the blue or the dashed green $f_1(z)$, $f_2(z)$ pairs will generate the red *top hat* error function. However, the blue densities define a far superior classifier to the dashed green, because less of their mass is within the uncertain and erroneous central region $0.25 < z < 0.75$, and much more of the mass is within the confident and correct regions $z < 0.25$ and $z > 0.75$.

making frameworks, because it will make errors with extremely high confidence. The second error function is the *perfect* classifier, which makes no mistakes. It does not correlate confidence with correctness, but it makes no errors and so regardless of the chosen threshold T, it will always indicate the correct decision to make.

We have proposed some idealised classifiers in terms of where they make errors along the range of Z. However, an error function itself is not sufficient to define a classifier, because for any given error function there are an infinite number of pairs of densities $p(z \mid C_i)$ which can generate it. An example is shown in Figure 4.4, where the red top-hat error function can be generated from either the blue or the dashed green pairs of density functions. The classifier defined by the dashed green density functions is far inferior to the blue, because much more of the masses of each density function lie within the central, disputed region, and much less are within the confident and correct surrounding regions. The classifier defined by the blue densities will have a lower value for $p(e)$ than the classifier defined by the dashed green densities.

In the next section we present full definitions of the idealised classifiers by deriving a method of mapping each error function to a pair of density functions, subject to a few assumptions.

4.3.1 Determining the Density Functions

We have chosen a number of error functions to describe some idealised classifiers, but in order to define them fully we need the probability density functions $f_1(z)$ and $f_2(z)$. Having shown that there is more than one pair of density functions which generate a given error function (see Figure 4.4), here we describe the process of choosing a pair while maintaining the same $p(e)$ for each classifier. Thus, each classifier will have the same accuracy according to the conventional metrics of classification, but they will behave differently in terms of their classification uncertainties. This is to demonstrate that the standard metrics of classification are insufficient to capture important differences in classifier behaviour.

First, let

$$E(z) = p(e \mid z), \tag{4.4}$$

$$p_1 = p(c = C_1), \text{ and} \tag{4.5}$$

$$p_2 = p(c = C_2), \tag{4.6}$$

and recall that T is a threshold on z, below which the decision-making process chooses action a_1, and above which it chooses a_2. By the definition of conditional probability,

$$E(z) = \frac{p(e, z)}{f(z)}, \tag{4.7}$$

where

$$p(e, z) = p_1 p(e, z \mid c = C_1) + p_2 p(e, z \mid c = C_2) \tag{4.8}$$

$$= p_1 f_1(z) I(z > T) + p_2 f_2(z) I(z < T), \tag{4.9}$$

I is the indicator function, and

$$f(z) = p_1 f_1(z) + p_2 f_2(z). \tag{4.10}$$

Therefore,

$$E(z) = \begin{cases} \dfrac{p_2 f_2(z)}{p_1 f_1(z) + p_2 f_2(z)}, & \text{if } z < T \\ \\ \dfrac{p_1 f_1(z)}{p_1 f_1(z) + p_2 f_2(z)}, & \text{if } z > T. \end{cases} \tag{4.11}$$

First let us derive a general solution for $p_1 f_1$ and $p_2 f_2$, and then we will motivate a particular case for further study. By (4.10) and (4.11),

$$p_1 f_1(z) = \begin{cases} f(z)(1 - E(z)), & \text{if } z < T \\ f(z)E(z), & \text{if } z > T \end{cases} \tag{4.12}$$

$$p_2 f_2(z) = \begin{cases} f(z)E(z), & \text{if } z < T \\ f(z)(1 - E(z)), & \text{if } z > T. \end{cases} \tag{4.13}$$

By making some assumptions, we can use (4.12) and (4.13) to derive classifiers, that is, pairs of $f_1(z)$ and $f_2(z)$, given the error functions $E(z)$.

Assumptions

Thus far we have made no assumptions about p_1, p_2, $f(z)$, $E(z)$, or T, aside from their inherent nature as probabilities, probability density functions, or conditional probability functions. These are free to be chosen, although there are consistency conditions between them.

We wish to compare various predetermined choices of $E(z)$. We have stated our desire for z to be interpreted probabilistically with maximum uncertainty at $z = 0.5$, so the first assumption is that all of our error functions are symmetrical about this

value of z (like commonly-used measures of uncertainty, see Section 5.2 for more details). The objective is to find a pair of probability density functions $f_1(z)$ and $f_2(z)$ that generate each of the error functions with a predetermined total error rate.

By the same reasoning, the location of maximum uncertainty indicates the boundary between favouring one class over the other. It is therefore natural to set $T = 0.5$, which is our second assumption.

Choosing these idealised classifiers to be without class bias, we assume that $p_1 = p_2 = 0.5$. We believe this is reasonable in the absence of problem-specific information.

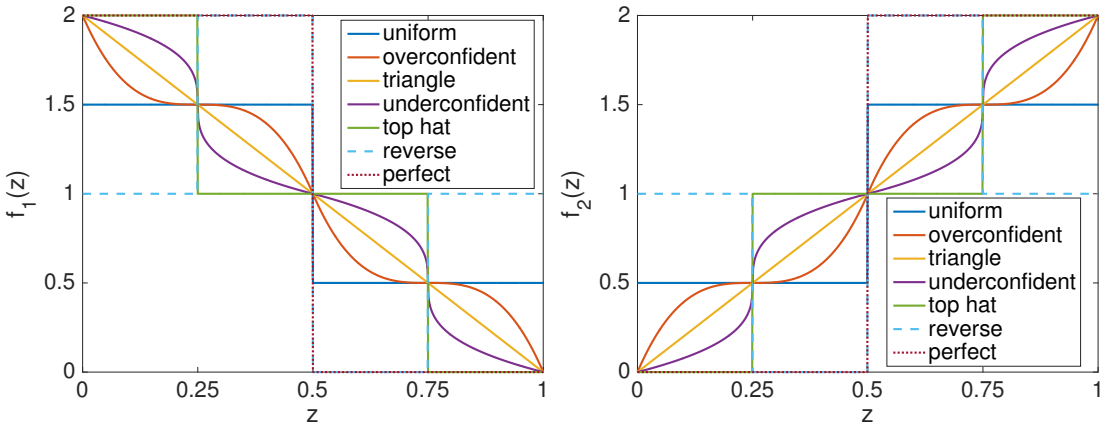
Lastly, we must choose a constraint on $f(z)$. There is merit in choosing exemplar density functions $f_1(z)$ and $f_2(z)$ which are simple to interpret. This is achieved if we allow fair representation across z and determine $f(z)$ to be the uniform density function, therefore

$$f(z) = 1. \tag{4.14}$$

This implies that no value of z is more likely to occur than any other value. While this is a bold assumption, we value simplicity and interpretability, and it will not lead to a loss of generality. The benefit of this assumption is that the expected error rate, defined as

$$p(e) = \int_0^1 p(e \mid z) f(z) dz \tag{4.15}$$

becomes equal to the area under the error functions. Analytically we can show that the integral of all error functions is equal to 0.25 (with the exception of the *perfect* error function, for which error rate is 0), meaning that they are expected to make an error in one of every four measurements, and will perform equivalently in terms of overall accuracy. This accuracy is chosen owing to the desire for the *triangle* error function together with the uniform density in (4.14).



(a) $f(z \mid c = C_1)$ (b) $f(z \mid c = C_2)$

Figure 4.5: Here we show some probability density functions that generate Figure 4.3. We draw samples from these distributions to simulate the classifiers' behaviours. Note the similarity between the densities for the *perfect* classifier and those in Figure 4.2. (Best viewed in colour.)

By the above assumptions, together with equations (4.12) and (4.13),

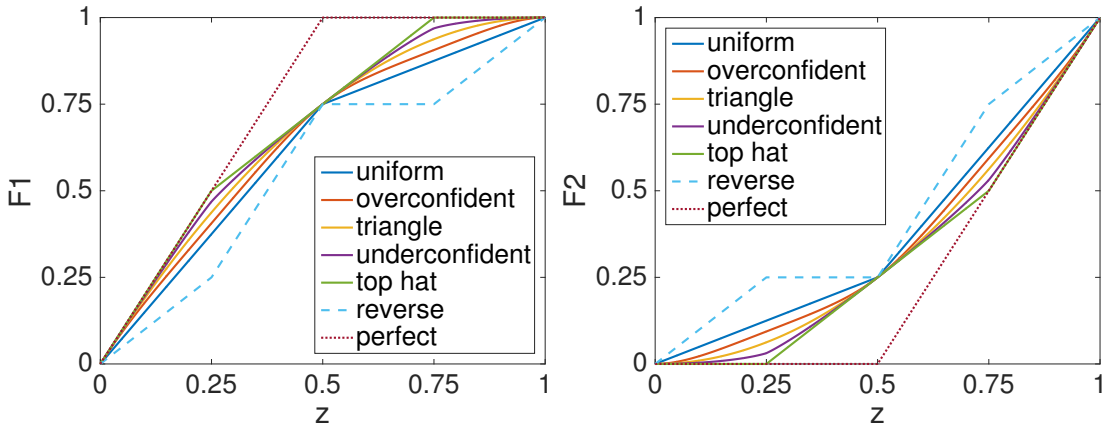
$$f_1(z) = \begin{cases} 2(1 - E(z)), & \text{if } z < T \\ 2E(z), & \text{if } z > T \end{cases} \tag{4.16}$$

$$f_2(z) = \begin{cases} 2E(z), & \text{if } z < T \\ 2(1 - E(z)), & \text{if } z > T. \end{cases} \tag{4.17}$$

We calculate the density functions $f_1(z)$ and $f_2(z)$ for each error function $E(z)$ and show them in Figure 4.5. These probability density functions $f_1(z)$ and $f_2(z)$ are sufficient to define the idealised classifiers.

Upon examining the density functions, we see that for the more introspective classifiers the mean of $f_i(z)$ is closer to the extremes, i.e. nearer to 0 for $f_1(z)$ and nearer to 1 for $f_2(z)$. This is a consequence of correlating correctness with confidence.

If we do not assume that $p_1 = p_2$ or that $f(z) = 1$, the densities would be skewed to one side, and the error rates would be different for the error functions we have



(a) (b)

Figure 4.6: Here we show the cumulative distribution functions from Figure 4.5. Note the clear ordering of the functions in (a), which is reversed in (b). (Best viewed in colour.)

specified.

Cumulative distribution functions

Knowledge of the cumulative distribution functions (shown in Figure 4.6) is useful for the process of sampling measurements z from a classifier given a class label i. Given a particular density function $f_i(z)$ where $i \in \{1, 2\}$ we calculate the cumulative distribution function $F_i(x) = p(Z < x)$ and use it to perform inverse transform sampling to generate a measurement, given by

$$z = F_i^{-1}(u), \tag{4.18}$$

where u is a random number generated from the uniform distribution on $[0, 1]$ [Devroye, 1986].

4.4 Summary

In this chapter we have discussed the ideal behaviour of a classifier, and motivated the importance of correlating correctness with confidence, a quality which we term *introspection*. We have alluded to the fact that this quality must be consistent for future test data, an idea which we develop in the following chapter. We have derived a number of idealised classifiers, which we can later use to benchmark a number of commonly-used real classification frameworks used in robotics. The idealisations allow us to simulate decision-making scenarios and to measure the effects of varying the introspective capacity, without results being affected by overall error rate or data set peculiarities.

In the next chapter we introduce those real classification frameworks, and examine their introspective capabilities in simple scenarios. In Chapter 6 we choose two different decision-making scenarios and compare the quality of the decisions made by the real classifiers by comparing them against the idealisations.

Chapter 5

Introspection in Practice

In Chapters 2 and 4 we proposed a desired classifier behaviour in order for a robot to make appropriate decisions: to correlate confidence with correctness. In this chapter we examine a number of commonly-used classification frameworks and predict, based on their internal methodology, which are likely to possess this introspective trait. We do so specifically by examining their treatment of *distance* between points in feature space. We then apply them to situations in which they are destined to give a wrong answer, and examine their uncertainties. Those exhibiting higher uncertainties indicate stronger correlations between incorrectness and uncertainty, indicating an introspective capacity.

Intuitively, a test datum which is far away in feature space from the training data should be more likely to be misclassified than one which lies in the middle of a dense cluster of one class. Thus, an introspective classifier should exhibit greater uncertainty for data which are far away from the training data. Most classification frameworks use a model to represent the training data, such that instead of comparing the test datum to all the training data, the model is used to generate a measure of similarity between the two. The nature of the model and the measure of distance determine the uncertainty with which a classification is made. Some

frameworks consider one single discriminant to separate the classes, while others consider a variety of possible discriminants and combine their responses. Often this is done 'under the hood' of the classification algorithm. The results we present in this chapter indicate that the latter tends to be characteristic of classifiers with a better sense of introspection, as a result of their ability to consider the variance in the numerous responses to reflect uncertainty.

After outlining the notation used in this chapter, we motivate the use of normalised entropy as a measure of uncertainty, which is used henceforth. Then we give an intuition as to why multiple-discriminant classifiers are likely to exhibit more introspective behaviour than single-discriminant classifiers in Section 5.3. In Section 5.4 we examine the literature on the use of distance and the number of discriminants in classification.

In Section 5.5 we introduce a number of commonly-used classification frameworks, specifically examining their treatment of distance, and making predictions about their introspective capacities.

Next, in Section 5.6 we test these real classifiers on what we call the 'third class experiment', in which classifiers are trained on two classes and tested on a third, unseen class. We expect introspective classifiers to respond with high uncertainty. We do this with both the synthetic and real data sets described in Chapter 3. We argue that third class examples occur commonly in scenarios where our robots are expected to function for long periods of time in extremely varied worlds, such as autonomous driving in urban environments. This is due to non-stationarity between the training and testing data.

We move away from the explicit third class experiment in Section 5.7, examining the uncertainties of the classifiers in scenarios where they must find a distinct, positive class against a broad background class that contains all other classes. In this setting we see rather different behaviours to the third class experiments. We

conclude with a discussion of why the behaviour changes in Section 5.8.

5.1 Notation

Let a classifier map an input $\mathbf{x} \in \mathbb{R}^d$ to one of a set of classes $\mathcal{C} = \{C_1, \ldots, C_c\}$ via an associated label $y \in \{1, \ldots, c\}$, where c is the number of classes and d is the dimensionality of the data. The training data set is defined by $\{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denotes the set of N feature vectors and $\mathcal{Y}$ denotes the set of corresponding class labels. Test data and labels are denoted by the use of a star, for instance $\{\mathbf{x}_*, y_*\}$.

During training, a classification framework may converge, given the training data, on a particular model m from a family of possible models $\mathcal{M}$. The space of $\mathcal{M}$ is called the *hypothesis space*, of which the *version space* is the subset which correctly classifies the training data.

5.2 Measures of Uncertainty

In this thesis we frequently refer to uncertainty, particularly for its relationship with a classifier's correctness. Specifically, we look for a relationship between both confidence with correctness, and lack of confidence with incorrectness. Most binary classification frameworks can be calibrated to take a test datum and return a 'probability', generally with little justification for being called as such, except for the fact that it lies within the range $[0, 1]$, and the extrema represent the two possible classes. The values between allow the framework to express confusion between the two. If we consider a mapping from probability to this confusion or uncertainty, there are some obvious constraints: the values for $p(C_i) = 0$ and $p(C_i) = 1$ denote the least uncertain positions on the scale, and in the absence of further information, $p(C_i) = 0.5$

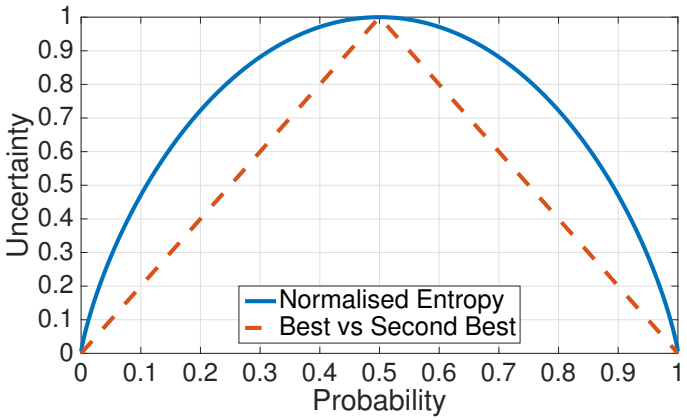


Figure 5.1: Normalised entropy and best-versus-second-best as measures of uncertainty in the binary classification case.

denotes the highest uncertainty. With these constraints, there are infinitely many possible mappings between probability and uncertainty. The literature promulgates these two metrics: normalised entropy (NE), and best-versus-second-best (BvSB) [Joshi et al., 2009].

Normalised entropy H_N is defined as

$$H_N(\mathbf{x}) \quad = \quad -\sum_{i=1}^{c} p(y = C_i \,|\, \mathbf{x}) \log_c [p(y = C_i \,|\, \mathbf{x})]. \tag{5.1}$$

This is equivalent to the Shannon entropy measure normalised by its maximum, which is the entropy of the c-dimensional uniform distribution, $\log(c)$. The result is a measure ranging between 0 and 1 where a *higher* value indicates *greater* uncertainty in the classifier's belief, as shown by the blue curve in Figure 5.1.

An alternative uncertainty measure proposed in the active learning literature is the best-versus-second-best heuristic [Joshi et al., 2009] which equals 1 minus the difference between the largest and the second largest class likelihood estimates, as shown by the red curve in Figure 5.1 for the binary case. This measure attempts to characterise the reliability of the maximum likelihood estimate rather than encoding the shape of the full distribution over class labels. BvSB is used by Joshi et al.

[2012] to obtain a measure of uncertainty which is unbiased by large numbers of unlikely classes. The BvSB and normalised entropy measures are closely related in the binary-classification setting, which is that of this thesis.

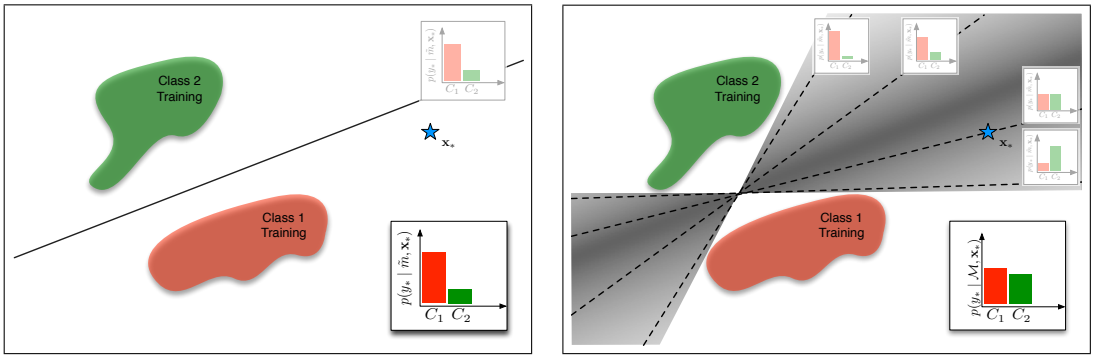
We use normalised entropy throughout this thesis for two reasons: firstly, it is formed by an information-theoretic argument, compared to BvSB which is an ad-hoc heuristic; secondly, in multi-class settings it considers the entire distribution over classes, rather than BvSB which only takes into account only the two classes with the highest probability.

5.3 A Distance-Based View on Introspection

Introspection concerns not the final class decision but rather the confidence with which this decision is made. The concept is motivated by the desire to take appropriate action when a classifier indicates high uncertainty. Prior to training, domain-specific knowledge is often used to constrain the family of classification models employed (for example in the form of a kernel or a type of base classifier). Classifier training then involves learning a set of (hyper-) parameters given a training dataset $\{\mathcal{X}, \mathcal{Y}\}$. The training data implicitly give rise to a probability distribution over the set of all possible models (or *discriminants*) within the chosen family $\mathcal{M}$, such that

$$\{\mathcal{X}, \mathcal{Y}\} \rightarrow p(m \,|\, \mathcal{X}, \mathcal{Y}) \,, \quad m \in \mathcal{M}. \tag{5.2}$$

With a slight abuse of notation, m here denotes any member of the family of possible models, $\mathcal{M}$. In the following we make this relationship explicit by conditioning on both a model (or family of models) as well as on a test datum $\mathbf{x}_*$. Typically, training leads to the selection of a *single* model, $\tilde{m}$ from $\mathcal{M}$ such that a prediction y_* for a



(a) (b)

Figure 5.2: An illustration of the two types of classification frameworks considered: (a) during training a *single* model is selected to classify an unknown datum $\mathbf{x}_*$ some way removed from the training data; (b) training leads to a distribution over models which is considered entirely to arrive at the final prediction. This illustration is for the family of linear models (indicated by solid (a) and dashed (b) lines). Each predictor is further annotated with its individual prediction for a point at the blue star. The overall predictive distribution is shown in the bottom right of each subplot. The shading in part (b) indicates the probability weights associated with individual models. Darker regions contain more weight. Note that the overall predictive distribution in (a) stems from the single model used and is, in this case, inappropriately confident. In part (b), however, the overall predictive distribution is moderated by computing the expectation over all models. This gives rise to a much more appropriate uncertainty estimate — the introspective quality we seek. (Best viewed in colour.)

new, unseen feature vector $\mathbf{x}_*$ is obtained by approximating

$$p(y_* \,|\, \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* \,|\, \tilde{m}, \mathbf{x}_*)\,, \quad \tilde{m} \in \mathcal{M}. \tag{5.3}$$

This is illustrated in Figure 5.2a. Common examples of this type of classification framework include SVMs and Boosting classifiers, where an optimisation is performed to select the single best model given the training data (see Section 5.5). The *iid* assumption here is inherent since it is assumed that $\tilde{m}$ remains the best model for all predictions of unseen data. Breaking this assumption therefore often renders the chosen model suboptimal. We call these methods *single-discriminant*. A real example of this is shown by the linear SVM in Figure B.1e.

An alternative to the single model approach are classification frameworks which take into account the *entire set* of possible models in the specified family conditioned on the training data, such that

$$p(y_* \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* \mid \mathcal{M}, \mathbf{x}_*). \tag{5.4}$$

This case is illustrated in Figure 5.2b. Here the shading indicates the distribution $p(m|\mathcal{X}, \mathcal{Y})$ with darker shades indicating increased probability. To aid intuition, predictions of four randomly selected members of $\mathcal{M}$ are also illustrated. Final predictions are made by taking into account opinions from all members of $\mathcal{M}$, often via the computation of an expectation such as for a Gaussian Process Classifier (GPC, see Section 5.5.1). Crucially, when considering an expectation over all of $\mathcal{M}$, the increased variance in feasible (and therefore likely) models at a distance from the training data leads to a moderation of the class predictions. The linear GPC in Figure B.1c demonstrates this.

Between the two extremes lies the random forest, which chooses a number of differing samples from $\mathcal{M}$, and averages over their responses. We call classifiers which consider more than one model *multi-discriminant*.

We believe that this marginalisation over plausible models in version space is a key component of an introspective classifier. A great variation in test response between models indicates that, given the training data, there are multiple valid interpretations, and thus that the uncertainty should be high. A single-discriminant classifier is incapable of such moderation.

5.4 Related Works

The concept of introspection as introduced here is closely related to considerations in active learning, where uncertainty estimates and model selection steps are often

employed to guide data selection and gathering for an incremental learning algorithm. Kapoor et al. [2010], for example, present an active learning framework for object categorisation using a GPC where classifications with large uncertainty (as judged by posterior variance) lead to a query for a ground-truth label and are subsequently used to improve classification performance. Joshi et al. [2009] address multi-class image classification using SVMs and propose criteria based on entropy and best-versus-second-best measures (see Section 5.2) for disambiguating uncertain classifications. Holub et al. [2008] propose an information-theoretic criterion that maximises expected information gain with respect to the entire pool of unlabelled data available. Hospedales et al. [2013] discuss optimising rare class discovery and classification using a combination of generative and discriminative classifiers.

Our treatment of introspection is further informed by an ongoing discussion in the machine learning community regarding how best to account for variance in the space of feasible classifier models when training on, essentially, an incomplete set of data. For example, Tong and Koller [2002] present an incremental algorithm for text classification using SVMs and the notion of a *version space*, the set of consistent hyperplanes separating the data in a feature space induced by the kernel function. Zhang et al. [2012] introduce an SVM formulation which weights each training datum by how uncertain it is, based on how close it is to the opposite class. Distance is measured orthogonal to a hyperplane. A global classifier then incorporates these uncertainties as margin constraints, yielding a classifier that places less confusing instances farther away from the global decision boundary. We share the intuition that accounting for variance in version space when selecting a model leads to an increased introspective capacity. As a secondary contribution, therefore, our results serve to further corroborate this intuition.

Niculescu-Mizil and Caruana [2005] recognise that the question of whether the probabilities produced by various classification frameworks are appropriate is impor-

tant, a sentiment we clearly share. They conclude that poorly-calibrated frameworks (in a probabilistic sense) can be effectively corrected using an additional learned calibration using either Platt's method or isotonic regression. They find random forests to perform well pre-calibration (although they did exhibit a tendency to be under confident, consistently with our findings), and that SVMs perform well post-calibration. They associate the need for further calibration specifically to the classifiers using max-margin optimisation, rather than the treatment of distances in feature space and the distribution of models over version space as we do. They also do not explore the effects of making decisions using these probabilities.

Berczi et al. [2015] have confirmed the introspective power of multi-discriminant GPCs over single-discriminant SVMs, employing them to avoid areas of terrain for which the height may be misclassified.

5.5 Commonly-Used Classification Frameworks

In this section we examine the methods by which a number of classification frameworks derive probabilities (and therefore uncertainties), and then make predictions regarding the introspective properties of each. In particular, we examine their treatment of distance to determine those probabilities. We consider and compare three types of Gaussian Process classifiers (linear and non-linear kernels, and the sparse non-linear IVM), two Support Vector Machines (SVMs, with linear and non-linear kernels), the boosted classifier LogitBoost, and random forests.

We consider these classification frameworks because they represent some key ideas within the hundreds of available classification algorithms available. While one classifier may outperform another given a specific application, there is no 'silver bullet' when it comes to classification. As Wolpert and Macready [1997] show, "any two algorithms are equivalent when their performance is averaged across all possible

problems" [Wolpert and Macready, 2005]. Therefore, one classification framework may be superior for one subset of problems, but there is no guarantee that it will be superior for another. This is why it is important to consider multiple frameworks over multiple problems. Of course, in this thesis we restrict ourselves to a particular subset of problems, but nevertheless we are looking for consistency in behaviour across these problems rather than superiority in one or two.

We note that we ran tests with the K-nearest-neighbour (KNN) classifier, but found that it did not perform well in terms of classical metrics like precision, recall, and F-measure. As a result, it is not a commensurate alternative to the classifiers considered in this thesis, and so has not been included in our analysis.

5.5.1 Gaussian Processes Classification

Binary classification using a Gaussian Process (GP) [Williams and Barber, 1998, Rasmussen and Williams, 2006] is formulated by first introducing a *latent* function $f(\mathbf{x})$ and then applying a sigmoid function Φ (similar to the sigmoid described later in Section 5.5.2, except that the predictive variance of the GP is used as well as the predictive mean) to obtain the prediction $p(y_* = C_1 \,|\, \mathbf{x}_*) = \Phi(f(\mathbf{x}_*))$. A GP prior is placed on the latent function $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ characterised by a *mean* function $\mu(\mathbf{x})$ and a *covariance* (or kernel) function $k(\mathbf{x}, \mathbf{x}')$. GPC training establishes values for the hyper-parameters specifying the kernel function k by maximising the log marginal likelihood of the training data.

Probabilistic predictions for a test point are obtained in two steps. First, the distribution over the latent variable corresponding to the test input is obtained using

$$p(f_* \,|\, \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int p(f_* \,|\, \mathcal{X}, \mathbf{x}_*, f) p(f \,|\, \mathcal{X}, \mathcal{Y}) df, \tag{5.5}$$

where $p(f \,|\, \mathcal{X}, \mathcal{Y}) = p(\mathcal{Y} \,|\, f) p(f \,|\, \mathcal{X}) / p(\mathcal{Y} \,|\, \mathcal{X})$ is the posterior distribution over latent

variables. This is followed by *marginalising* over the latent f_* to yield the class likelihood $p(y_* = C_1 \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$ as

$$p(y_* = C_1 \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_* \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) df_*. \tag{5.6}$$

Exact inference is analytically intractable due to the sigmoid likelihood function. Instead, we leverage expectation propagation (EP) [Minka, 2001], a method widely used for this purpose.

The GPC framework offers two key benefits over the other approaches discussed here [Rasmussen and Williams, 2006]. Firstly, the classification output has a clear probabilistic interpretation as it directly results in the class likelihood. In contrast, neither the SVM nor the Boosting framework provide an inherently probabilistic output in the Bayesian sense, but instead estimate a suitable calibration. Secondly, and crucially, the GP formulation addresses uncertainty or *predictive variance* in the latent function $f(\mathbf{x})$ via *marginalisation* (or averaging) over all models induced by the training set resulting in the estimate $p(y_* = C_1 \mid \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$ from Equation (5.6). This process also gives rise to the well known property of increased variance while far away from the data in GP regression (at least with the squared-exponential kernel and many others). Again this is in contrast to the SVM or Boosting estimate $p(y = C_i \mid \hat{f}, \mathbf{x}_*)$ that rely on a single discriminant estimate $\hat{f} : \mathcal{X} \to \mathcal{Y}$ learnt via minimisation. In the context of introspection, the ability to account for predictive variance is a key advantage of Bayesian classification approaches.

GPCs are used in many domains, including terrain classification [Paul et al., 2012], failure detection [Plagemann et al., 2007], and terrain traversability [Berczi et al., 2015]. Kulick et al. [2013] use GPCs to classify geometric relations between objects, or 'symbols', for use in an active learning scenario where a robot learns these symbols via interactions with a human. In most of these works the authors state

that their choice of classifier is based upon the information-theoretic probabilities that it produces.

Throughout this work we use the GPML toolbox [Rasmussen and Nickisch, 2010] for GPC training and testing. In this implementation, the hyper-parameters are optimised by minimisation using conjugate gradients. This method chooses a single value for each hyper-parameter, and is prone to finding a local rather than a global minimum. Instead of choosing a single value for each hyper-parameter as we do, it is possible to instead consider a distribution over it (a *hyper-prior*) [Rasmussen and Williams, 2006]. Due to the potential intractability of considering the entire hyper-prior, it is often approximated using methods such as Markov chain Monte-Carlo (MCMC). More commonly, a cruder Laplace approximation is used. It has been found that in terms of traditional metrics, choosing a single value often gives commensurate performance to a more thorough treatment [MacKay, 1999]. However, it is possible that by effectively considering the aggregate responses of multiple, different GPCs, an even more informative *predictive variance* can be calculated. The quality of this predictive variance is what we consider to give an introspective advantage to the GPC.

The Informative Vector Machine

A key drawback of a GPC is its significant computational demand in terms of memory and run time during training and testing, more than any of the other frameworks considered here. This is due to the fact that the GP maintains a mean $\boldsymbol{\mu}$, as well as a covariance matrix Σ, which is computed from a kernel function and is of size $N \times N$. A number of sparsification methods have been proposed in order to mitigate this computational burden. For efficiency, in this work we adopt one such sparsification method: the Informative Vector Machine (IVM) [Lawrence et al., 2002].

The IVM has been used primarily in the signal processing domain. Elattar [2013] using it to predict short-term fluctuations in the price of electricity. Lu et al. [2012] wish to optimise the locations of mobile sensors to measure some spatial function, and use an IVM to inform the placement of the nodes based on spacial uncertainty estimates. It has also been used in image classification, where Bazi and Melgani [2010] use the IVM for classification of ground types in hyper-spectral images.

The main idea of this algorithm is to only use a subset of the training points denoted the *active set*, $\mathcal{I}$, from which an approximation $q(f \mid X, \mathbf{y}) = \mathcal{N}(f \mid \boldsymbol{\mu}, \Sigma)$ of the posterior distribution $p(f \mid X, \mathbf{y})$ is computed. The IVM algorithm computes $\boldsymbol{\mu}$ and Σ incrementally, and at every iteration j selects the training point $(\mathbf{x}_k, y_k)$ which maximises the entropy difference ΔH_{jk} between q_{j-1} and q_j for inclusion into the active set. As q is Gaussian, ΔH_{jk} can be computed by

$$\Delta H_{jk} = -\frac{1}{2} \log |\Sigma_{jk}| + \frac{1}{2} \log |\Sigma_{j-1}|. \tag{5.7}$$

We use an efficient form of this, the details of which can be found in Lawrence et al. [2005]. The algorithm stops when the active set has reached a desired size. We choose this size to be a fixed fraction q of the training set, which we set to be 0.8. Throughout this work we use the IVM MATLAB toolbox [Lawrence, 2012] for both training and testing.

To find the kernel hyper-parameters $\boldsymbol{\theta}$ of an IVM, two steps are processed in a loop for a given number of times: estimation of $\mathcal{I}$ from $\boldsymbol{\theta}$ and minimising the marginal likelihood $q(\mathbf{y} \mid X)$, thereby keeping $\mathcal{I}$ fixed. Although there are no convergence guarantees, in practice a small number of iterations is sufficient to find good kernel hyper-parameters.

Importantly for our work, since inference with the IVM is similar to that with a GPC, the IVM retains the model averaging described in (5.6). We argue, therefore,

that the IVM provides a significant and well-established improvement in processing speed over a GPC without affecting its introspective capacity.

5.5.2 Support Vector Machine

SVM classification is well established in robotics so that we provide here only an overview. For a detailed account the reader is referred to, for example, Burges [1998]. SVMs are based on a linear discriminant framework which aims to maximise the margin between two classes. The model parameters are found by solving a convex optimisation problem, thereby guaranteeing the final classifier to be the best feasible discriminant given the training data. Once training is complete, predictions on future observations are made based on the signed distance of the observed feature vector from the optimal hyperplane, defined by the weight vector $\mathbf{w}$ and bias w_0, such that

$$f(\mathbf{x}_*) = \mathbf{w}^\top \phi(\mathbf{x}_*) + w_0 = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_*) + w_0, \tag{5.8}$$

where N is the size of the training set, α_i refers to a Lagrange multiplier associated with datum i, w_0 denotes a bias parameter, ϕ refers to the feature map, and $k(\mathbf{x}_i, \mathbf{x}_j)$ denotes the kernel function.

The parameters α_i and w_0 characterising the discriminant function are obtained by an optimisation procedure, and α_i is then non-zero only for *support vectors* $\mathbf{x}_i$. The SVM algorithm selects a particular weight vector (as defined by the *support vectors*), which gives rise to a *maximum* margin separator.

The kernel function amounts to a scalar product between two data, which have been transformed from d-dimensional feature space into some higher dimensional space. The nature of this mapping between spaces is inherent in the choice of kernel and need not be specified explicitly (the kernel trick). The regularisation and kernel parameters are learnt over a grid of powers of two, from 2^{-4} to 2^4, using ten-fold

cross-validation. We discuss our choices of kernel functions in Section 5.5.5.

In its original form, the SVM classifier output is an uncalibrated real value. A common means of obtaining a probabilistic interpretation is by using Platt's method [Platt, 1999]. This algorithm was later improved by Lin et al. [2007], which is implemented in the library we use for all SVM training, calibration, and testing, LIBSVM [Chang and Lin, 2011]. Here, using a hold-out set not used for classifier training, a parametric sigmoid model is fit directly to the class posterior $p(y_* = C_2 \,|\, f(\mathbf{x}_*))$, such that

$$p(y_* = C_2 \,|\, f(\mathbf{x}_*)) \quad = \quad \frac{1}{1 + \exp(Af(\mathbf{x}_*) + B)}. \tag{5.9}$$

The sigmoid parameters A and B are determined using Newton's method with backtracking line search. Note that class likelihoods are derived here using only a *single* estimate of the discriminative boundary obtained from the classifier training procedure. No other feasible solutions are considered. Hence, the predictive variance of the discriminant $f(\mathbf{x})$ is not taken into account while determining probabilistic output [Rasmussen and Williams, 2006]. Although there is no guarantee that the method converges, in general it works very well and finds the global optimum owing to the convexity of the objective function.

SVMs are an extremely popular method for classification in a multitude of settings, including but not limited to the classification of human gestures in videos [Schüldt et al., 2004], regions of images into semantic groups such as sky, building, skin, road [Cusano et al., 2003], face detection in images [Osuna et al., 1997], the angle of a face in an image [Huang et al., 1998], text classification [Tong and Koller, 2002], and protein classification [Eskin et al., 2002]. Their reasonable training times, readily available source code, and competitive accuracy across many domains have made them particularly attractive for machine learning applications. The linear

SVM especially is popular for object detection in video due to the number of methods available to make testing images very fast [Dubout and Fleuret, 2012]. SVMs are also being used as the classification step for image detection systems in which the features are learned via a deep neural network [Weston et al., 2012].

5.5.3 LogitBoost

Boosting is a widely used classification framework which involves training an ensemble of weak learners in sequence [Bishop, 2006]. Each weak learner $h(\mathbf{x})$ is trained using a weighted combination of the points in the dataset, where the importance of each point depends on whether it was correctly classified by the previous classifiers. Predictions from a boosted classifier are obtained using a weighted combination of the individual weak learner outputs such that

$$\text{sign}(f(\mathbf{x}_*)) \quad = \quad \text{sign}\left(\sum_{i=1}^{M} w_i h_i(\mathbf{x}_*)\right), \tag{5.10}$$

where M is the number of weak learners used.

LogitBoost [Friedman et al., 1998] is a popular choice for a boosting-based classifier as it natively outputs class probability estimates following a calibration via a sigmoid. Weak learners are often chosen to be decision trees and training is conducted by fitting additive logistic regression models by stage-wise optimisation (using Newton steps) of the Bernoulli log-likelihood. The algorithm works in the logistic framework and yields a predictor function $f(\mathbf{x})$ learnt from iterative hypothesis training. Cross-validation is used to set parameters like the learning rate, tree depth, and the number of boosting rounds. The class-conditional probabilities are obtained from the predictor function via

$$p(y_* = C_1 \,|\, \mathbf{x}_*) \quad = \quad \frac{\exp(f(\mathbf{x}_*))}{\exp(f(\mathbf{x}_*)) + \exp(-f(\mathbf{x}_*))}, \tag{5.11}$$

which is the same sigmoid used in the SVM in Section 5.5.2 with parameters $A = -2$ and $B = 0$. The procedure possesses asymptotic optimality as a maximum likelihood predictor [Friedman et al., 1998, Hastie and Tibshirani, 1990]. However, the method of converting the output of the predictor function to class-conditional probabilities is not fully probabilistic and does not account for variance in the underlying predictor function. In our experiments we use 500 learners for training. Throughout this work we use the MATLAB implementation of LogitBoost for classifier training and testing.

LogitBoost has been used for classifying protein structures [Cai et al., 2006], text classification [Kotsiantis et al., 2006], and speech segmentation [Ziólko et al., 2008]. Dettling and Bühlmann [2003] use LogitBoost for tumor classification, on the basis that it outperforms AdaBoost in cases with particularly noisy data or class imbalances. Boosted decision trees in particular appear to be popular for applications with large quantities of data, because the trees are very quick to train.

Because the LogitBoost classifier does not consider a variety of possible models, we consider it to be a single-discriminant classifier and expect that it will suffer from the same introspective issues as the SVM.

5.5.4 Random Forests

Random Forests [Breiman, 2001] are made up of an ensemble of decision trees generated via bagging. Bagging (a partial portmanteau of "bootstrap aggregating") involves creating multiple classifiers using different subsets of some aspect of the training data, in this case two aspects are bagged simultaneously: the training data, and the feature dimensions. During testing, the output $p(C_2)$ is the fraction of the individual trees which classified the datum as being from that class.

The trees contain multiple binary nodes or branches, each of which thresholds on a particular feature dimension of the data, learning the threshold which helps

split the training data into the two classes. We have set each tree to use a number of feature dimensions equal to the square root of the total number, as recommended by the literature, with a total of 500 trees. Throughout this work we use the Bagged Decision Tree functions in the MATLAB statistics toolbox (which is an implementation of Random Forests) for both training and testing.

Random Forests have been used for classifying invasive plant species and rare lichen species in various US national parks [Cutler et al., 2007], land cover in remote sensing scenarios [Gislason et al., 2006], road signs [Zaklouta et al., 2011], ranking the importance of certain driver and environmental characteristics on pre-crash manoeuvres in car accidents [Harb et al., 2009], and object segmentation in images [Schroff et al., 2008]. Fernández-Delgado et al. [2014] compare 179 classification algorithms on 121 data sets and find Random Forests to be the best family of classifiers of all in terms of accuracy.

This combination of many differing decision boundaries (one boundary per tree) represents a sampling and then averaging over the version space, similar to the marginalisation over version space which takes place in the Gaussian process classifier. A crucial difference is that in the GPC, each possible model is weighted by its likelihood, and in Random Forests each tree is weighted equally. However, these trees are carefully selected to separate the chosen subset of training data, so this biasing is in a sense a $\{0, 1\}$ weighting. This could be thought of as sampling 500 decision boundaries from the shaded region in Figure 5.2b and taking an expectation over their responses. This suggests that they should behave in a more introspective manner than the other single-discriminant frameworks like LogitBoost and the SVMs, but perhaps a more sensitively weighted combination of the trees could perform better.

5.5.5 Kernels

Evaluation of the discriminant function for an SVM and the covariance matrix for GPC inference both require the specification of a kernel (or *covariance*) function, $k(\cdot, \cdot)$. A rich body of literature exists on different choices of kernels for both frameworks. However, since our focus here is on a like-for-like comparison of different classification frameworks we choose two representative kernels rather than performing exhaustive model selection to optimise performance for a particular application. Firstly, as an example of a simple kernel function, we consider the linear kernel defined as

$$k_{LIN}(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \mathbf{x}_i^T \mathbf{x}_j + r, \tag{5.12}$$

where r is a constant real number. The linear kernel is an apt choice where a linear separation of the data in feature space leads to adequate performance or where computational efficiency is of the essence. Often, however, a more sophisticated, non-linear kernel is required. In this category we use the *squared exponential* (SE) function as a canonical representative. The SE kernel with length scale parameter l and variance σ^2 is defined as

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \sigma^2 \ \cdot \ \exp\left(-\frac{1}{2l^2}||\mathbf{x}_i - \mathbf{x}_j||^2\right). \tag{5.13}$$

In the context of an SVM, the SE function is more commonly known as a *radial basis function* (RBF).

We expect the choice of kernel to be crucial to a classifier's introspective capacity. A classifier using a SE kernel is able to 'wrap' the training data in a way that a classifier using a linear kernel cannot, allowing it to envelop a much larger volume of the feature space with high uncertainty. This is shown in Appendix B, where in

Figure B.1d we see the SVM with a non-linear kernel enclose the data, with much of the surrounding space near $p = 0.5$. However, the linear SVM in Figure B.1e is only uncertain for the narrow band between the green contours. The same is true for the linear GPC, although the contours fan out away from the data, resulting in a larger proportion of the space away from the training data being high-uncertainty. We therefore expect a classifier with a non-linear kernel to be more uncertain than its linear counterpart when the test data are far from the training data, and therefore to be more introspective.

Cover's theorem states that any collection of binary data are more likely to be linearly separable if the dimensionality is increased [Cover, 1965]. One effect of this is that for high-dimensional data, classifiers with linear kernels can have commensurate accuracy to those with non-linear kernels [Garrett et al., 2003]. However, the linear kernel is a degenerate case of the SE kernel, and so in theory the linear classifier should never outperform a non-linear kernel with the correct hyper-parameters [Keerthi and Lin, 2003]. However, to our knowledge the effects of kernels and dimensionality on introspection have not yet been investigated.

Because the kernel function is the metric used to determine the distance between two points, this choice is tightly bound up with our proposition that a good treatment of distance leads to introspective power. It is common to choose a kernel based on knowledge of the structure of the data in question, for instance the periodic kernel can be used to classify astronomy star surveys based on the changing variance of their brightness [Wachman et al., 2009]. If more flexibility is required, one can use a *locally-periodic kernel*, allowing us to model data for which the length scales or amplitudes of the periodicity change over time [Duvenaud, 2014].

Kernels can be combined to produce interesting effects, often by adding or multiplying two existing kernels together to produce another [Duvenaud, 2014]. Another variation is to use a kernel that changes its properties depending on the particular

element of the input in question. In this thesis we use multiple-input kernels because our features are represented by a vector $\mathbf{x}^d$ (where d is the dimensionality or number of inputs) rather than a scalar (where $d = 1$), but the SE kernel we use is *isotropic*. This means that there is a single length scale for all the input dimensions. Alternatively, it is common to use Automatic Relevance Determination (ARD), which is a SE kernel for which there is a length scale *per dimension*. However, this results in a number of hyper parameters which scales with the dimensionality of the input, leading to much longer training times.

Another possible way to improve introspection could be to use particular non-stationary kernels. In this thesis we use both: the SE kernel is non-stationary (it depends only on $\mathbf{x_i} - \mathbf{x_j}$), and the linear kernel is stationary (the values of $\mathbf{x_i}$ and $\mathbf{x_j}$ are relevant, not just their separation from each other). Perhaps it is possible to use a kernel which is explicitly more uncertain in some regions of feature space. We leave this as future work.

5.6 Analysis of Non-Stationary Data

In order to determine a classifier's behaviour when it makes incorrect classifications, we must first consider when a classifier can be wrong. The trained classifier will partition the feature space into regions, where any point within a particular region will return a corresponding probability value. Typically, we hope that the training procedure will use the training data to create this set of regions to correspond to accurate classification of the test data. There are many cases in which the regions will be chosen in such a way that this is not achieved. A simple case is where the training data are incorrect. Erroneous class labels for the training set will shape the model and hence the regions in such a way that they will generalise poorly to the test set. However, in this thesis we assume that the labels are correct. Another is

where the model is incorrectly learned or poorly chosen for the data. However, the case we wish to dwell on is one where the training set does not capture every aspect of the test set, otherwise known as *non-stationarity* between training and testing. This occurs commonly, for instance if we train a car classifier on only frontoparallel views, but then test it on images containing oblique views, or if the illumination of the test images is different.

We argue that regardless of the size of the training set, it will inevitably omit certain facets of the data. Consider, for example, the space of pictures of cars. Many aspects of the appearance of cars are captured by these data: their shapes, colours, illumination, possible occlusions, orientations of the viewer, and more. Typically the assumption is that the training and test data are independent and identically distributed (*i.i.d.*), and thus we hope that a representative subset of the space will allow the classifier to generalise well. However, this assumption is routinely violated in robotics, and in practice the test sets may appear to be from a different distribution from the training set. One possible result of this is that test data will stray further away from the training set in feature space. It is these data which are likely to be misclassified, and thus should be regarded with high uncertainty.

The choice of features is clearly important here. On the one hand, we can use a particular feature type which mitigates the effect of illumination, or perhaps rotation. However, we simultaneously desire it to be sufficiently rich to represent aspects of the data that allow a classifier to separate the classes. If this desire is achieved it is more likely that the less representative the training set is, the greater its distance in feature space to the test points. This would allow introspection to lead to error prevention. We desire the uncertainty of a classifier to increase as we query it on points further away from the training set. Therefore, we propose that it is the classifiers' treatment of distance in feature space which determines their introspective capacities.

In this section we test the commonly-used classification frameworks from Section 5.5 first on synthetic data, and then on data from real images of road signs. Specifically, we analyse the uncertainties of the classifiers as we test them on data which are not represented within the training set, and looking for those which consistently give higher uncertainty in these cases.

5.6.1 Synthetic Data

We start by analysing the uncertainty of the classifiers when tested on simple, multi-dimensional Gaussian data as described in Section 3.5. We hold the number of feature dimensions constant at 64, and vary the magnitude of the differences between the means of the training and test sets M_{μ}^{+} and M_{μ}^{-} from 0 to 2, while $M_{\sigma}^{+} = M_{\sigma}^{-} = 0$. Note that except where otherwise stated, the classifiers are trained on 50 positive examples and 50 negative examples. Figure 5.3 shows the uncertainties of the classifiers for the test sets. Each point on the graph is the mean of 10 experiments using the same methodology but with differing random draws. The main observation is that for the multi-discriminant classifiers (the GPC variants and the random forest) the uncertainty grows to unity as the test set diverges from the training set. However, for the single-discriminant classifiers (the SVMs and LogitBoost) the uncertainty is constant and seems relatively unaffected by this increase in distance. An introspective capacity would be evidenced by an increase in uncertainty as distance increases, so the multi-discriminant classifiers appear to be more introspective. We can repeat this experiment while changing M_{σ}^{+} and M_{σ}^{-}, and keeping M_{μ}^{+} and M_{μ}^{-} fixed at 0, the results of which are shown in Figure 5.4. Here we see exactly the same behaviour: the multi-discriminant classifiers become increasingly uncertain as the variance of the test distributions increase, but the single-discriminant classifiers remain approximately constant in terms of uncertainty.

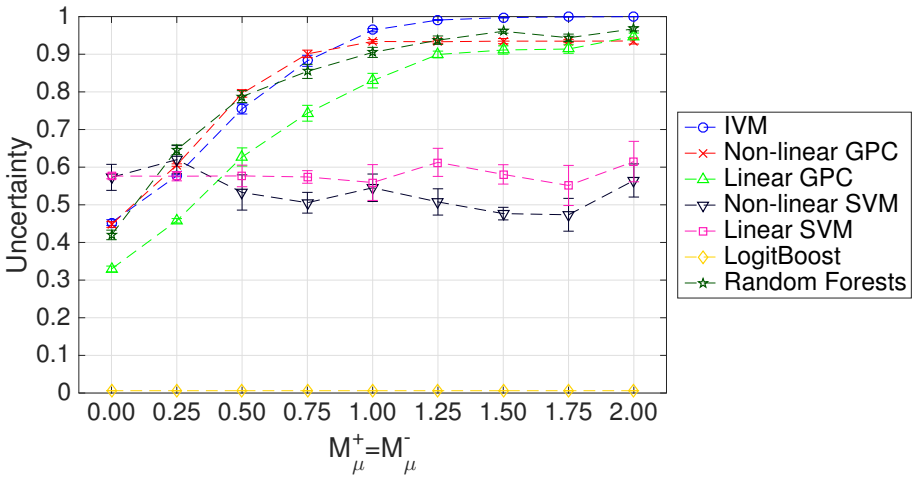


Figure 5.3: We show the test uncertainties of the classifiers as we change M_μ^+ and M_μ^-, affecting the means of the test distributions. As we travel along the horizontal axis, the test distribution is increasingly different from the training distribution, so an introspective classifier would show increasing uncertainty. There are 64 dimensions to the data. The error bars show the standard error of the mean uncertainty over 10 independent runs.

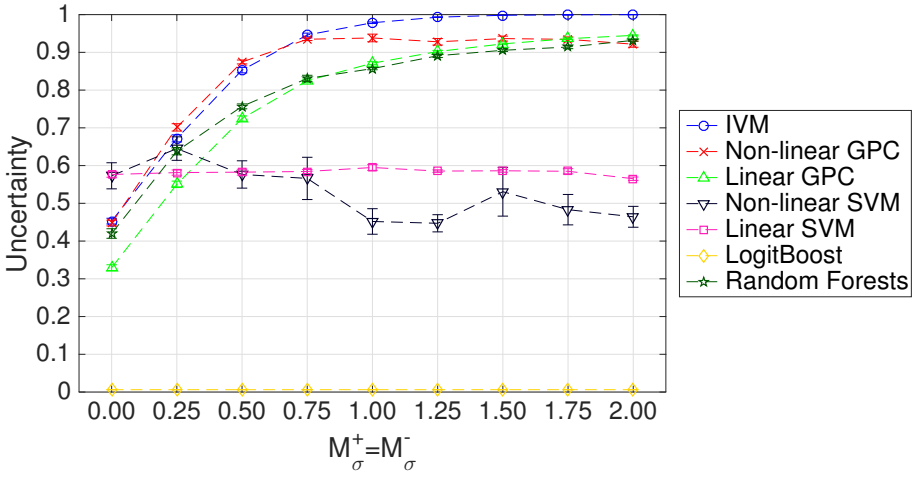


Figure 5.4: The test uncertainties of the classifiers as we change M_σ^+ and M_σ^-, affecting the standard deviations of the test distributions. As we travel along the horizontal axis, the test distribution is increasingly different from the training distribution, so an introspective classifier would show increasing uncertainty. The dimensionality is 50, and the error bars show the standard error of the mean uncertainty over 10 independent runs.

What aspects of the data can affect this behaviour? The concept of distance in high-dimensional space is complex and unintuitive, with various interesting effects. For instance, the majority of the points drawn from a 1D Gaussian distribution will lie near to the mean. However, as dimensionality increases, the distribution of points is distributed far from the mean, like the skin of an orange [MacKay, 2003]. Therefore if dimensionality plays a role in distance, it is reasonable to expect it to affect uncertainty. Setting $M_{\mu}^{+} = M_{\mu}^{-} = 3$, and $M_{\sigma}^{+} = M_{\sigma}^{-} = 0$ to ensure that the test means are far from the training means, we vary the dimensionality of the data from 1 to 256 in powers of 2. The results shown in Figure 5.5 demonstrate that the dimensionality does indeed greatly affect the uncertainties of some of the classifiers. The non-linear SVM starts off very uncertain, and then becomes more and more confident as the dimensionality increases. The non-linear GPC-based classifiers are consistently uncertain regardless of dimensionality. The classifiers with non-linear kernels are confident in a low-dimensional space, but as the dimensionality increases they diverge. The multi-discriminant linear-kernel classifiers become very uncertain, and the single-discriminant linear-kernel SVM and LogitBoost do not. Interestingly, the two SVMs converge in high-dimension. This shows evidence to support the proposition that increased feature space dimensionality decreases the effectiveness of using a non-linear kernel, as implied by Cover's theorem [Cover, 1965]. It further suggests that it is not the kernel which produces the difference between the GPCs and the SVMs in high-dimensionality, but the underlying algorithms.

We expect that the effects of increasing the dimensionality can be, to some extent, counteracted by an increase in the number of training data. An increase in the number of training data will likely result in more of the space being 'close' to a training datum, shrinking the version space and thus reducing the uncertainty of multi-discriminant classifiers. In Figure 5.6 we see that indeed, uncertainty generally decreases as we increase the number of training data. For the multi-discriminant

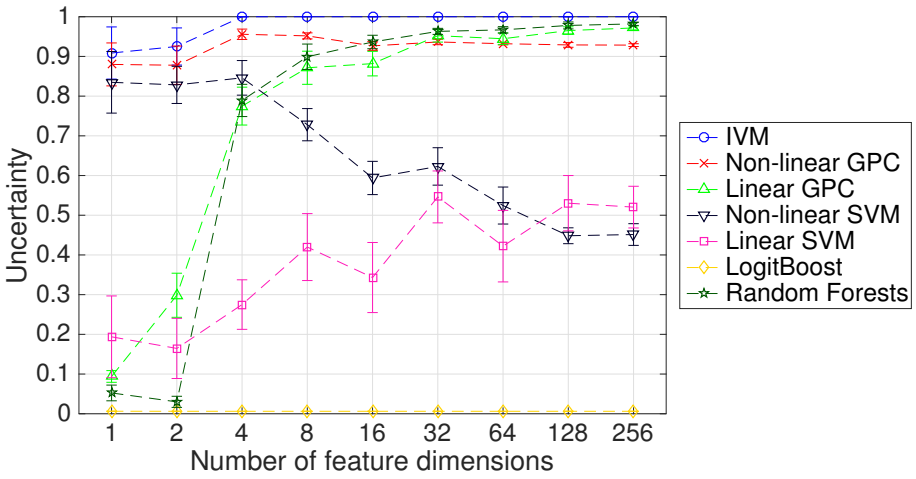
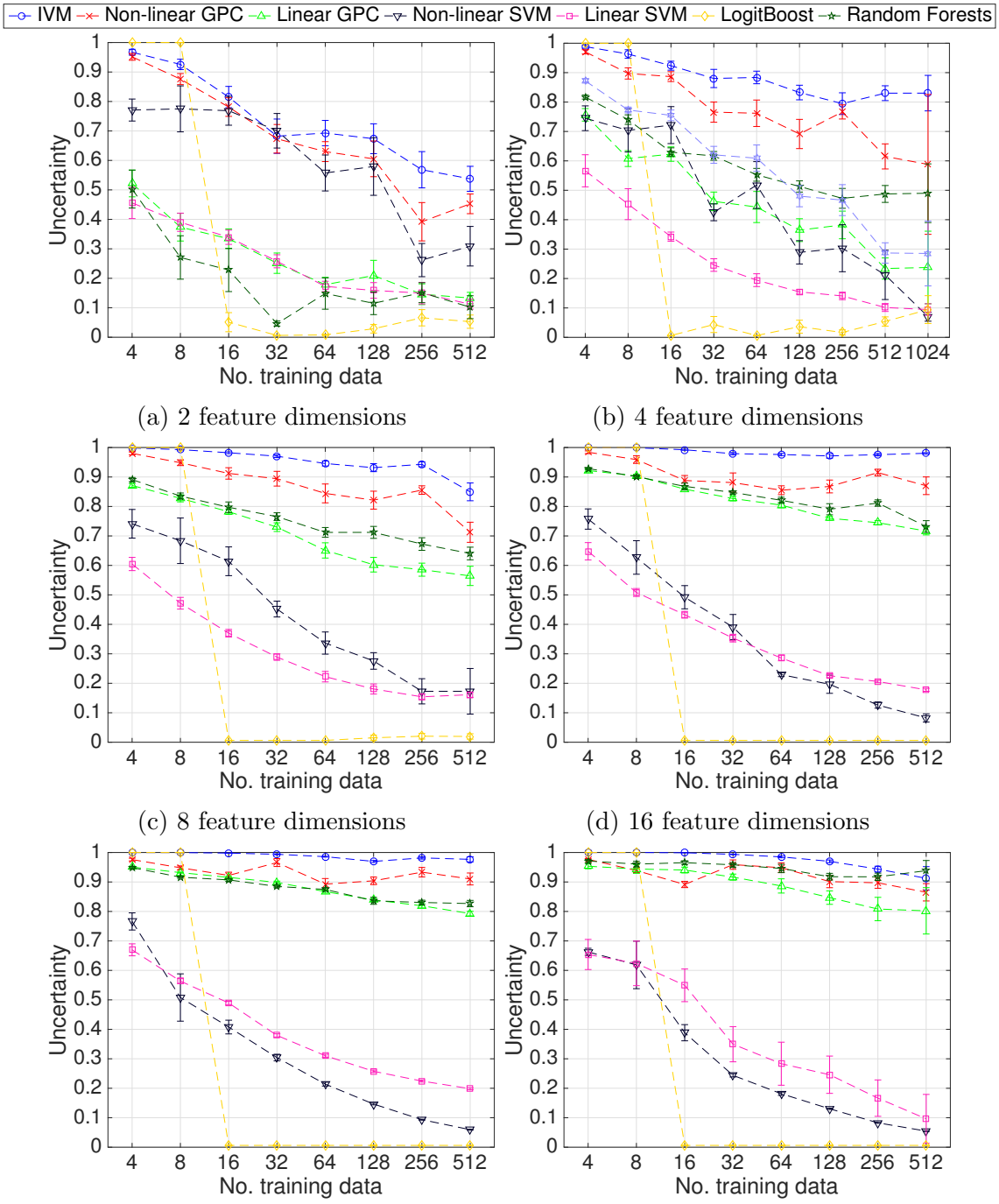


Figure 5.5: The test uncertainties of the classifiers as we vary the dimensionality of the data. The parameters of the test distributions are $M_{\mu}^{+} = M_{\mu}^{-} = 3$, and $M_{\sigma}^{+} = M_{\sigma}^{-} = 0$. The test distributions are very different from the training distribution, so an introspective classifier would show high uncertainty. Here, the single-discriminant classifiers are more confident than the multi-discriminant classifiers once we exceed 8 feature dimensions. The error bars show the standard error of the mean uncertainty over 10 independent runs.

classifiers, increasing the training data becomes less effective as the dimensionality increases. However, beyond four dimensions the SVMs and LogitBoost seem relatively invariant to dimensionality (as in Figure 5.5), but still become more confident with more training data, even in 64 dimensions. We propose that this is because the SVM is designed to be relatively insensitive to feature dimensionality, in that its generalisation error is independent of the feature dimension [Vapnik and Vapnik, 1998].

We have shown that with real classifiers on synthetic data, the multi-discriminant classifiers exhibit the desired high uncertainty when we test them on data which are far away from the training data, providing that the dimensionality is at least 8. This tendency generally increases with dimensionality and decreases with number of training data. The single-discriminant classifiers, however, seem to be more affected by number of training data than dimensionality, and overall exhibit much less uncertainty for distant test data.



(a) 2 feature dimensions

(b) 4 feature dimensions

(c) 8 feature dimensions

(d) 16 feature dimensions

(e) 32 feature dimensions

(f) 64 feature dimensions

Figure 5.6: Here we show the test uncertainties of the classifiers as we vary the number of training data for a given feature dimensionality. The test distributions are very different from the training distributions, and thus an introspective classifier will exhibit high uncertainty. As we increase the dimensionality, the multi-discriminant and single-discriminant classifiers diverge, with the former exhibiting a stronger introspective behaviour. The data are split evenly between positive and negative classes, and error bars show the standard error of the mean over 10 independent experiments.

Next we will explore the effects on real data, in order to examine the consistency our expectations and conclusions from this experiment.

5.6.2 Real Data

Now that we have an indication of the introspective tendencies of multi-discriminant classifiers on synthetic data, we must validate and confirm that the findings are consistent for real data. We push the idea of a 'distant' test class to the extreme by training classifiers on two distinct classes, and then testing them on a completely different class. The uncertainties of this unseen or 'third' class will demonstrate the behaviours of the classifiers when they are trained and tested on extremely non-stationary data. Similarly to the previous section, the best result is a high uncertainty for this third class. As examples of classes typically encountered in autonomous driving applications we use a subset of the GTSRB dataset (see Section 3.2).

We arbitrarily select two classes for training: *stop* and *lorries prohibited*. Classifiers are trained on these two classes using a balanced training set of 400 data (200 per class). Classifier performance is evaluated using standard metrics on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The results are shown in Table 5.1, and show that classification performance by the commonly-used metrics (precision, recall, and F_1 measure, as discussed in Section 3.7) is commensurate across all classifiers. The corresponding precision-recall curve confirms the perfect separation of the classes and has been omitted here as it is otherwise uninformative. The classifiers are then tested on 200 instances of previously unseen classes *roadworks ahead*, *keep left*, *70kph*, *yield*, and *right ahead*. The uncertainty histograms for both the seen and the unseen test classes are shown in Figure 5.7.

All classifiers are confident when tested on classes which were present in the

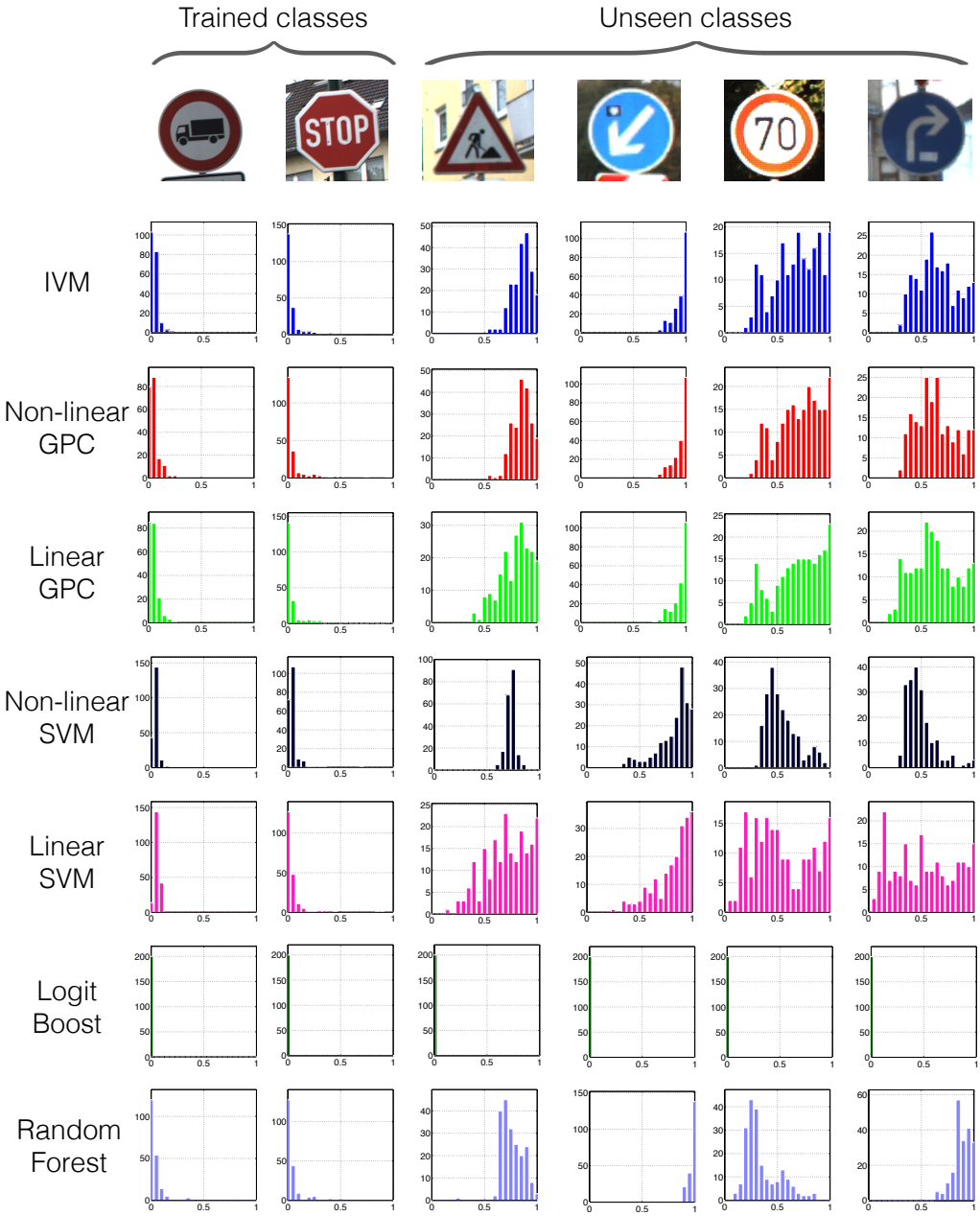


Figure 5.7: Normalised Entropy (NE, uncertainty) histograms of the marginal probabilities for a variety of classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on not only the training classes, but also classes which do not appear in the training set (*roadworks ahead*, *keep left*, *70kph*, and *right ahead*). Higher values for normalised entropy imply more uncertainty in classifier output, so we expect the more introspective classifiers to be certain (low NE, left-hand end of the horizontal axis) on the trained classes and uncertain (high NE, right-hand end of the horizontal axis) for the unseen classes.

Classifier	**Precision**	**Recall**	F_1
IVM	1.000	1.000	1.000
Non-linear GPC	1.000	1.000	1.000
Linear GPC	1.000	1.000	1.000
Non-linear SVM	1.000	1.000	1.000
Linear SVM	1.000	1.000	1.000
LogitBoost	1.000	1.000	1.000
random forest	1.000	1.000	1.000

Table 5.1: The classification performance when separating *stop* sign from the *lorries prohibited* signs from the GTSRB data set. Note that different class combinations were found to yield classifiers of similar quality.

training set, which is what we would expect. For the unseen test classes, the mean uncertainties for the GPC-based classifiers (IVM, non-linear GPC, and linear GPC) and the random forests are more consistently high than those of the other classification frameworks, indicating that they reliably exhibit greater uncertainty in their judgement. Conversely, the LogitBoost classifier is extremely confident in all of its classifications with a very narrow distribution, and the non-linear and linear SVMs have inconsistent levels of uncertainty. These are effects consistently observed throughout our experiments. The unseen sign for which the classifiers respond with the lowest uncertainty (greatest confidence) is the 70 kph sign. We propose that this is due to its visual similarity with one of the training classes, namely the 'lorries prohibited' sign. This is consistent with our findings on the synthetic data that multi-discriminant classifiers are more introspective than single-discriminant classifiers.

In order to mitigate any influences of the specific training and test data selected we repeated the above experiment across a number of random dictionaries, data samples, and unseen classes. Specifically, for each of five different unseen classes, we perform forty iterations of classifier training and testing with a random dictionary and training and test datasets resampled for each run. The results, presented in Table 5.2, are consistent with those in Figure 5.7 in that the GPCs and random

forest tend to be more consistently uncertain for the unseen test classes, while SVM and LogitBoost are more confident with an often significantly narrower distribution of normalised entropy values. The results in Table 5.2 indicate that the gap in uncertainty between the different frameworks is more pronounced for some unseen classes than for others. We attribute this to the varying degree of similarity in feature space between some unseen class and the classes in the training set. Our choice of template features results in the circle around the 70kph sign registering very closely to the one around the lorries prohibited sign.

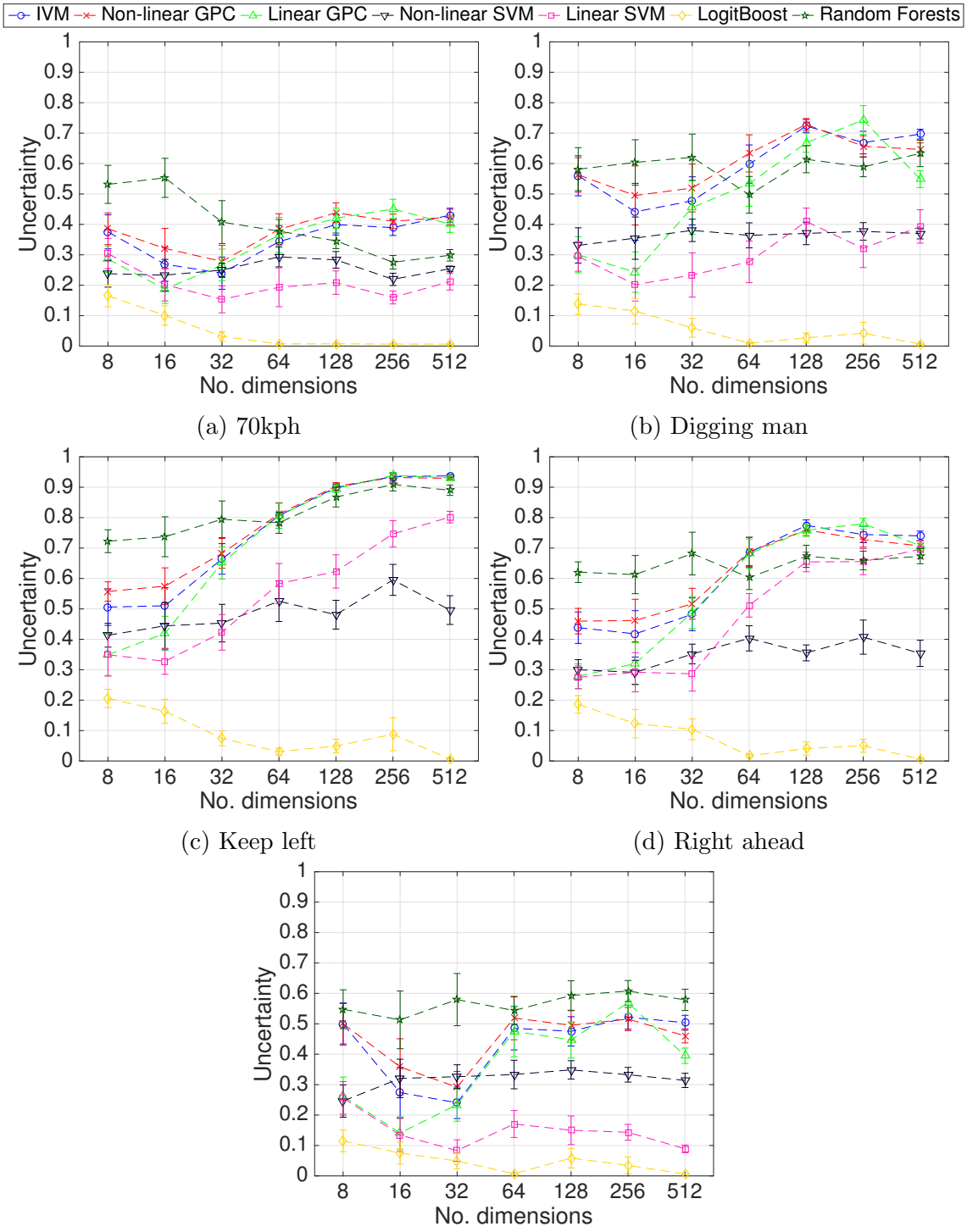
We draw the conclusion that when faced with test data which are not represented by the training data, the GPC-based classifiers and random forest are more consistently uncertain than the other classifiers, which is part of the introspective behaviour we seek.

Another direct comparison between synthetic and real data we can draw is to repeat the third class experiment, showing the change in uncertainty as we vary the dimensionality of the features. By choosing how many patches to use, we choose how many dimensions describe the data. In Figure 5.8 we show the uncertainties for the five unseen classes already discussed in this section. Notice that the order is preserved across all plots: the LogitBoost classifier is the most confident, then the SVMs, and lastly the multi-discriminant classifiers. Although the shapes of the plots are not exactly the same as those generated using synthetic data in Figure 5.6, the order of the classifiers is preserved, with the multi-discriminant classifiers exhibiting higher uncertainty than single-discriminant at 32 dimensions and above.

Is this third class situation relevant in robotics? Our classifiers will not always be encountering clear-cut third-class cases. More often, they will be used in detection tasks, where they must locate one particular class in an image against a broad background class. We expect that third class cases will not be rare in this detection

Test Class	Classifier	Uncertainty	
		$\mu\pm$ std. err.	$\sigma\pm$ std. err.
	IVM	**0.776 ± 0.081**	0.145 ± 0.030
	Non-linear GPC	0.751 ± 0.087	0.152 ± 0.029
	Linear GPC	**0.776 ± 0.108**	0.150 ± 0.041
	Non-linear SVM	0.476 ± 0.101	0.089 ± 0.056
	Linear SVM	0.664 ± 0.122	0.250 ± 0.041
	LogitBoost	0.019 ± 0.025	0.041 ± 0.073
	random forest	0.756 ± 0.137	0.149 ± 0.053
	IVM	**0.794 ± 0.117**	0.106 ± 0.026
	Non-linear GPC	0.779 ± 0.124	0.107 ± 0.024
	Linear GPC	0.777 ± 0.202	0.124 ± 0.058
	Non-linear SVM	0.537 ± 0.126	0.028 ± 0.036
	Linear SVM	0.494 ± 0.239	0.222 ± 0.049
	LogitBoost	0.016 ± 0.022	0.031 ± 0.059
	random forest	0.736 ± 0.166	0.078 ± 0.027
	IVM	0.539 ± 0.140	0.173 ± 0.023
	Non-linear GPC	0.546 ± 0.144	0.168 ± 0.023
	Linear GPC	**0.569 ± 0.166**	0.177 ± 0.026
	Non-linear SVM	0.407 ± 0.077	0.076 ± 0.053
	Linear SVM	0.315 ± 0.195	0.197 ± 0.058
	LogitBoost	0.008 ± 0.004	0.012 ± 0.026
	random forest	0.394 ± 0.121	0.138 ± 0.029
	IVM	0.579 ± 0.133	0.137 ± 0.020
	Non-linear GPC	0.577 ± 0.130	0.136 ± 0.019
	Linear GPC	0.585 ± 0.188	0.151 ± 0.029
	Non-linear SVM	0.488 ± 0.111	0.039 ± 0.034
	Linear SVM	0.177 ± 0.127	0.155 ± 0.056
	LogitBoost	0.014 ± 0.019	0.030 ± 0.056
	random forest	**0.668 ± 0.161**	0.113 ± 0.027
	IVM	**0.931 ± 0.025**	0.080 ± 0.026
	Non-linear GPC	**0.934 ± 0.021**	0.079 ± 0.023
	Linear GPC	0.925 ± 0.031	0.085 ± 0.027
	Non-linear SVM	0.641 ± 0.168	0.100 ± 0.047
	Linear SVM	0.705 ± 0.142	0.212 ± 0.049
	LogitBoost	0.059 ± 0.103	0.077 ± 0.127
	random forest	0.904 ± 0.089	0.088 ± 0.043

Table 5.2: The means and standard deviations of uncertainties from 40 iterations of classifier training and testing, each with a randomly created feature dictionary and both training and test datasets resampled. Results are presented for classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on five different unseen classes as shown. The most uncertain classifier category is consistently multi-discriminant.



(a) 70kph

(b) Digging man

(c) Keep left

(d) Right ahead

(e) Yield

Figure 5.8: Having trained the classifiers to differentiate between *stop* and *lorry prohibited* signs (380 of each), we test on clusters of new road signs, unseen in the training set. A high uncertainty is desirable. We vary the number of features used to describe each datum from 8 to 512, and perform 10 randomised train+test experiments per sign per feature size. The error bars show the standard error of the mean.

scenario. We may think of large data sets as being dominated by a few common modes (or classes or sub-classes). However, while a few classes may indeed be common, the majority of the data set is often made up of many rare classes. This is has been shown to be true across various domains, and is known as the *long tail* of big data (first documented in the context of book sales by Brynjolfsson et al. [2003]). The relevance of this to introspection is that in big data situations, a significant number of test cases are likely to be significant deviations from the training data, and thus the ability to detect 'third class' situations is likely to be important in decision-making.

5.6.3 Discussion

How much does the choice of kernel affect the introspective capacity of a classifier? Varying the choice of kernel produces different behaviours, and it appears to be possible to instil an improved introspective sense with an appropriate choice of kernel, particularly when the feature dimensionality is low. This is due to the relationship between classification confidence and distance in kernel space. In the GPC, a test datum which lies far away from the training data (in kernel space) yields a higher predictive variance and therefore a more uncertain classification. As we have illustrated, single discriminant models like the SVM function similarly, but one failing is that distance is only measured orthogonal to the decision plane, and that distance along the decision plane is not taken into account. But can we guarantee that new, unseen classes will be far away from our training data? We believe not. When using non-linear kernels, we expect that the kernel has found a warping of the feature space which adequately separates the two classes of training data, and that new, unseen classes will be sufficiently disparate from the training data in kernel space to yield an uncertain classification, but the latter is not guaranteed. Owing to the opaque nature of the kernel mapping, we cannot assume that points which

are desirably far apart in feature space will also be distant in kernel space.

It is also possible to be limited by the choice of feature with which to represent the data. We must also choose features which allow the distinction not only between the classes but which adequately represent the variations within them. Introspection would lead to a classifier which exhibits higher uncertainty in classification if the features are too rudimentary to make accurate decisions.

5.7 Uncertainty in Detection

In this section we examine the uncertainties of the classifiers when used in the context of image-based object detection leading to a decision. Specifically, we plot the uncertainties for each of the four possible decision outcomes: true positives, true negatives, false positives, and false negatives.

In both the synthetic and real experiments in Section 5.6 we deal with classification problems in which there are two distinct classes, and then we test on a third unseen class. The detection case is arguably more commonplace, in which a single class is separated from a broad (in terms of intra-class variation) *background* class. Here, the concept of a previously unseen class does not exist explicitly: the inherent assumption is that the data representing the background class capture any non-class object likely to be encountered. In practice this is rarely true, leading to a significant number of novel instances which often result in misclassification. While it could be argued that this issue can be ameliorated by increasing the size of the dataset used for training, we propose that the complexity of the feature space encountered during persistent, long-term autonomy will keep perplexing even the most expansively trained classifiers. This implies that it is reasonable to generalise from results on modestly sized data sets to what we might refer to as 'big data' [Murphy, 2012]. Our approach to introspection is grounded in the fact that the often cited assumption of

independent and identically distributed training and test data is routinely violated in robotics, because many 'rare' instances will be seen during testing that were not present during training.

We investigate the same classification frameworks as before on various detection tasks, which each have a salient positive class and a broad background class. We evaluate the classifiers on three data sets: TLR (traffic lights), Daimler Pedestrian, and KITTI (cars), as detailed in Chapter 3.

Number of data

As with the classification task, we first verify the efficacy of the features selected and the training procedures employed. Table 5.3 shows the classification performance for classifiers trained using the number of data shown in Table 3.3. We have chosen these quantities of data for several reasons.

Firstly, we are trying to highlight low-probability catastrophic events, which will be few in number for the size of the test sets we are considering here, but over the life-long autonomy we envisage for our robots will occur in non-negligible numbers; larger training sets reduce the prevalence of these low-probability events, but will never be able to eliminate them because of the aforementioned long tail characteristic of data sets.

Secondly, we are using off-the-shelf implementations of commonly-used classification frameworks to keep the comparisons relevant for the average user, and some of these (namely the GPC implementations) struggle to train successfully when using many (approximately 1000 or more) data with the numbers of feature dimensions we are considering here (shown in Table 3.1). In order to keep the numbers of training data below this while maintaining a reasonable number of positive examples, the training data are in the ratio 1:2 of positives to negatives.

The ratio of positive to negative data can vary wildly from application to appli-

cation. A very simple perception pipeline might use a sliding-window detector over an entire image, which may yield 0 to 100 positives to approximately 10^5 negatives. However, due to the fact that even the state of the art classifiers would make many false positive detections with this pipeline, it is common to use 3D information or prior maps to greatly reduce the portion of the image that needs to be searched [Enzweiler et al., 2012, Fairfield and Urmson, 2011]. Benenson et al. [2011] do this by creating a 'stixel world' directly from a stereo image, reducing the number of test windows in an image to 650. Barnes et al. [2015] use 3D 'scene priors' in which traffic lights have been labelled in order to predict where they will appear in images as the car revisits the same area. This prediction returns a confidence ellipse on the pixel location of the traffic light based on the accuracy of the vehicle's localisation within the prior map. This ellipse greatly reduces the search area for the traffic light. Gerónimo et al. [2010] reduce the number of test windows to around 2000 by computing a rudimentary depth image of the road using a stereo camera. Kalal et al. [2012] reject test windows with a small grey-value variance, eliminating "more than 50% of non-object patches (e.g. sky, street)".

Taking into account the plausibility of reducing the number of test windows in each image, we choose the sizes in the test sets to roughly contain a 1:10 ratio of positives to negatives.

Experimental set up

In the following experiment we train each of the classifiers on the same random subset of the training data, and test them on the same random subset of the test data. If the data set is a contiguous collection of frames, the overlap between the training and test sets is minimised by using the first N frames for training and the rest for testing, where no same positive instance is common to both. From the data sets which provide full images (TLR and KITTI), we extract a number of

random well-cropped positive instances, and a number of random negative instances at various scales such that there is never more than a 50% overlap with a positive window. The DP data set is available already cropped, so no further processing is required. Note that the KITTI data set contains a great many occluded and blurry positive examples in the ground truth, and so we expect that this data set will be the most challenging for the classifiers.

Having extracted the positive and negative training and test instances, we train each classifier on the training set. Each classifier then gives a probability of each test instance belonging to the positive class. We then calculate the normalised entropy for each probability using (5.1).

We wish to examine 'rare' classifier errors, and thus we use well cropped windows rather than performing sliding window detection over frames in order to avoid the need for any non-maximal suppression techniques. These techniques can use the relative strength of some well-classified overlapping windows (at various scales) to overcome a poorly-classified window in the same region. We are adamant that we must examine each possible test example individually in order to measure the effects of the varying levels of introspection that the classifiers exhibit. We carry out this process 20 times per data set, thus varying the exact training and test data each time.

Results

Figure 5.9 shows the corresponding precision-recall curves for the classifiers across the data sets. The detection task, having a varied background class and generally greater variation within the positive class, is more challenging than the classification task. Whereas in the classification setting the two distinct classes are likely to be more grouped in feature space, in the detection setting the broad background class is likely to be much more spread over the feature space than the positive class.

	TLR			DP			KITTI		
Classifier	**P**	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1
IVM	0.995	0.916	0.954	0.953	0.872	0.911	0.868	0.725	0.790
Non-lin. GPC	0.992	0.912	0.950	0.956	0.874	0.913	0.853	0.735	0.790
Linear GPC	0.988	0.899	0.941	0.956	0.875	0.914	0.816	0.708	0.758
Non-lin. SVM	0.996	0.920	0.956	0.959	0.869	0.912	0.836	0.749	0.790
Linear SVM	0.967	0.910	0.938	0.932	0.876	0.903	0.813	0.709	0.757
LogitBoost	0.978	0.908	0.942	0.961	0.794	0.869	0.826	0.681	0.747
Random F.	1.000	0.897	0.946	0.984	0.598	0.744	0.894	0.551	0.682

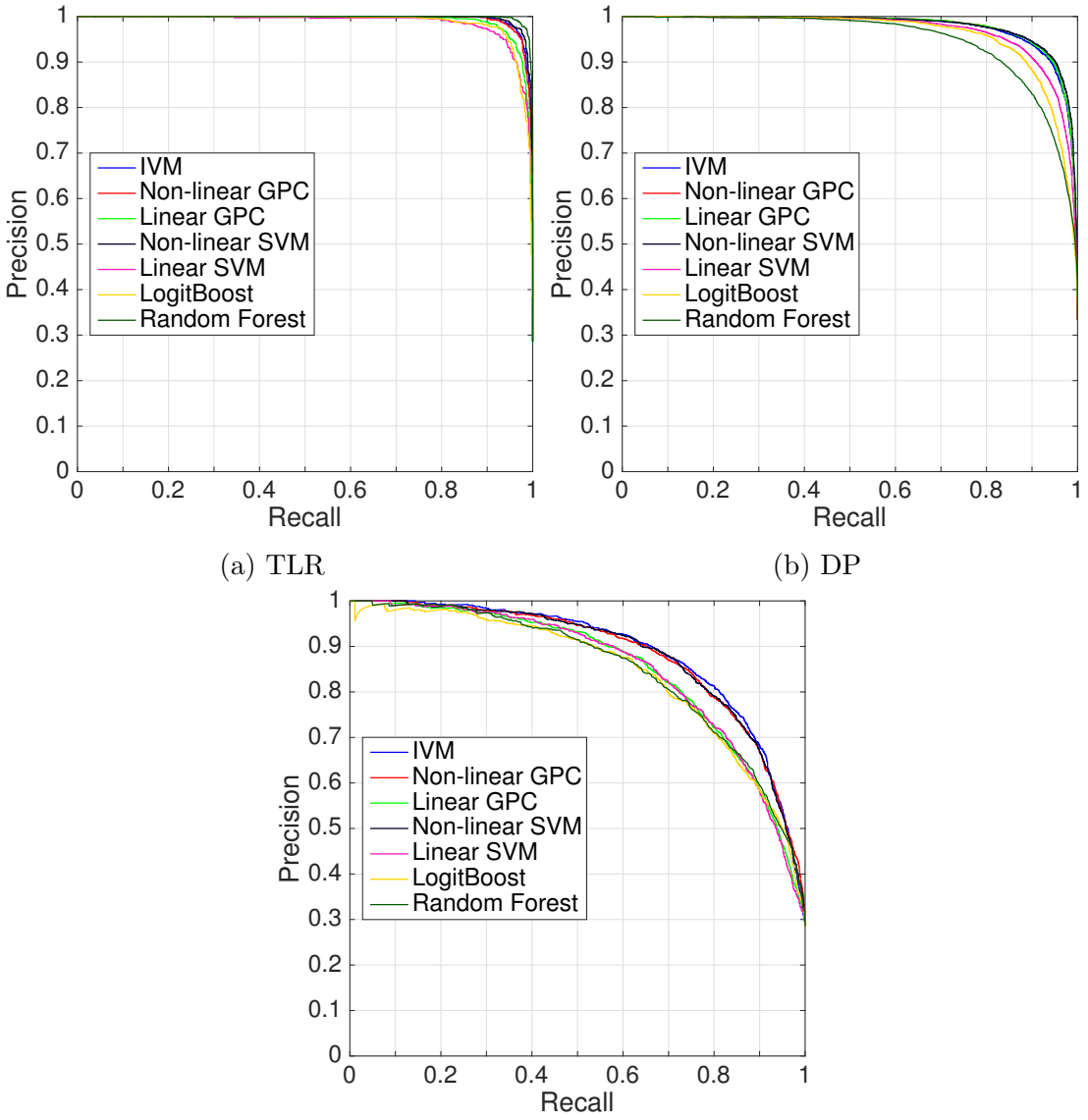
Table 5.3: The classifiers' performances for the detection tasks across data sets according to conventional metrics. Precision, recall, and F-measure are calculated by thresholding the classifiers' probabilities at 0.5. The SVMs and GPCs give very similar results across the data sets, with the LogitBoost and random forest performing slightly worse than the others with the more difficult data sets.

That said, classification performance according to the conventional metrics is still commensurate across all frameworks. According to Figure 5.9, the random forest performs best for the TLR data set, and the non-linear SVM and IVM perform consistently highly in the Daimler Pedestrian and KITTI data sets. The GPC-based classifiers all have commensurate performance in terms of precision and recall.

In Figures 5.10, 5.11, and 5.12 we show the frequency of each potential outcome (true positive, true negative, false positive, and false negative), specifically how many of each occurs below a certain threshold of uncertainty. In other words, we show the number of each outcome when examining decisions made with up to a certain level of confidence. There is one figure per data set.

The intuition here is that if we have an additional *safe* action such as waiting and gathering more data (e.g. [Velez et al., 2011]) or asking a human for guidance, we might use it for all test data that the robot perceives to be above a certain threshold of uncertainty [Lewis and Gale, 1994]. In that case, we are looking at how many correct and incorrect classifications (and therefore decisions) the robot considered safe.

It is desirable for a classifier to be close to the top left hand corner of the



(a) TLR (b) DP

(c) KITTI

Figure 5.9: Precision-recall curves for the three data sets. Note the increasing difficulty of the data sets, and the consistency and commensurate nature of the classifiers in terms of these metrics. In general we can see that the non-linear classifiers tend to perform better than the linear classifiers. (Best viewed in colour.)

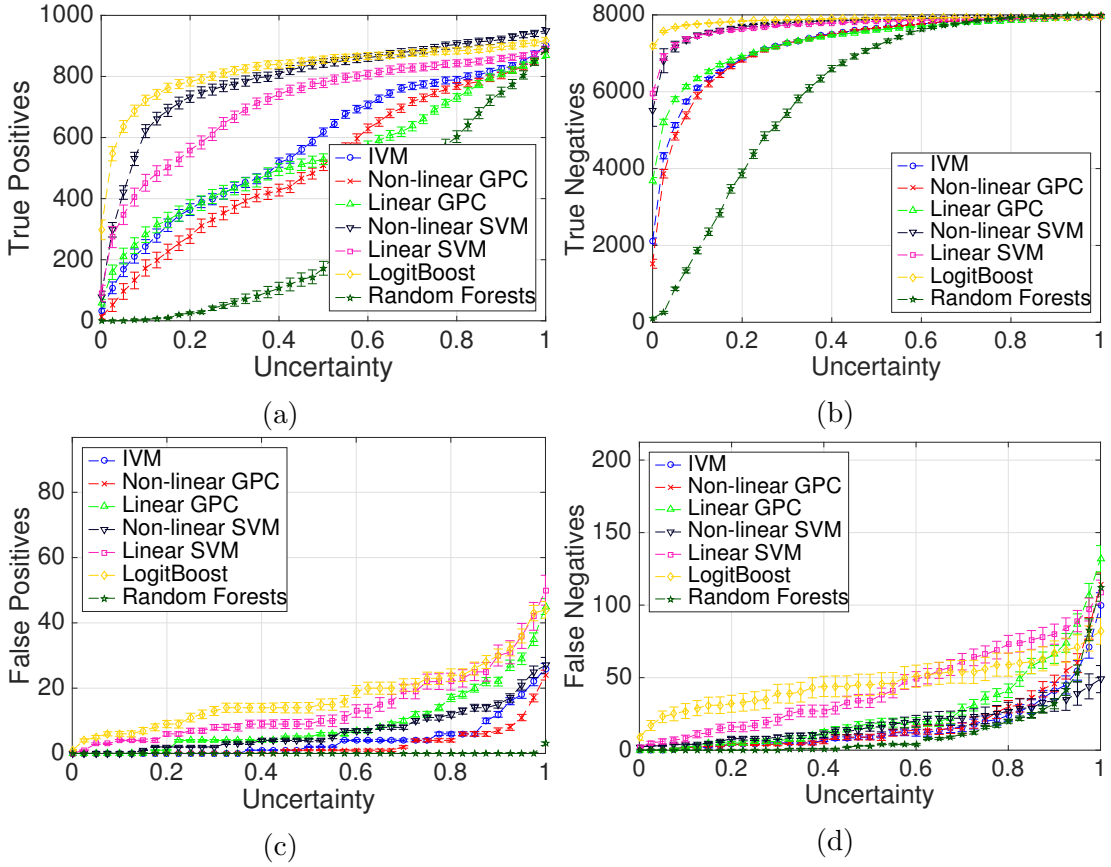


Figure 5.10: TLR: Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against classification uncertainty. The classifiers have been trained on 250 traffic lights against 500 background patches, and tested on 1,000 instances of traffic lights and 8,000 background patches. A more introspective classifier is one that simultaneously exhibits higher uncertainty when processing difficult instances (bottom right corner for false positives and negatives) and is more confident when it is correct (top left corner for true positives and negatives). Consequently, class decisions above a given uncertainty threshold are deferred since the output is deemed ambiguous. This is desirable since a single bad decision can have disastrous consequences. Note that of the linear classifiers, the GP and the random forest are most uncertain when they make mistakes, and thus more introspective than the linear SVM and LogitBoost classifiers. Of the non-linear classifiers, the SVM is more confident when making false negative errors than the GPC-variants. The error bars indicate the standard error over 20 independent runs. (Best viewed in colour.)

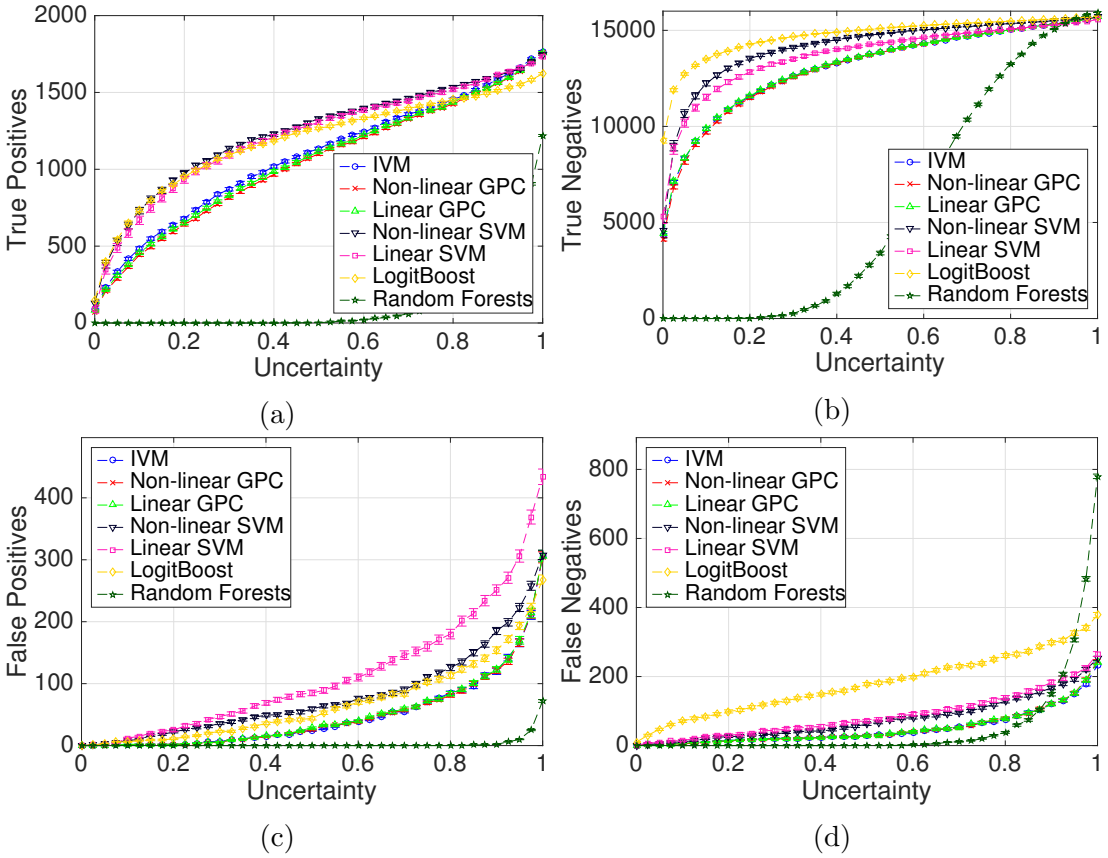


Figure 5.11: DP: Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against classification uncertainty. The classifiers are trained on 250 and 500 instances of pedestrians and background respectively, and are tested on 2,000 and 16,000 of those classes. See the caption for Figure 5.10 for more detail. Note that the multi-discriminant classifiers (IVM, both GPCs, and the random forest) are more uncertain when they make mistakes than the single-discriminant classifiers. The error bars indicate the standard error over 20 independent runs. (Best viewed in colour.)

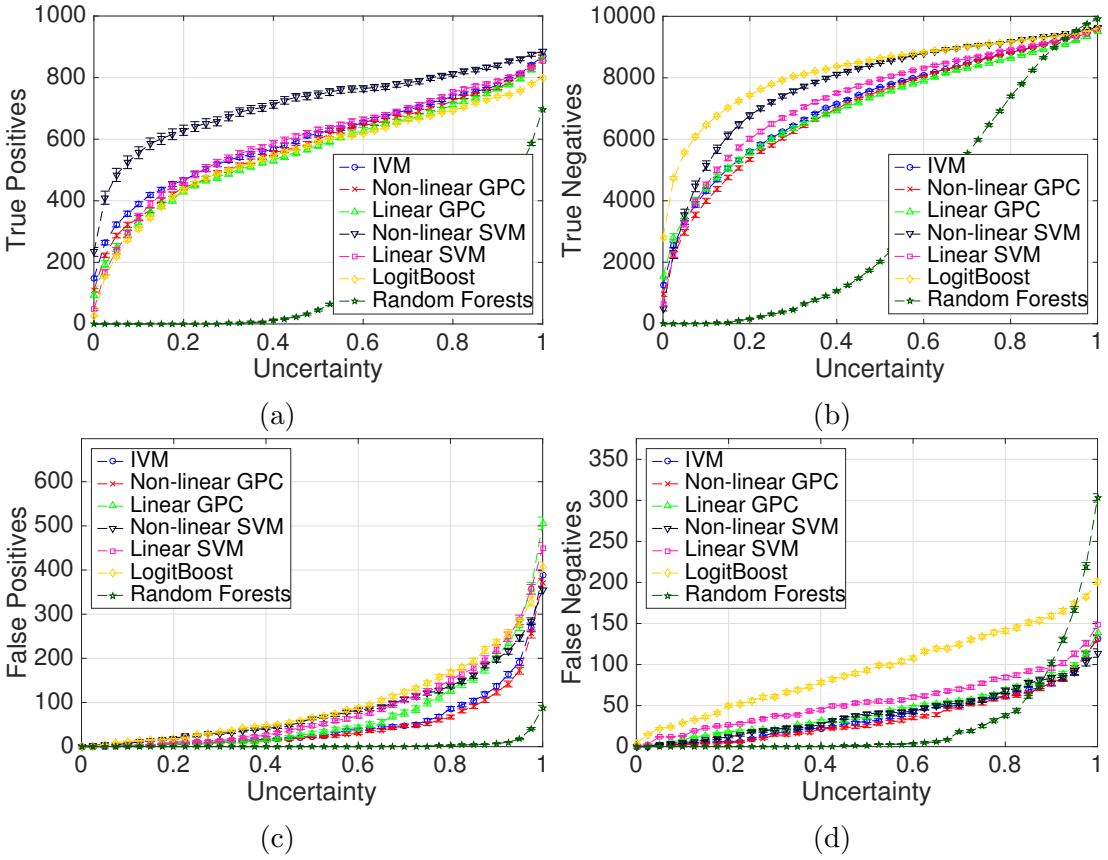


Figure 5.12: KITTI: Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalised entropy (uncertainty). The classifiers are trained on 200 and 500 instances of pedestrians and background respectively, and are tested on 2,000 and 5,000 of those classes. See the caption for Figure 5.10 for more detail. Note that on this particularly difficult data set, the difference between the classifiers is smaller than in Figures 5.10 and 5.11, although the LogitBoost is consistently confident in all graphs, and the random forest is consistently uncertain. The error bars indicate the standard error over 20 independent runs. (Best viewed in colour.)

graphs pertaining to *true* classifications (top row) and close to the bottom right of the graphs pertaining to *false* classifications (bottom row). This would correspond to making true classifications with low uncertainty (high confidence) and making incorrect decisions with high uncertainty. That in turn would allow us to correlate confidence with correctness.

Note that test points with a high classification uncertainty are not misclassified, but rather marked such that some other remedial action might be taken, for example obtaining label confirmation from a human or gathering otherwise additional data to aid disambiguation (e.g. Rosenthal et al. [2011]). This will indeed be a key component of our experiments in Chapter 6.

Examining first the true decisions (the top row of each figure), we notice that the single-discriminant classifiers are consistently more confident than the multi-discriminant classifiers. The LogitBoost is often the most confident, and the random forests are always the least confident by a large margin. In our desire to correlate confidence with correctness, we might initially consider this a strength of single-discriminant classifiers. However, we propose that the overall classification confidence of the classifier over the test set should reflect the difficulty of the problem. Introspection in robotics should, however, always focus on making mistakes with high uncertainty. The ability to disambiguate correct and incorrect decisions based on uncertainty is what leads to excellence in decision-making, but an introspective classifier applied to a difficult problem could be reasonably uncertain about most of its decisions.

Examining the false or incorrectly-made decisions (the bottom row of each figure), we see that for every data set, the non-linear GPCs are slightly more uncertain than the non-linear SVM, and equally the linear GPC is more uncertain than the linear SVM. Uncertainty in making errors is crucial in terms of introspection. Despite the differences being consistent, they are not large in magnitude. Secondly, in

the TLR data set the kernel appears to play a role, with the linear-kernel classifiers being less uncertain than the non-linear kernel classifiers. This is not true for DP and KITTI data sets, where all the GPC-based classifiers are more uncertain than the SVMs regardless of the kernel used. Thirdly, the LogitBoost is always most confident, and the random forest is always least confident. This is consistent with the correctly made decisions.

From the PR curves in Figure 5.9, the difficulties of the data sets clearly vary, with TLR being the easiest, followed by Daimler Pedestrian data set, and then KITTI being the most challenging. This is likely to be a result of the variation within the positive class paired with the modest number of positive exemplars in the training set (see Table 3.3). Ideally an introspective classifier would express a higher overall level of uncertainty for more difficult data sets. None of the classifiers seem to exhibit this behaviour.

We can compare these behaviours to those of the idealised classifiers described in Chapter 4. The equivalent plots are shown in Figure 5.13. Notice that the classifiers are tightly grouped in the true decisions, but exhibit much greater variation for the false decisions. This reinforces the idea that introspection is most concerned with uncertainty in incorrectly-made decisions. However, the *top hat* classifier is indeed the one which best allow us to relate confidence with correctness, by being the most confident for the true decisions and the least confident for the false decisions.

Comparing first the profiles of the bottom row of curves between Figure 5.13 and Figures 5.10, 5.11, and 5.12, we see that most real classifiers lie between the *top hat* and the *uniform* idealised classifiers. This supports our choice of idealised classifiers, with the LogitBoost and linear SVM demonstrating comparable introspection to the *uniform* classifier, which does not correlate confidence with correctness.

Examining the correct decisions, we see that the idealised classifiers are typically much more uncertain than the real classifiers (with the exception of the random for-

est). This discrepancy lies in the difference between $f(z)$ for our idealised classifiers (which is the uniform distribution) and the empirical density functions of our real classifiers, examples of which are shown in Figures D.1 and D.2. There we see that the empirical $f(z)$ is a heavily bimodal distribution around $z \approx 0$ and $z \approx 1$ for all classifiers save the random forest. The empirical distribution functions for the random forests resembles bell-shaped distributions which are much closer to $z = 0.5$ than the other classifiers. In order to reflect this tendency, to change $f(z)$ of our idealised classifiers to a bimodal distribution we could make the functions $f_1(z)$ and $f_2(z)$ more peaked, and decrease the mass near $z = 0.5$. This could also change the error profiles.

5.8 Conclusions

In this chapter we have introduced the idea that introspection could be achieved via an appropriate treatment of distance in feature space. We have examined a number of common classification frameworks, making predictions about how introspective they are likely to be based on their treatment of distance. In order to examine the effects of distance on non-stationary data, we have trained and tested those classifiers on synthetic data, exploring the effects of non-stationarity between the training and test distributions, the feature dimensionality, and the number of training data. From these experiments we deduce that the multi-discriminant classifiers are generally more uncertain in the face of unseen classes. We note that this is most visible when the dimensionality of the data is greater than 32 in our experiments with synthetic data.

We confirmed that these observations are also shown for the third-class experiments with real data for the GTSRB data set. The multi-discriminant classifiers tend to be more uncertain than the single-discriminant classifiers, and dimensional-

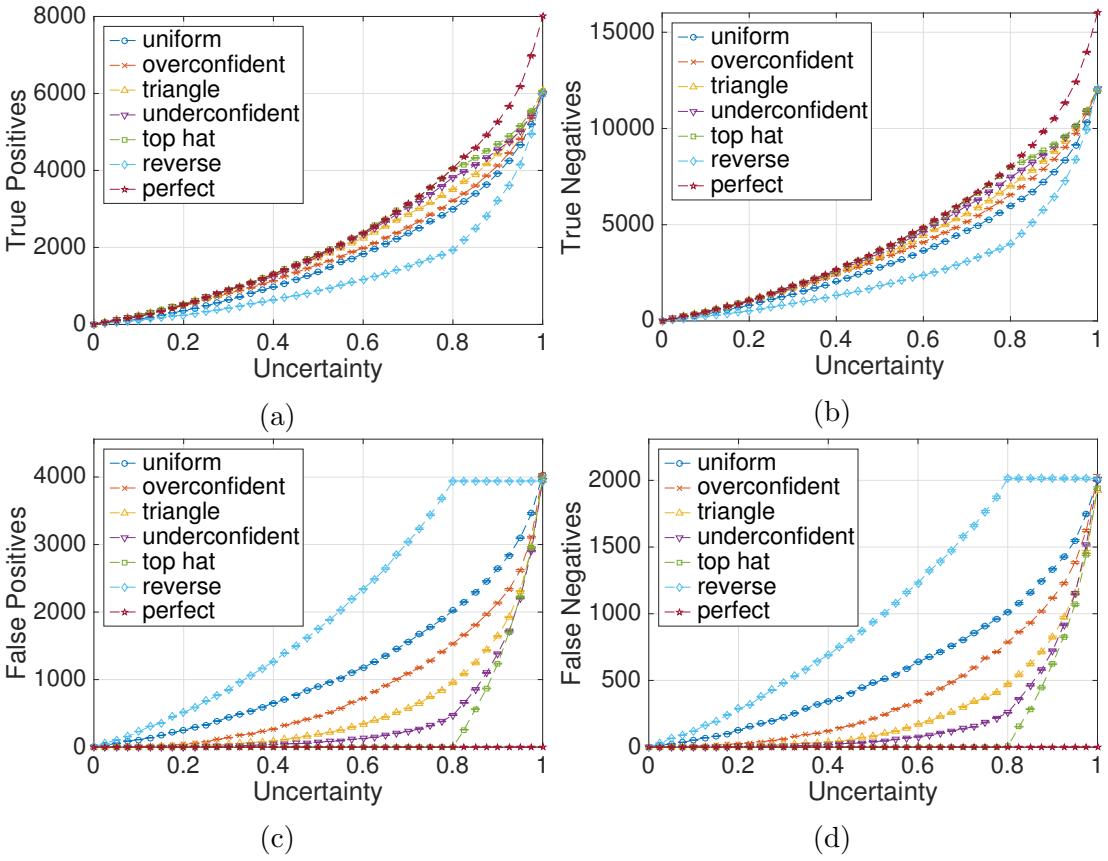


Figure 5.13: Idealised classifiers: cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against classification uncertainty. A more introspective classifier is one that simultaneously exhibits higher uncertainty when processing difficult instances (bottom right corner for false positives and negatives) and is more confident when it is correct (top left corner for true positives and negatives). Consequently, class decisions above a given uncertainty threshold are deferred since the output is deemed ambiguous. This is desirable since a single bad decision can have disastrous consequences. For each run, the classifiers generated 8,000 measurements from the positive class and 16,000 from the negative class. The error bars indicate the standard error over 20 independent runs. (Best viewed in colour.)

ity plays an important part.

In the detection experiment we have confirmed that indeed the multi-discriminant classifiers tend to be more uncertain than the single-discriminant classifiers in terms of incorrect decisions, as the third-class experiments suggested. However, more generally we can say that the multi-discriminant classifiers tend to be more uncertain about all decisions, true and false alike. We consider the uncertainty of incorrect decisions to be of paramount importance, but a simultaneous confidence in correct decisions is what allows us to reduce $p(e)$ (see (4.3)). Thus, we conclude that the multi-discriminant classifiers do exhibit a greater introspective capability, but the difference is not overwhelming so far.

We have investigated a variety of applications, feature types, class types and quantities, as well as the nuances between classification and detection. This is because it is not always possible to determine the introspective quality of a classification framework based on a single classifier and test set. Can we say whether a classifier which is uncertain about *all* decisions (like the random forest) is introspective or not? We suggest that it is perhaps introspective, but certainly not as useful as one which can also make correct classifications confidently. The overall uncertainty should reflect the difficulty of the data set. We have not seen evidence of this behaviour from any of our real classifiers.

In the next chapter, we apply the real classifiers to two decision-making scenarios. In the first scenario we examine the most uncertain classifications for each classifier, and apply some of them to active learning, where a human oracle labels some data chosen by the classifier. In the second scenario we examine the decisions made as a result of each classifier's most confident classifications. We apply large costs to particular outcomes and calculate the total cost of the decisions for each.

Chapter 6

Introspection in Decision Making

In Chapter 5 we investigated the ability of a number of real classification algorithms to correlate correctness with confidence. In this chapter we apply those classifiers to two decision-making scenarios, benchmarking their performance against the idealised classifiers motivated in Chapter 4.

The two decision-making scenarios we investigate represent the two ends of the uncertainty spectrum. The first is active learning, where the ideal behaviour will require mistakes to be made with high uncertainty. The second is decision making under large costs, where the ideal behaviour requires the most confident classifications to be made correctly.

In Section 6.2 we apply two classifiers to active learning. We allow the classifiers to choose a number of high-uncertainty queries from an unlabelled test set. These points are relabelled and added to the original training set. We attempt to determine which of the two classifiers chooses the best queries in terms of F-measure.

In Section 6.3 we examine the real classifiers' abilities to make decisions which risk incurring large losses, for instance the cost of a collision with a car or person. We illustrate the importance of introspection via the performance of the idealised classifiers, and in so doing, show the dissatisfactory nature of the real classifiers.

No real classifiers avoid costly errors in all data sets. We conclude the chapter by discussing the wider ramifications of introspection in robot decision making, and what can affect the introspective properties of these classifiers.

6.1 Making Errors with Uncertainty

Part of our requirement for introspection, correlating confidence with correctness, is to make errors with high uncertainty (see Section 4.1). In this section we show the extent to which this is true for our real classifiers.

In Figure 6.1 we examine the proportion of a classifier's errors which are made with high uncertainty by examining its error function, of which two examples are shown in Figure 6.1a. To do this we show the integral of the error function from $0.5 - w$ to $0.5 + w$ (where w is the half-length of the orange window shown in Figure 6.1b). This window can be thought of as an uncertainty window. We show the shape induced by a more introspective classifier in blue, and a less introspective classifier in dashed purple in Figure 6.1c. A classifier which makes its mistakes with high uncertainty will induce a curve which is close to the top-left of the figure. We will plot this integral for both the idealised classifiers and the real classifiers from their empirical distribution functions, clearly showing the variations between similar error functions.

Applying this to our real classifiers in Figure 6.2, we see the proportion of errors made as we increase the window half-length in Figure 6.1c. Note that this is a combination and re-parameterisation of the bottom rows of Figures 5.10, 5.11, and 5.12. This re-parameterisation amplifies the density of errors in the most uncertain classifications. Another crucial difference between these representations of the data is that Figure 6.2 is scaled to show the proportion of the total number of errors *for that classifier*, and so no benefit is shown for a classifier which makes fewer errors

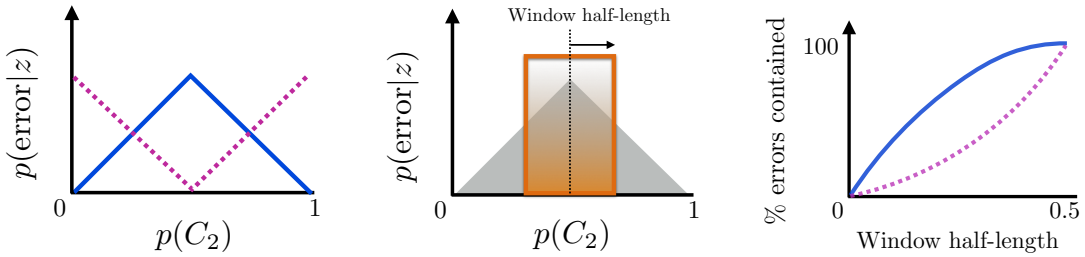


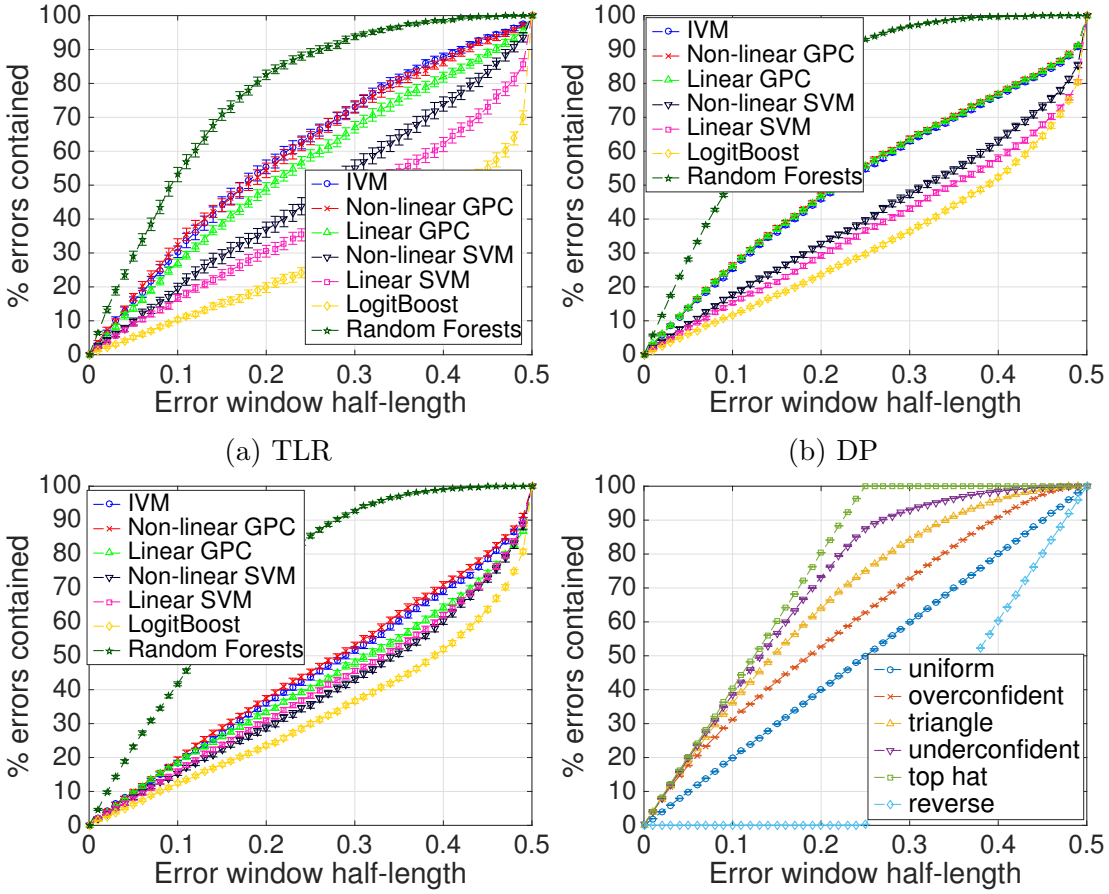
Figure 6.1: (a) A more introspective classifier (blue) will make most of its errors with high uncertainty, when $p(C_2)$ is near 0.5. Less introspective classifiers (dashed purple) will make errors with low uncertainty. (b) As we grow the orange window outward from the centre, we can calculate how many errors are contained for a particular distribution in (a). (c) We show the result of plotting the number of errors contained as we grow the orange window for the two idealised classifiers in (a). The blue (more introspective) classifier catches more errors when the box is small than for the dashed purple (less introspective) classifier. It also reaches steady-state because there are very few errors around $p(C_2) = 0$ and $p(C_2) = 1$, where the classifier is confident.

overall, it only shows the distribution of those errors over z.

First, we confirm that the random forest is much more uncertain than the other classifiers when it makes mistakes. Secondly, we confirm that the GPC-based classifiers outperform the SVMs and LogitBoost when it comes to being uncertain when they make mistakes. The KITTI data set, however, appears to be most difficult for the GPC-based classifiers.

In Figure 6.2d we show the same process as applied to our idealised classifiers. As we expect, the more introspective classifiers make more mistakes with high uncertainty, and so have a steeper initial gradient. They have been designed with this situation in mind, by making mistakes with high uncertainty.

The real classifiers range from being similar to the *under-confident* idealised classifier (in the case of the random forest) to being slightly worse than the *uniform* idealised classifier (for the LogitBoost, and occasionally the SVMs). The GPC-based classifiers are close to the *overconfident* idealised classifier for the TLR and DP data sets, and close to *uniform* for the more difficult KITTI data set. All the



(a) TLR (b) DP

(c) KITTI (d) Idealised classifiers

Figure 6.2: We show the proportion of errors made by a classifier contained within a region around $p(C_2) = 0.5$. The horizontal axis shows the size of that region, the orange window as described in Figure 6.1. To generate these curves, we randomly sample 1,000 positive and 1,000 negative test data and count the number of errors within a certain window. Note that a classifier which is uncertain when it makes mistakes will be closer to the top left of each plot. Benchmarking the real classifiers against the idealised classifiers, we see that generally they lie within the range of *under-confident* to slightly worse than *uniform*. We show the mean and standard error over 20 independent runs.

classifiers seem to perform least introspectively on the KITTI data set, exhibiting overconfidence when making errors. The KITTI data set is the most challenging by the traditional metrics, but ideally this would lead to an increase in overall uncertainty rather than overconfidence in errors as we see here.

Making mistakes with high uncertainty is crucial for active learning. Next we investigate the behaviours of two of our real classifiers on an active learning task.

6.2 Active Learning

In this section we concern ourselves with the classifications made with high uncertainty. An introspective classifier, which correlates truth with confidence, should harbour its potential errors in areas of high uncertainty. Therefore, we expect that if we label the test points in those high uncertainty areas and retrain the classifier, its performance should improve. This is the idea behind active learning: allow a classifier to choose some queries which are then labelled by a human (or algorithmic oracle) and fed back into the learner [Cohn et al., 1996]. In this section we are interested in *uncertainty sampling*, where the classifier chooses queries with maximum classification uncertainty.

Given our findings in Section 6.1, we would expect that the classifiers with more errors contained in the windows with smaller half-lengths will do better. However, it is important to note that if all classifications are made with high uncertainty, thresholding on uncertainty alone might catch many errors but also many true detections. The value of labelling true detections is expected to be less than labelling errors. Therefore, a classifier must correlate correctness with confidence in order to gain the most from the procedure.

Whereas in the later sections of this chapter there will be one action corresponding to each of the two classes that a datum may belong to, here we explicitly

introduce a third action: *help*. This *help* action is the classifier's declaration that, given its training data and resulting model, it is not qualified to take appropriate action on its own. Having humans label data sets is the status quo in fields such as semantic mapping (see Chapter 2), but it is an expensive task. Human operators may not be able to determine which subset of an unlabelled body of test data will be the most beneficial for a classifier if labelled, so we hope that by allowing the classifier to choose, fewer data need to be labelled for equal classification performance. The expectation for us, therefore, is that an enhanced introspective quality will allow a classifier to accurately discern which test data will be most useful, and thus have a larger performance boost relative to a non-introspective classifier.

The power of an active learning framework lies in its ability to select a suitable training set in an application-oriented way. It thus should inherently allow the system to adapt naturally to the non-stationarity of the data often encountered in long-term robotics applications, if the non-stationarity of the data reflects in a higher level of classification uncertainty.

After a literature review in Section 6.2.1, we test the relative increase in value of two classifiers, the IVM and non-linear SVM, based on test points they consider to be the most uncertain. We choose these classifiers because they represent both single and multi-discriminant classification frameworks, and are the best-calibrated of those considered in this thesis. The random forest is too uncertain to be discerning in active learning, and LogitBoost is too confident and would never select any points. Using the original training set together with each classifier's (different) selection of now-labelled points, we retrain both classifiers on both augmented training sets and examine their performances. We call this the *cross-over experiment*, and it is described in more detail in Section 6.2.2.

6.2.1 Related Works

A variety of methods have been proposed for data selection on the basis of information. One method is to consider disagreement within a committee of classifiers [Freund et al., 1997] as a criterion for active data selection. A similar approach has been applied to text classification [McCallum and Nigam, 1998]. More recently, Joshi et al. [2009] address multi-class image classification using SVMs and propose criteria based on entropy and best-versus-second-best (BvSB) measures (see Section 5.2) based on closeness to a hyperplane for determining uncertain points. Similarly to the work reported in this section, an active learning system using a GPC has been used for object categorisation, using the posterior mean and variance to estimate the test uncertainties. Those with the highest uncertainties are labelled and added to the training set [Kapoor et al., 2010].

More specific to robotics, active learning and directed information acquisition has received attention in recognition, planning and mapping tasks. Dima et al. [2004] aim to reduce the number of labelled training examples for outdoor terrain classification through the use of a kernel density estimator. The unlabelled data are sorted according to how 'surprising' they appear given the estimator, and the most surprising are hand-labelled and added to the training data. Verbal human-robot interaction is also a topic for active learning. Tellex et al. [2013] show that a robot can improve its understanding of the world based on confusing natural language utterance by using an information-theoretic approach to asking a specific question which resolves the confusion, using the new information to better understand the state of the world.

This section is based largely upon [Triebel et al., 2013], which applied active learning to a semantic mapping task. A characteristic of this work is that it introduced and demonstrated the benefits of *introspective* active learning. Note that the related work of Kapoor et al. [2010] also uses an inherently introspective classifier,

but its use is not explicitly motivated by its quality of introspection.

Next, we describe the active learning cross-over experiment we use to determine the value of the query selection for two classifiers.

6.2.2 The Cross-Over Experiment

We are investigating whether the use of a more introspective classifier leads to more informative questions being asked of the human expert. The information gain from retraining with a particular set of labelled questions can be expected to improve subsequent classification performance. In order to test this expectation we perform a cross-over experiment (see Figure 6.3 for a visual description) which starts by training an IVM and a non-linear SVM on the same data, 200 positive examples and 200 negative background examples. Then, 1,000 new data (with a class balance of 1:1, the same as during training) are shown to both classifiers for testing. For each classifier, the test examples with an uncertainty over 0.99 are manually labelled, up to a maximum of 50 examples. We do this to give each classifier the opportunity to express how many points are needed. In a scenario where classification is trivial, the classifiers might require no additional points. Because the classifiers can choose different examples, we form two new training sets: the 'IVM set' and the 'SVM set', which comprise the original training points together with that classifier's choice of manually labelled test examples. A new IVM and SVM are now trained on *each* of the two new training sets and evaluated on a further 1,000 new data points. This process thus gives rise to four classifiers: two IVMs trained on data selected by an IVM and a SVM respectively, and two equivalent SVMs. We compute the F-measure for all four classifiers (see Section 3.7). Then we must compare the two final IVMs against each other, the only difference between them being which queries were used for their training, and equally for the two final SVMs.

In Section 6.1 we found that neither the IVM nor the non-linear SVM make all

their mistakes with high uncertainty, although the IVM is superior in this regard. We therefore suspect a slightly larger increase in performance for classifiers trained on the IVM set than on the SVM set.

The results for each data set after 80 repetitions of this experiment are shown in Figures 6.4. As expected, if you consider the two final IVMs, the one trained on the IVM set performs better than the one trained on the SVM set. This, perhaps, is to be expected, but the same is true for the SVM: the final SVM trained on the IVM set performs better than the final SVM trained on the SVM set. In all three data sets, this increase is statistically significant to the 99% level, following a 1-sided paired t-test.

The uncertainty (normalised entropy) threshold of 0.99 means that we only re-label test points for which the scores $0.4412 < z < 0.5588$, which can be validated by visual inspection of Figure 5.1. This equates to a window half-length of 0.0588, indicating that we are only catching the errors in the very left-most region of Figure 6.2.

Although the increases are consistent, they are small. They are also in keeping with the fact that high-uncertainty behaviour of the IVM and SVM are most similar for the KITTI data set, and more distant in the other two.

In the next section we discuss the implications of these findings.

6.2.3 Discussion

Using the linear SVM as an example, the points with highest uncertainty will be those closest to the decision boundary, as shown in Figure B.1e. If we consider this SVM trained on linearly-inseparable data (see Figure B.2e), there are likely to be many points around the decision boundary, and we note that the position of that boundary is influenced by the amount of error incurred by the points in the soft margin. A test point lying on the boundary itself will yield a maximum uncertainty,

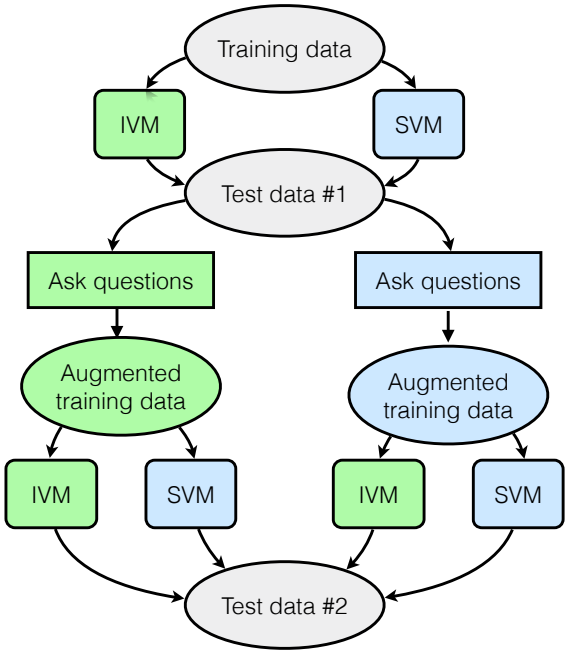
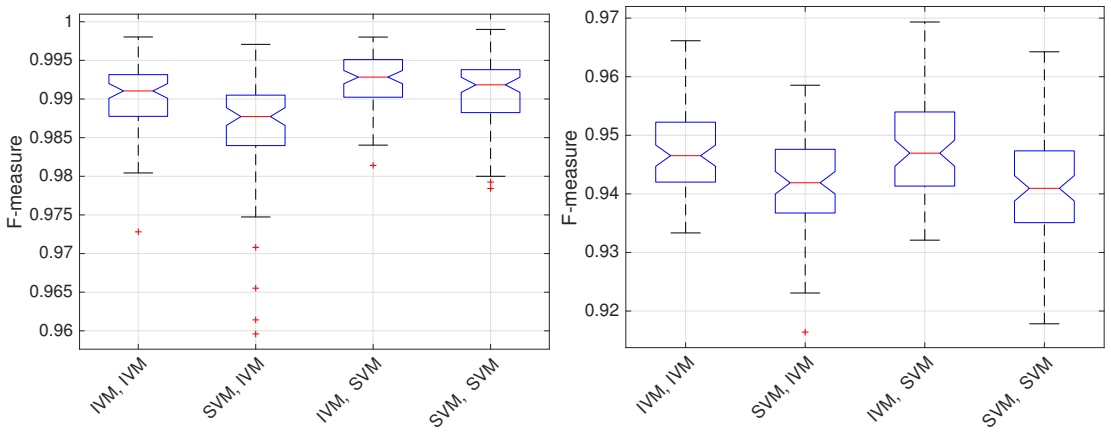
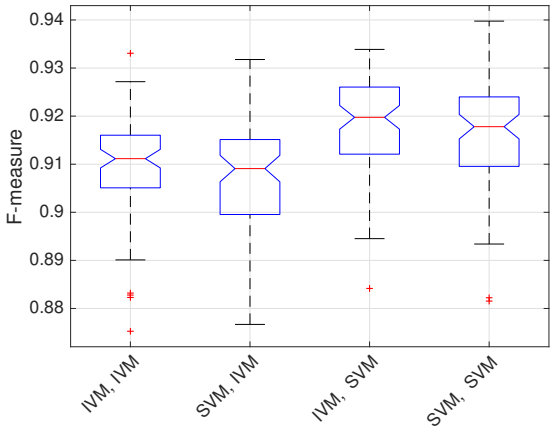


Figure 6.3: Here we show the procedure for the cross-over experiment, designed to test whether one classifier chooses points which do not only benefit itself in the next round, but are consistently more useful for the other type of classifier as well. We compare an IVM and an SVM, and choose the test points with highest normalised entropy to be labelled to augment the original training set.



(a) TLR

(b) DP



(c) KITTI

Figure 6.4: Data selected by the IVM lead to an improved learning rate in terms of F-measure for both an IVM and SVM over those selected by the SVM. The x-axis labels indicate which of the four classifiers from Figure 6.3 we are showing: 'IVM, SVM' indicates an SVM trained on the data set augmented by queries from the IVM. The box shows the interquartile range, the red line is the median, the whiskers show the range, and the red crosses are outliers as determined by their distance from the interquartile range. Here we show that classifiers trained on the IVM set (IVM, *) perform better than those trained on the SVM set (SVM, *). Results are shown for 80 experimental runs, and increases are significant to the 99% level.

but adding that training label will not incur a large misclassification error, and thus is unlikely to provoke a significant change in the position of the boundary upon retraining. The greatest change would be effected by a heavily mis-classified point, incurring a large training error. This is the aim of *hard negative mining* [Dalal and Triggs, 2005], but requires a labelled test set. In the case of a linear decision boundary such as the SVM, this unlabelled but misclassified test point would be labelled with confidence, and thus not chosen by the active learning algorithm. Therein lies the strength of an introspective classifier: the misclassified point far from the decision boundary has a chance of being labelled with high uncertainty if it lies far from any training data, and thus it may be chosen as a query and significantly affect the position of the decision boundary. However, it is also desirable for points close to the decision boundary to be classified with high uncertainty, so ideally we seek a manner to distinguish between the two: one point close to the decision boundary, and another far away from both the decision boundary and any training data. We propose that perhaps the latter is preferable when dealing with active learning queries. It is possible that the use of the GPC's posterior mean and variance could be used to distinguish between the two. Uncertainty alone will not permit this distinction. We suggest that this is another strength of multi-discriminant classifiers, which are able to provide a predictive variance. This is a promising avenue for further research.

One issue with uncertainty sampling is that it is *myopic*, that is, in each iteration we select the k most uncertain test data without considering their information content relative to each other. They may be very similar, in which case selecting different data may be more informative. There exist non-myopic solutions, such as GLASSES [González et al., 2016], which approximates what is called the 'look-ahead loss function', which might allow us to select k points offer the most information gain *as a collective*.

Classifiers with different models will not necessarily benefit equally from the same queries, and that benefit is difficult to predict. We have noticed that the non-linear SVM tends to have a slight class bias in terms of its probability contours, even when the training data are normally distributed with the same standard deviation. This tendency is suggested in Figures B.1d and B.2d, where we can see that the space far away from the data is not at exactly $z = 0.5$. In the case of this experiment, we might therefore expect the IVM to query points which are far away from the training data slightly more than the SVM. These more distant queries may contribute to the increase in performance.

One concern is that a classifier which asks for the maximum number of queries is likely to create a training set which benefits the classifiers by virtue of quantity and not quality. We suggest that the desired behaviour would be to ask an appropriate number of questions, given how generally competent the classifier is at a particular task. We might hope for that number of questions to decrease as the size of the initial training set increases. This is another avenue for further work.

6.3 Decision Making with Costs

In Sections 6.1 we examined one half of the introspective coin: which classifiers made mistakes with high uncertainty, and how that can be of use. In this section we examine the other half, that is whether the most confident classifications also tend to be true. We do this in the context of making decisions where the costs of particular outcomes are significantly imbalanced.

We compare the real classifiers with the idealised classifiers in a scenario similar to the 'robot crossing the road' problem discussed in Chapter 4. The robot must choose to *go* or *wait* depending on what the classifier tells it about whether the road is clear or not. The cost of *going* when the path is blocked incurs a collision and with

it a large cost. The cost of *waiting* unnecessarily, however, incurs only a small cost. In Section 6.3.1 we discuss the classical theory of decision making with costs. In Section 6.3.2 we consider an alternative manner of choosing a probability-threshold, by calibrating it given a test set. Finally we apply the classical decision making theory to a real scenario in Section 6.3.3.

6.3.1 Bayesian Decision Theory

Autonomous robots typically have a set of actions at their disposal, with varying degrees of appropriateness in particular situations. The difficulty lies in estimating which action is most appropriate when there is uncertainty about the state of the world. Following standard decision theory (e.g. LaValle [2006]), we calculate the expected loss of performing a particular action when we have a distribution representing the likelihood for each state of the world $(p(C_1), p(C_2), \ldots, p(C_{|\mathcal{C}|}))$, defined as:

$$\mathbb{E}[L(a)] = \sum_{i=1}^{|\mathcal{C}|} L(a, C_i)p(C_i), \tag{6.1}$$

where $L(a, C_i)$ is the cost or loss associated with each potential outcome. We then choose to perform the action a which minimises this expected loss. There are many ways in which to choose the values of the loss function $L(a, C_i)$. A user might estimate the financial or time cost or the effort involved, or the utility to a patient [Pauker and Kassirer, 1980], and use those values. A user might prefer to limit a particular outcome to occurring, say, once every N tests.

In robotics it is common to use a classifier to estimate the likelihood of the state of the world $(p(C_1), p(C_2), \ldots, p(C_{|\mathcal{C}|}))$. As we have established earlier in this chapter, many of the classifiers considered in this thesis are overconfident, and thus provide uncertainty estimates of varying quality. The question we wish to answer is: which classifiers give estimates which allow us to make decisions which capture

the priorities imposed by our loss function?

For our classical decision-making experiments we revisit the 'robot crossing the road' problem discussed in Chapter 4. To summarise, there are two states the world can be in: either there is an object in the way (C_2, e.g. a pedestrian, car, or traffic light), or there is not (C_1). There are also two available actions $a \in \{\text{wait}, \text{go}\}$. We wish our robot to *wait* if there is an object in its path, or *go* if the way is clear. In the case of autonomous driving it is sensible to associate a very high cost to performing the *go* action when there is in fact an object in the way (C_2), resulting in a collision, and a lesser cost to performing the action *wait* when the path is clear (C_1), resulting in an unnecessary delay. While inefficient, this false positive error is more desirable than running a red light or colliding with another vehicle.

In the case of driver assistance systems (e.g. automatic emergency braking) the costs are reversed: the loss associated with a false positive (an un-necessary emergency stop) is very large, and a false negative (a missed opportunity to perform an emergency stop) is a less undesirable outcome.

In Figure 6.5a we show the expected losses of the two actions when there is equal cost associated with each type of error. We can see that the intersection between the two lines occurs at $p(C_1) = p(C_2) = 0.5$. Therefore, by (6.1) the robot should stop if $p(C_2) > 0.5$ and go otherwise. As we increase the cost of a false negative (performing the *go* action when there is a person, C_2), the range of detection probabilities $p(C_2)$ which result in a *go* action reduces, as seen in Figure 6.5b. These actions are optimal if the probabilities $p(C_i)$ are correct, but since we are using estimates of the probability there is no guarantee that the actions will be optimal. Since an introspective classifier is uncertain when it makes mistakes, these high-uncertainty errors will be close to $p(C_2) = 0.5$. When the costs are imbalanced, those errors will largely be subsumed by the *wait* action, such as in Figure 6.5b. A less introspective classifier will make more mistakes near the extremes of z and so

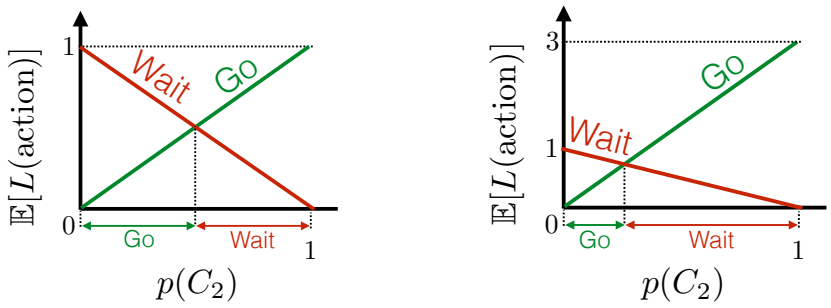


Figure 6.5: (a) We have set equal cost to a false positive (take the *wait* action when there is, in fact, no person: C_1) and the false negative (take the *go* action when there is a person: C_2). The expected losses $\mathbb{E}[L(\text{a})]$ from the two actions meet at $p(C_1) = p(C_2) = 0.5$, and we choose the action which minimises the expected loss. (b) We have made the cost of a false negative three times the cost of a false positive, which reduces the probability region for which we choose the *go* action. By increasing the cost of accidentally hitting a pedestrian, we are trying to create a more cautious system, which will take the *wait* action more of the time.

more of those errors will occur in the *go* region, resulting in a greater prevalence of very expensive errors.

Thus, ideally, as we make the cost of a false negative much greater than that of a false positive, our classifiers should become more and more cautious, employing the safer *wait* action when the test datum is challenging, and incurring less overall cost. Crucially, this relies upon the assumption that most of a classifier's mistakes lie in the middle of the probability spectrum. The *top hat* classifier is designed with this in mind. The threshold $T = 0.25$ (which occurs when the cost of a false negative error is 3) will result in all expensive potential false negative errors being avoided by *waiting*. For all higher cost ratios, no false negative errors can occur.

It may be tempting to tune the costs in order to steer the robot towards the 'desired' behaviour. Instead, we perhaps ought to focus on whether the costs are appropriate, and allow the decision theory to enact a behaviour which is true to that cost function. This is only possible if the classifier is introspective.

6.3.2 Relating Costs to ϵ-bounds

By defining the loss matrix and adopting the decision-making system in Section 6.3.1, we are choosing the decision threshold T by the process illustrated in Figure 6.5. We then choose the *go* action for all measurements below the threshold, and *wait* for those above it. As we stated in Section 6.3.1, this assumes that the measurement is a probability of class, and therefore does not take into account the non-ideal characteristics of a real classifier.

More pragmatically, we might instead apply a classifier to a labelled test set and choose a threshold which results in the best decisions given the tendencies of that particular classifier. For instance, we could determine an acceptable probability that any decision results in, say, a false negative error. This probability is ϵ in Section 4.1. The threshold T is then determined by

$$T = F_2^{-1}(\epsilon), \tag{6.2}$$

where $F_2(z)$ is the empirical cumulative distribution function of z given that $c = C_2$, and $F_2^{-1}(x)$ is its inverse. This method is, in certain situations, likely to produce better decision-making than the classical pipeline described in Section 6.3.1. However, this perpetuates the assumption that the training and test data are stationary, because we are choosing a threshold based on a classifier applied to a a particular test set. If that test set is not representative of future test sets, the choice of threshold will result in the future violation of the ϵ-bound. Both problem-specific and classifier-specific thresholds will result in poor decisions if the classifier is not introspective in its ability to deal with unseen test data. As a result, in this thesis we have chosen to focus on problem-specific thresholds, and describe the alternative strategy here.

6.3.3 Experiments

In Section 6.3.1 we discussed the importance of the loss function $L(a, C_i)$ and how it shapes the decision of which action a to choose, given a distribution of the state of the environment $(p(C_1), \ldots, p(C_{|\mathcal{C}|}))$. Here, we apply those principles to the classifiers we trained in Section 5.7, and examine the costs incurred, comparing them to the idealised classifiers. The more introspective idealised classifiers, designed with decision-making in mind (in Section 4.1), will perform well in this task.

We seek classification frameworks which allow our robots to make decisions which are faithful to the loss function. For instance, if we make the cost associated with a particular outcome very large, then the actions which can lead to that outcome should be chosen more infrequently, or at least only when the classifier gives a very confident estimate of the state of the environment. We characterise the 'appropriateness' of a classifier's decisions by comparing the total cost incurred when it is employed as part of the decision-making pipeline. We vary the ratio of the costs of false negative and false positive outcomes. The ideal behaviour is to incur a consistently low overall cost at any particular cost ratio.

We use the measurements from the classifiers as input to the decision-making system, and evaluate the decisions made. We set the costs of true positive and true negative outcomes as 0, and the cost of a false positive outcome as 1. The cost for the last outcome, the false negative or missed pedestrian, is varied from 1 to 10^7. Mission-critical decisions are likely to come with high costs associated with particular outcomes, so we need our classifiers to be consistent across a wide range of high costs. We plot two things: (a) the number of true outcomes (both positive and negative together), and (b) the total cost of all the decisions made. (b) is effectively a function of (a), weighted by the costs.

In Figure 6.6 we show these graphs for the idealised classifiers. We sample measurements from $f_1(z)$ and $f_2(z)$ and apply them to the decision-making system.

As we increase the cost of a false negative error, we see that the more introspective classifiers (a) make the greater total number of true decisions, while eventually becoming very conservative, and (b) incur lower total cost at any cost. Notice that for these classifiers, performing well in terms of costs corresponds to a high number of true decisions. This is expected because they have been designed with decision-making specifically in mind.

These pairs of graphs demonstrate the trade-off between classifiers which avoid catastrophic decisions, and those which might be so cautious that they never take the higher-risk action. The left-hand graphs demonstrate the rate at which the classifiers' decisions become more and more cautious as the cost of a false negative increases. On the right-hand graphs, the ideal is for a curve to be as low as possible (close to the horizontal axis) at every cost ratio. This would represent the classifier which makes better decisions given any particular cost ratio.

In Figure 6.7 we show the results for the real classifiers. Firstly, we see that the classifiers are more spread in terms of true decisions than the idealised classifiers. The confidences of the decisions varies substantially, from the random forests to LogitBoost. Unlike for the idealised classifiers, performing confidently in the left-hand graph does not guarantee good performance in the right-hand graph. In fact, the overconfidence of making many true decisions in the face of high costs is apparently associated with catastrophic errors. We do this to demonstrate the fact that making the most true decisions given a particular set of costs is not sufficient for mission-critical decision making processes in robotics. Secondly, we see that the classifiers with non-linear kernels appear to perform better than those with linear kernels. In Grimmett et al. [2015b] we see that no classifiers perform safely across all three data sets, with the non-linear SVM causing catastrophic decisions in the DP data set. In that publication, we train the same number of data as in these results, but test on 8,000 positive examples per run, with 10 runs (rather than 2,000 positives and 20

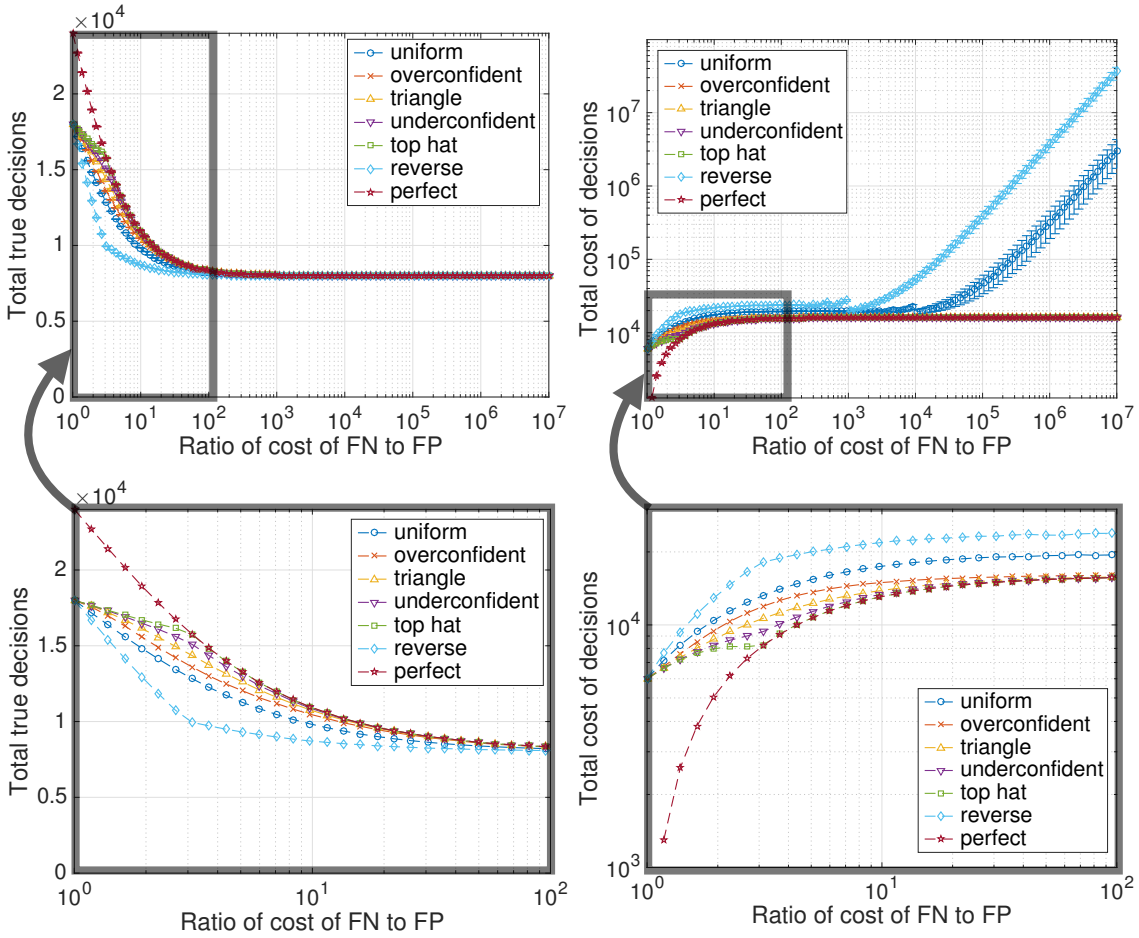


Figure 6.6: Decision making for the idealised classifiers. The horizontal axes represent the cost of a false negative (FN) error, while the cost of a false positive (FP) error is 1. On the left we show the number of correct decisions made (positive and negative), and on the right we show the total cost of all decisions (correct decisions carry 0 cost). Ideal behaviour is to incur minimal total cost at every point on the horizontal axis. The error bars indicate the standard error of the mean, from 20 runs each comprising 8,000 samples from the positive class and 16,000 from the negative class. The second row of graphs shows the magnification of the top row without error bars.

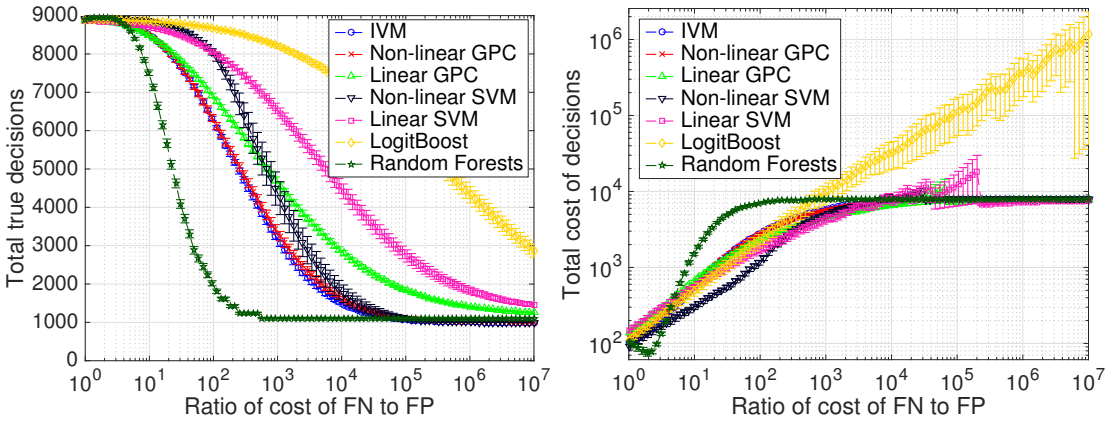
runs, as seen here). This failure of the non-linear SVM is visible due to an increase of possible false negatives at testing. In Figure 6.7, the non-linear SVM, IVM, and non-linear GPC perform well on two of three data sets, but generate some dangerous errors in the KITTI data set. In summary of both sets of results, no classifier is capable of avoiding all catastrophic errors in all three data sets. This implies that they are all overconfident to some extent.

Note that the right-hand graphs are very unforgiving of high-confidence errors. The spike in the linear GPC in Figure 6.7c is the result of a single error over the 20 runs, each with 1,000 possible false negatives. However, when one single error costs 10^6 more than any other, it must be avoided.

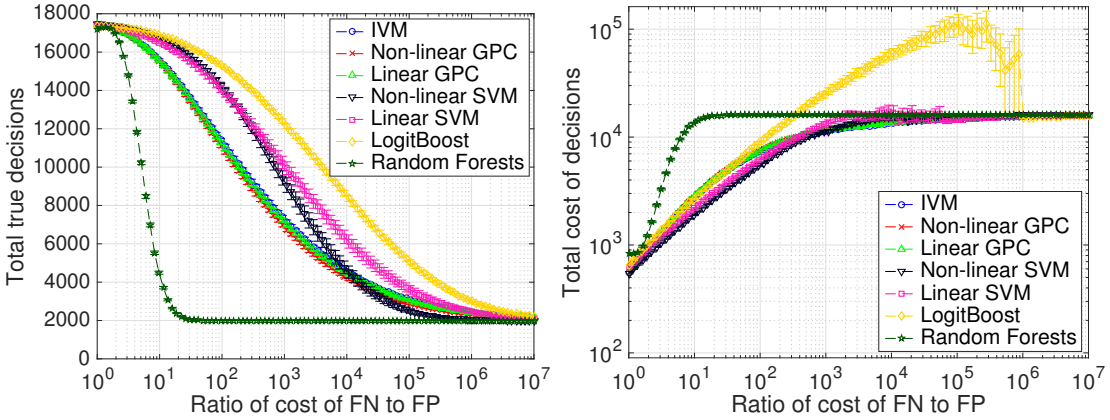
We cannot conclude that the multi-discriminant classifiers benchmarked here avoid more catastrophic errors than the single-discriminant classifiers. The difference in introspection between the classifiers is not sufficient to highlight a preference here, perhaps due to an insufficient degree of non-stationarity in the data. The test samples are drawn from separate, but visually similar video sequences. If the weather or time of day were varying, we may see the effects we expect given the previous experiments in this thesis.

6.4 Conclusions

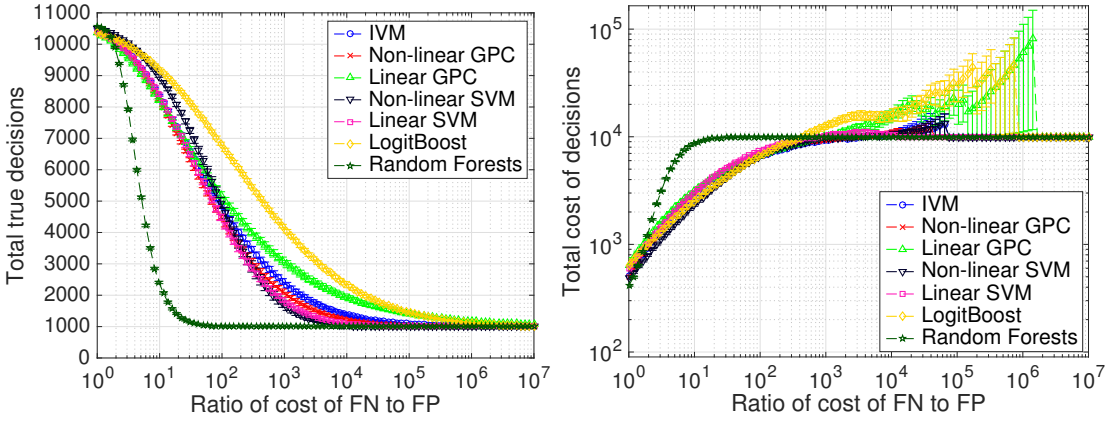
In this chapter we benchmarked the real classifiers against the idealisations in the contexts of active learning and high-confidence decision-making. In the active learning experiment, we saw that the multi-discriminant classifiers are more uncertain than the single-discriminant classifiers when making mistakes, and that the IVM asked questions which were of more benefit than the non-linear SVM in terms of F-measure. The benefit is small but consistent. In the high-confidence decision-making experiment, we see that all the real classifiers are overconfident and capable



(a) TLR - 1,000 positive and 8,000 negative examples



(b) DP - 2,000 positive and 16,000 negative examples



(c) KITTI - 1,000 positive and 10,000 negative examples

Figure 6.7: High-confidence decision making. The error bars indicate the standard error of the mean over 20 independent runs. See the caption of Figure 6.6 for more information.

of making expensive errors. We see that there a large improvement will be required until they behave with the caution of of the top-four idealised classifiers (*top hat*, *under-confident*, *triangle*, and *overconfident*).

In both contexts, introspection enables good decision making. In the active learning experiment, a large proportion of the *top hat* classifier's mistakes are made near $z = 0.5$ and so a small window will capture significant errors. These errors, once labelled, will likely result in a larger classifier improvement than for the other idealised classifiers. In the high-cost scenario, the *top hat* classifier maintains the highest number of true decisions for a given cost ratio, inducing the lowest total cost (excluding the *perfect* classifier). The cost ratio corresponding to a threshold of $T = 0.25$ results in the selection of the *wait* action for all the potential false negative errors. For all higher cost ratios it performs equally to the *perfect* classifier. The way it could be improved is to reduce the region around $z = 0.5$ in which it makes errors.

We propose that while we have seen differences in the third class experiments, the data sets we are using in this chapter are not significantly non-stationary to demonstrate the effects to the same extent. Applying them to data sets which are more different between training and testing is left as further work.

The work in this chapter demonstrates how performance metrics traditionally used in machine learning for classifier training and evaluation may be insufficient to characterise system performance in a robotics context, where a single misjudgement can have disastrous consequences.

We have designed an experiment which highlights unlikely events. The expensive mistakes in Figure 6.7 are often as the result of only a handful of errors over tens of thousands of opportunities for expensive mistakes. Regardless of the small total number of mistakes, we consider them to be unacceptable. With larger labelled data sets, it would be possible to more accurately estimate the frequency of such catas-

trophic errors. We expect the number of mistakes to decrease with an increase in training data, but that the non-stationary nature of the data sets precludes complete prevention of these mistakes.

Having examined a multitude of one-off decisions, we wish to see the effects of introspection in sequential decision making. When an agent must aggregate a series of measurements in order to maintain a state estimate, the differences between the classifiers will be apparent. We explore this in the next chapter.

Chapter 7

Introspection in Sequential Decision Making

In Chapter 6 we investigated the effects of applying classical decision-making to the outputs of real classifiers, whose measurements were used to make one-off decisions. In this chapter we allow the decision-making module (or *planning* module) to *learn* how a classifier behaves, allowing it to make more informed inferences given a measurement. However, we then present that planning module with measurements generated from classifiers exhibiting *different* behaviours from those learned, demonstrating the importance of measurement consistency, a tendency not shown by non-introspective classifiers in the face of non-stationary test data.

The measurements in Chapter 6 were used to make one-off decisions, by weighing up potential outcomes. In robotics it is common for decisions to follow from each other, for instance an agent choosing the next driving direction based on an estimate of its current position, and for errors in judgement to accumulate over time. In this chapter we apply the idealised classifiers to a sequential planning problem modelled by a POMDP, or Partially Observable Markov Decision process. This allows us to teach an agent to navigate across standard POMDP benchmark problems while re-

ceiving information about the environment according to its noisy sensor or classifier. Specifically, we compare agent efficiency given a particular idealised classifier. We will see that the characteristics for success are different to those in in Chapter 6: that introspection is sufficient but not necessary for good decision-making in this context. It turns out that the entropy of the probability density functions is more relevant than correlating confidence with correctness. That said, an introspective classifier will still achieve the best result.

However, we restate the importance of consistent behaviour in non-stationary data streams. Recall that for a classifier to be introspective, it needs not only to correlate confidence with correctness, but to do so consistently in the face of previously unseen data. We demonstrate that a change in the sensor characteristics without compensation by the decision-making system will result in bad decisions, even if that change is to a classifier which *increasingly* correlates confidence with correctness or provides more informative measurements. This consistency requirement to our sensors is paramount. We restate that if an autonomous vehicle enters a novel environment and its classifier returns specious measurements, the resulting decisions will be poor.

In the interests of thematic continuity, we present basic MDP and POMDP theory in Appendix E. In Section 7.1 we detail how we apply the concept of a classifier to the POMDP. In Section 7.2 we outline the related work in the field. In Section 7.3 we describe the two example scenarios which will be used for testing. In Section 7.4 we discuss the relevance of entropy to the efficiency with which agents solve the tasks, which is one of our main contributions. In Section 7.5 we present the results of the experiments, and our conclusions are discussed in Section 7.6.

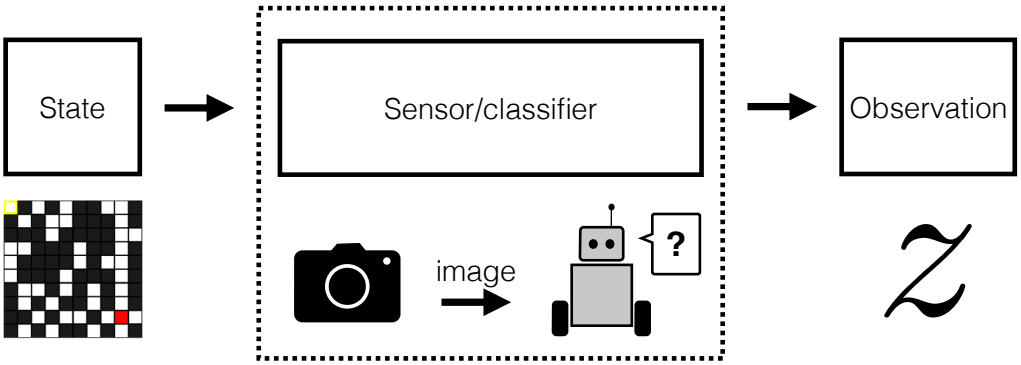


Figure 7.1: Here we model the sensor or classifier as the process which maps the agent state to an observation z. In the context of Chapter 6 it encompasses the physical processes mapping the world through a camera via an image to the measurement.

7.1 The Observation Probability Function as a Classifier

In the background section in Appendix E we have only considered the measurement as a function of the agent's true state: $p(z \mid a, s')$. Let us assume that all actions elicit measurements with the same distribution, so that $p(z \mid a, s') = p(z \mid s')$. This is the true model of the sensor. This might represent the mapping from the agent's position in a particular world (the state) to an observation, as in the grid world scenario described in Section 7.3.1. In a more practical case, and indeed in the context of Chapter 6, it might represent the physical processes which map, say, the lighting and geometry of a scene, via a camera, via a classifier, before finally yielding an observation, e.g. $p(\text{car}) = 0.8$. However, here we do away with the complexity of the image and simply consider a sensor (or classifier, we use the terms interchangeably in this chapter) to take a state s and return an observation z, as shown in Figure 7.1.

So far in this thesis we have considered the observation or measurement coming from a classifier as a probability in the range $[0, 1]$. While there exist formulations of the POMDP framework that incorporate continuous observations (e.g. [Hoey and

Poupart, 2005]), we choose to deal with this in a simple manner by discretising the observation space $[0, 1]$ (the output of the classifier) into eight equally sized bins. As a result, the POMDP models use a fairly crude approximation to the probability density functions and error functions. On the other hand, the computational complexity grows exponentially with the number of bins, and the number used here is sufficient to demonstrate the effects in which we are interested. We experimented with the number of bins, and found that there was a negligible effect in performance in both scenarios by increasing the number of bins from eight to twelve or sixteen, but a huge increase in computation time.

Consider an agent as possessing an onboard sensor, and that the observation probability function is the agent's model of how that sensor behaves. That model is going to be based on one of the idealised classifiers from Section 4.3. This onboard sensor, like the idealised classifiers, returns a measurement of some binary aspect of the world. Recall that these observations are discrete rather than continuous, with 8 possible values. Therefore, for each state s we can use the discrete probability density functions $p(z \mid c = C_1)$ and $p(z \mid c = C_2)$ to calculate $p(z \mid s)$. These density functions are approximations to the probability density functions $f_1(z)$ and $f_2(z)$ used throughout this thesis. Thus, there is one observation probability function per idealised classifier. This is the agent's *model of the sensor*.

In order to generate measurements during the live running of an agent through a test scenario, we perform inverse transform sampling given the cumulative distribution functions, as described in Section 4.3.1.

After a literature review, we describe the two problem scenarios used for the rest of the chapter in Section 7.3. In both we consider the observation probability distribution $p(z \mid s)$ to represent a classifier.

7.2 Related Works

The concept of POMDPs in robotics dates back to the 1970's [Sondik, 1971], but had little impact in the robotics community until a resurgence in the 2000's due to the development of point-based approximations. These permit useful solutions to problems with up to a few thousand states, when previously the algorithms and computational power limited POMDP use to applications with a dozen or so states. POMDPs have been used in a variety of situations, including but not limited to the following. Spaan and Vlassis [2004] present a point-based value iteration algorithm which is effective in planning trajectories for delivering mail around an office environment. Outside robotics, Hauskrecht and Fraser [2000] apply POMDPs to medical therapy planning for patients with ischemic heart disease, reducing the problem's complexity by combining dynamic programming and decision tree techniques. The agent must choose between various tests and treatments in order to correctly diagnose and manage the therapy of a patient. Atrash et al. [2009] perform real-time tracking of the dialogue state between a wheelchair user and his or her wheelchair, allowing it to move the user around using voice commands. If the wheelchair detects an error in understanding, it can ask specific queries for the user to clarify certain parts of the instruction. Hsiao et al. [2007] apply POMDPs to grasping tasks for robot manipulators. The manipulator has a tactile sensor on it, and the pose and shape of the target object are unknown.

El Ghaoui and Nilim [2005] and Bertuccelli and How [2008] are both concerned with MDPs with uncertainty in the state transition matrix. In these cases the state is fully observable.

Jaulmes et al. [2005] use active learning to address the problem of estimating noise in the state transition and observation probability models. They present two formulations. The first adds a *query* action, which incurs a large cost but returns the

true state of the agent. They also add an extra state feature per uncertain parameter in the model, which takes one of a number of discrete values, indicating constraints in the likelihood of particular state transitions. They also vary the probability of the sensor giving the correct solution, giving results for $p(\text{correct}) = \{0.7, 0.8, 0.9\}$. They apply this to the so-called tiger problem [Kaelbling et al., 1998], and show that the agent manages to learn the observation probabilities even when the cost of *query* is so high that it is never chosen. This ability to learn is attributed to the use of the *listen* action, which can be seen as a noisy and cheap alternative to *query*. The authors present a second formulation which is less computationally expensive, in which state-action pairs where either the state transition or observation probability functions are uncertain are modelled by Dirichlet distributions. The agent samples a number of POMDP models using the Dirichlet distribution, and then takes an action and receives a measurement. Then it can decide whether to obtain the true state from the oracle, in which case the Dirichlet parameters are updated. These two steps are iterated until there is sufficient knowledge in the distribution over models. New models are sampled, and the least probable existing models are removed. This formulation is applied to the tiger problem again and learns the correct parameters after 200 to 300 queries, and achieves the optimal reward after 2,000 queries.

Oliehoek et al. [2008] also sample POMDP models, this time using the cross-entropy (CE) method to estimate the value of a particular policy for a multi-agent POMDP (or Dec-POMDP). They maintain a distribution over models, and drawing samples from that distribution, select the most valuable and use them to update the distribution, much like in Jaulmes et al. [2005] but for policies rather than individual parameters of the state transition or observation distributions.

7.3 Test Scenarios

In this section we motivate and detail the two test scenarios used for evaluation in this chapter. Shani et al. [2013] provide a summary of six commonly-used POMDP benchmark problem scenarios. Each benchmark differs from the others, but they can be grouped into two classes: one where an agent navigates around a known map where the state is partially observable seeking a goal, and another where the agent's position is fully observable but the map is unknown at the start and only partially observable. We therefore consider a scenario in each of these two classes of problem in order to examine the effects of the observation models. The two scenarios we have chosen, a generic *grid world* scenario and a *wumpus* scenario [Russell and Norvig, 2003], are both appropriate for our analysis because the observations can be made stochastic and the thing being measured is binary. The essence of the popular RockSample problem [Smith and Simmons, 2004] is captured by the analysis of the grid world and wumpus problems, because it is a hybrid of the two: the map is known (like the grid world scenario), and there are multiple objects within the map, some of which yielding positive and some negative reward (like the wumpus scenario). We consider the grid world and wumpus problems to be representative of the body of benchmarks, and hence are the ones we use to demonstrate the idealised classifiers.

7.3.1 Grid World

An agent moves around a world represented by a discrete, two-dimensional, square grid of given dimensions. The agent's goal is to navigate to the exit (marked as the red square in Figure 7.2). At each time step, the agent must choose one of four actions $a = \{\text{up}, \text{down}, \text{left}, \text{right}\}$, and after each move it receives a sensor measurement z to help it update its belief state, which is a distribution over positions

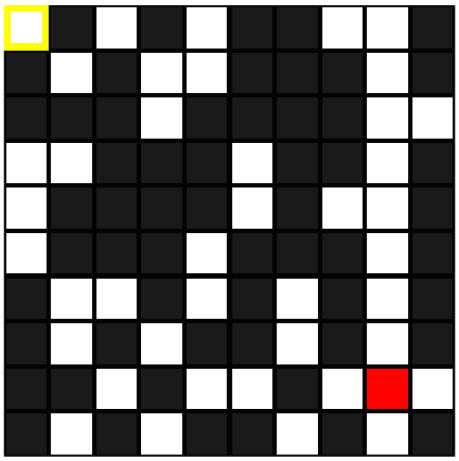


Figure 7.2: Here we show a two-dimensional grid world in which each square is randomly black or white with $p = 0.5$. The robot starts in the yellow square in the top left, and must make its way to the red goal near the bottom right hand corner as efficiently as possible by estimating its current state and choosing appropriate actions at each time step.

in the grid. However, the state transition function is stochastic; the chosen action may not yield its desired result: if, for example, $a =$ right it could skid and end up moving in a different direction or fail to move entirely. The state in this scenario is an index into the agent's position in the world. The belief state is thus a distribution over the grid locations.

This scenario is episodic, meaning that if the agent occupies the same square as the exit, it is placed in an absorbing or terminal state (i.e. it stays in that state forever). At the start of the next episode, the agent restarts in the top left corner of the grid world. The reward function defined over states is 0 everywhere except for the goal, which gives reward 1.

The agent has a pre-determined number of sensors which give a reading for the colour of a pattern of squares adjacent to its current position; for instance it may have a sensor returning the colour of the currently occupied square, and another returning the colour of the square immediately above it. These two measurements are independent given the map. Each sensor will return a measurement z between 0 and 1, where 0 corresponds to a black square and 1 corresponds to a white square.

In the classical POMDP formulation there is one single observation per time step, so we consider each possible combination of the individual sensor outputs to be an observation. As a result, the observation space is of size $(N_b^{N_s})$, where N_b is the number of bins used to discretise the otherwise continuous sensor space, and N_s is the number of sensors. If our agent has three sensors and eight bins per sensor, each observation takes one of 512 possible values. With a slight abuse of notation, we do not distinguish between a measurement from a single sensor and the index into these potential values: both are referred to as z.

For each experiment we start by defining the necessary functions listed in E.2 and generate a policy using SARSOP [Kurniawati et al., 2008]. The method is outlined in Appendix E.2 and is terminated after a fixed period of time. We choose the period of time experimentally by increasing it until the performance increases become small.

The state transition distribution function is determined via the following property: once a desired move has been chosen by the policy, there is a distribution over whether the agent ends up relative to the desired move.

$$p(\text{agent moves 1 square in desired direction}) = 0.4, \tag{7.1}$$

$$p(\text{agent moves 2 squares in desired direction}) = 0.3, \tag{7.2}$$

$$p(\text{agent moves 1 square to the left of desired direction}) = 0.1, \tag{7.3}$$

$$p(\text{agent moves 1 square to the right of desired direction}) = 0.1, \tag{7.4}$$

$$p(\text{agent skids and does not move from current square}) = 0.1, \tag{7.5}$$

so for instance if the agent is in some square and it chooses to move right, it will move right once with $p = 0.4$, right twice with $p = 0.3$, down with $p = 0.1$, up with $p = 0.1$, and stay still with $p = 0.1$. Note that if any of the moves could take the agent off the boundaries of the map, the probability of making that move is

transferred to the skid outcome.

7.3.2 Wumpus World

For this scenario we use a wumpus problem [Russell and Norvig, 2003] in which an agent also navigates around a grid map, but the configuration of the map is not known at the start, and thus the state comprises the layout of the map along with the agent's position. The map is once again a rectangular grid, one cell of which contains gold, and some others contain obstacles. The goal of the agent is to *grab* (one of the available actions) the gold, while avoiding the deadly wumpus. The world has the following properties: squares adjacent to a wumpus are smelly, and the square containing the gold glisters. The agent has two sensors which measure the smelliness and glister of the agent's currently occupied square. The agent receives a measurement from each after it performs an action from the set {up, down, left, right, grab}. The state space is the number of possible worlds for a fixed size $m \times n$ which do not violate the following constraints: the agent will always start in the top-left corner and this square can be occupied by neither the gold nor a wumpus; they must occupy distinct squares and do not move. Here we differentiate the 'map', comprising the unmoving gold and wumpusses, from a 'world', which for this scenario is equivalent to a state and includes the position of the agent as well as the unmoving map.

The state transition distribution is deterministic with a known starting position (the top-left square). The reward function is as follows: every move has a reward of -1; grabbing the gold is terminal and yields a reward of $+1000$; and entering a square containing a wumpus is terminal and yields a reward of -1000. The agent can choose to move into the boundary of the world, but it will stay in the same place.

For example, we first generate a map and place the agent in the top-left square,

as shown in Figure 7.3. This particular agent has a deterministic observation probability distribution (for the sake of illustration, although this will generally not be the case later in the chapter), and knows that it is in a 3×3 map containing exactly one wumpus and one pile of gold. In Figure 7.4 we show the belief distribution become progressively less uniform as the agent navigates around the map and its sensors give it information about its surroundings. Figure 7.4a is a list of all possible starting worlds, and the initial belief is a uniform distribution over these. The agent's first action is *right* which deterministically moves the agent one square to the right. It then receives a measurement $z_1 = \{\overline{\text{smell}}, \overline{\text{glister}}\}$ (where $\overline{A}$ means "not A"). As a result of the move and the measurement, the belief distribution is adjusted to reflect the agent's change of position (such that the agent's position is correct) and incorporate the new information gained from the measurement (by (E.18)). From the agent's new position, there is no smell, therefore there is no wumpus either in the top-right corner or in the middle square, and those maps are no longer plausible, so the belief over those worlds becomes zero. All remaining worlds are equally plausible because of the deterministic observation probability distribution function in this example. As the agent traverses the map, it narrows down the positions of the wumpus (which is deduced at $t = 4$) and the gold, eventually entering a glistering square at $t = 9$, prompting it to choose the grab action next, yielding $+1000$ reward, with a total reward of $-8 + 1000 = 992$. Note that the observation probability distribution is deterministic here for the sake of the example, and that this function will later be replaced by a distribution function generated using our idealised classifiers described in the next section.

The most costly part of running the experiment is generating a policy. That policy is then valid for any map of a predetermined size with a predetermined number of wumpusses. As with the grid world, the policy is generated using the SARSOP algorithm [Kurniawati et al., 2008] (outlined in Appendix E.2) that is terminated

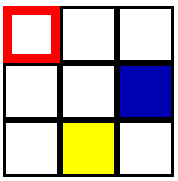


Figure 7.3: This is the real starting world for our worked example. The agent occupies the square bordered red, there is one wumpus in the dark blue square, and the gold is in the yellow square. The goal is for the agent to *grab* the gold. Figure 7.4 shows the agent's belief space contract to the solution as it safely navigates around the map.

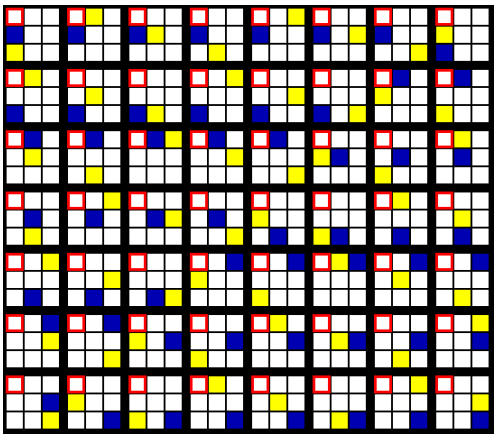
after a fixed period of time. In the small problems, we choose the period of time experimentally by increasing it until the performance increases become small. This is standard practice. At test time we start by randomly selecting a world from the set of valid starting worlds of a the appropriate size, and allow each agent (one per idealised classifier) to explore it until they reach an absorbing state. For each experiment we choose a number of starting worlds and a number of runs per world, which are detailed in the caption for each figure.

7.4 Entropy

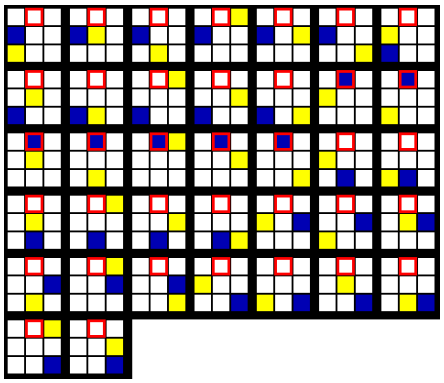
The choice of a classifier's probability density functions affects the information value of a measurement. In order to examine a classifier's information value, we can calculate the Shannon entropy of its density functions, which encapsulates a measure of information content. However, Shannon entropy is applicable to discrete distributions, while ours are continuous. Therefore, we consider the continuous entropy of a density function [Cover and Thomas, 2006], defined as

$$h(f) = -\int_0^1 f(z) \log_b f(z) dz, \tag{7.6}$$

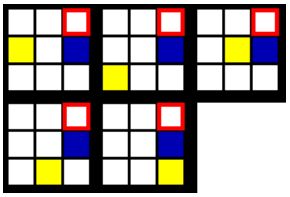
where b is the chosen base of the logarithm. For these experiments we choose to represent information in nats.



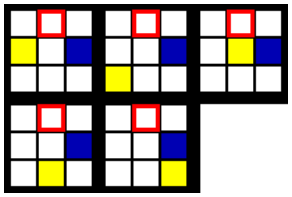
(a) $t = 1$, $a_1 = \rightarrow$



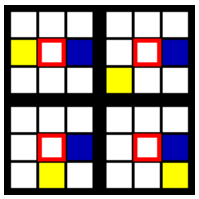
(b) $t = 2$, $z_1 = \{\overline{\text{smell}}, \overline{\text{glister}}\}$, $a_2 = \rightarrow$



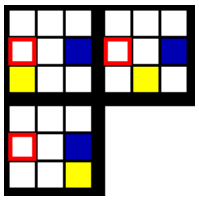
(c) $t = 3$, $z_2 = \{\text{smell}, \overline{\text{glister}}\}$, $a_3 = \leftarrow$



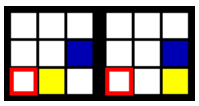
(d) $t = 4$, $z_3 = \{\overline{\text{smell}}, \overline{\text{glister}}\}$, $a_4 = \downarrow$



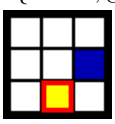
(e) $t = 5$, $z_4 = \{\text{smell}, \overline{\text{glister}}\}$, $a_5 = \leftarrow$



(f) $t = 6$, $z_5 = \{\overline{\text{smell}}, \overline{\text{glister}}\}$, $a_6 = \downarrow$



(g) $t = 7$, $z_6 = \{\overline{\text{smell}}, \overline{\text{glister}}\}$, $a_7 = \rightarrow$



(h) $t = 8$, $z_7 = \{\overline{\text{smell}}, \text{glister}\}$, $a_8 = \text{grab}$

Figure 7.4: We show the plausible states (or worlds) as the agent navigates through the true 3×3 map shown in Figure 7.3. This can also be thought of as the *belief state* at each time step, where each world is equally likely. The square containing the agent is bordered red, the wumpus (to be avoided) is dark blue, and the gold (to be sought after) is yellow. The agent chooses an action $a_t \in \{\uparrow, \downarrow, \leftarrow, \rightarrow, \text{grab}\}$. Immediately after each action is completed, the agent receives a sensor measurement $z_t \in \{\text{smelly, glistery, both, neither}\}$; smells indicating the nearby (adjacent) presence of a wumpus, and glister indicating that the agent's square contains gold. As the agent chooses actions and receives measurements, the list of possible worlds narrows until finally it deduces the true map, and grabs the gold. In this example, the observation probability function is deterministic and always gives the correct value.

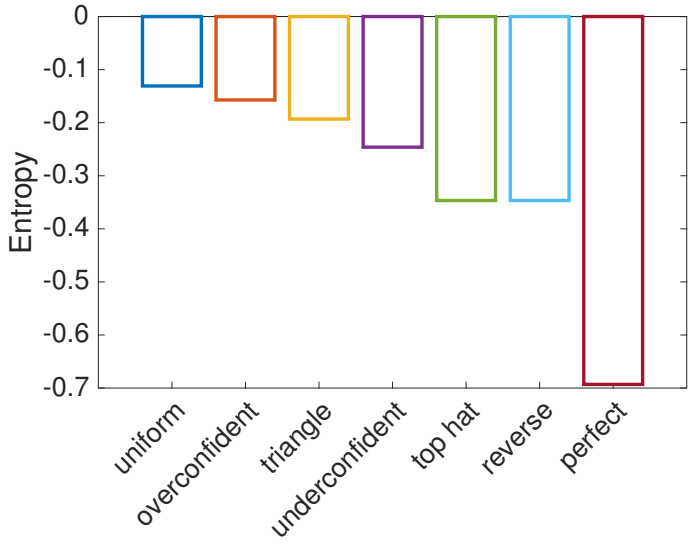


Figure 7.5: Here we show the continuous Shannon entropies of the probability density functions shown in Figure 4.5. A larger negative entropy is more desirable because it indicates a greater information content. Note that the entropies of the $f_1(z)$ functions are the same as those of the $f_2(z)$ functions because they are mirror images of each other (see Figure 4.5).

In Figure 7.5 we show $h(f_1(z))$ for each idealised classifier. We would like the measurement z to contain information about the class of the test datum, and that information will increase with $-h(f_1(z))$. Therefore, we see that a measurement from the *uniform* classifier contains the least information, that the *top hat* and *reverse* classifiers carry the same information, and that the *perfect* classifier is the most informative.

The entropy of a density function is invariant to rearrangements of the horizontal axis. Therefore, because $f_1(z)$ is a mirror image of $f_2(z)$ for each classifier, their entropies are the same. This is further demonstrated by the fact that both the *top hat* and *reverse* classifiers have the same entropy. Intuitively, and especially following the results from Chapter 6, we might expect the *reverse* classifier to do poorly. However, this is not the case in the POMDP framework, because the mapping from input z to class C_i is explicitly stated for each classifier. The classical decision-making system described in Section 6.3.1 assumes a probability, and with that assumption comes

the expectation of an introspective classifier. In the POMDP framework, rather than a correlation between correctness and confidence, we require a consistent mapping from measurement to class. That is to say, the classifier providing the measurement must do so predictably throughout its operation. This mapping is explicitly stated in the observation probability function.

One of the key findings of this chapter is that the information content of the classifiers is the most important factor when comparing them in sequential decision making, and that a more informative sensor allows a robot to achieve its goal more efficiently, both in terms of number of steps taken and total reward.

7.5 Experiments

The explanation of how the classifiers can be applied to the POMDP framework in Section 7.1 is somewhat simplified. In reality, there are three places in the sequential decision-making pipeline which make use of the observation probability distribution and the probability density functions from Chapter 4:

(A) The sensor model used for the policy generation, specifically the term $p(z \mid a, b)$ in Equation (E.19),

(B) The true sensor model, which maps the agent's true state to the measurement: the cumulative distribution function $F_i^{-1}(x)$ in Equation (4.18), and

(C) The agent's decision-making system, or method of interpreting a measurement and updating its belief state: the $p(z \mid a, s')$ term in Equation (E.18).

These are illustrated in Figure 7.6. In this section we apply various combinations of the idealised classifiers to these three locations to explore the importance of consistency in non-stationarity and the information content of the measurements.

In Section 7.5.1 we illustrate the importance of consistency to introspection. We simulate a situation in which the decision-making system (C) expects measurements from a particular sensor 4.18, but the characteristics of that sensor have changed. This disparity between expectation and reality characterises the non-introspective lack of consistency in non-stationary data streams. Will our robots be able to cope with inconsistent sensors? The answer is no, even if the sensor 'improves', becoming one that better correlates confidence with correctness, or provides more informative measurements.

In Section 7.5.2 we demonstrate the improvement achieved when using sensors which maintain the same characteristics in the face of potentially non-stationary data. We show the solution efficiency if (B) and (C) agree with each other, so the sensor measurements are in keeping with what the decision-making system expects. In doing so, we also isolate the effects of changing (A), the classifier used to generate the policy.

In Section 7.5.3 we examine the effects of varying the world size. In this experiment, the sensor provides measurements consistent with the expectations of the decision-making system, so (B) and (C) are the same. Having already analysed the effects of (A), we fix this at the start.

Finally, in Section 7.5.4 we investigate the effect of varying the number of sensors available to the agent traversing the grid world, again with a consistent pairing between the sensor and decision-making system.

We carry out each of these experiments on both the grid world and wumpus scenarios wherever possible (we cannot experiment with the number of sensors in the wumpus scenario), reporting the average number of steps to completion for the former, and the average reward along with the average number of steps for the latter.

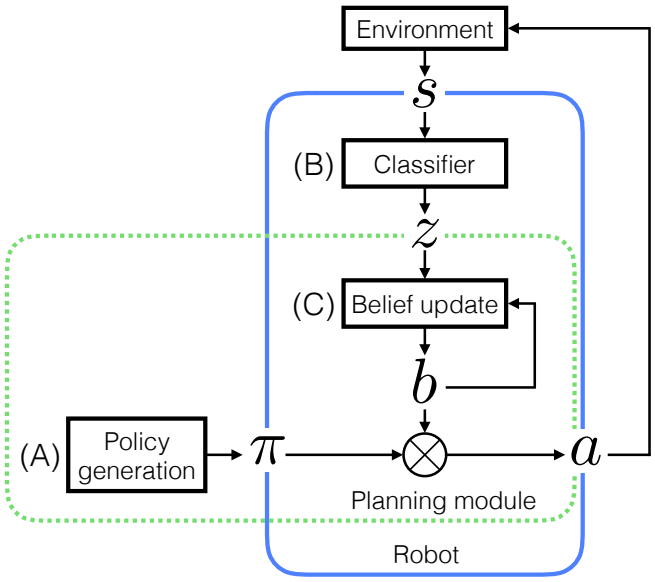


Figure 7.6: These are the three places in the sequential decision making pipeline that make use of the observation probability distribution (or sensor models). (A) The first is used for the *policy generation*, specifically in Equation (E.19). (B) The second is the true sensor model or *classifier*, mapping from state to measurement in Equation (4.18), and (C) the third is the *belief update*: the agent's interpretation of that measurement in Equation (E.18).

7.5.1 The Ill Effects of Non-stationarity

In this experiment we simulate the effect of a disjunction between the sensor measurements and the decision-making system. This is exactly the situation we observed in the high-cost decision-making experiment in Chapter 6. The classical decision-making system expects probabilities, but receives overconfident measurements, and the resulting decisions occasionally results in catastrophe.

To do this, we fix the policy ((A), see Figure 7.6) and test every pairwise combination of (B) and (C), the sensor's true model and the agent's model of the sensor. This forces the decision-making system to misinterpret the measurement given by the classifier, which could be the result of using a non-introspective classifier, in that it does not behave consistently in the face of unknown test data.

Note that it is not possible to carry out this experiment for all combinations of the idealised classifiers. It is only possible for the agent to model the sensor

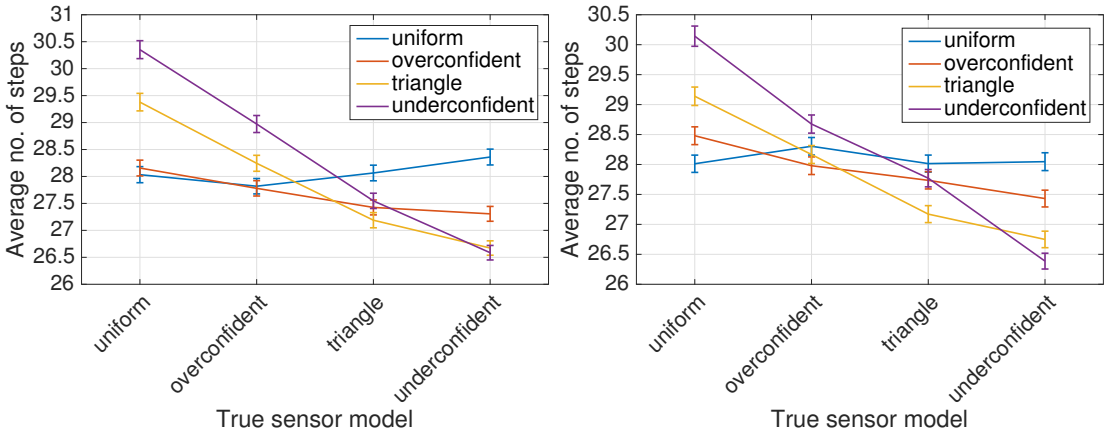
as those which are capable of generating measurements across the entire range of Z, namely the *uniform*, *overconfident*, *triangle*, and *under-confident* classifiers. An agent incorrectly modelling the sensor as *top hat*, *reverse*, or *perfect* will create logical impossibilities and lead to an invalid belief state.

For example, suppose that the agent is in a region of the grid world in which its current location and all the squares accessible within the next move are black. Also imagine that it is in a known starting state, so the belief distribution is zero everywhere except for the agent's true state. The agent, believing the sensor to be *perfect*, chooses a move and receives an observation. If the observation is in fact drawn from, say, a *uniform* sensor, and it happens to be an error (with $p = 0.25$), it may report $z > 0.5$, saying that the agent's new square is white. As we have stated, it is not possible for the agent to have moved from its starting square to a white square within one move. The denominator of the belief update equation (E.18) will be zero for all s', making the new belief b' infinity everywhere, which is not a valid belief distribution. This outcome is possible for any combination for which the range of measurements from (B) is not contained within the range of (C).

Grid World

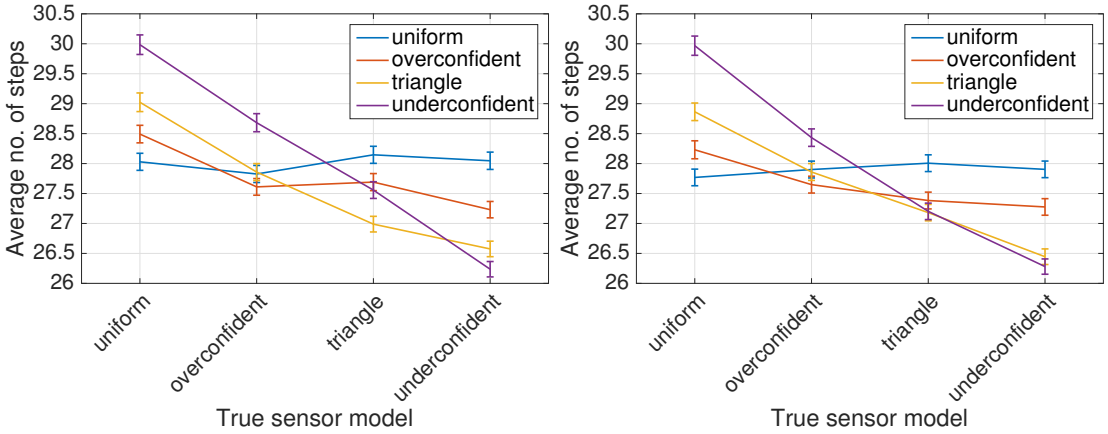
In Figure 7.7 we show the number of steps to completion for every combination of (A), (B), and (C). Firstly, it is always best for the agent's model of the sensor to match the true sensor model. This is to be expected because the agent chooses actions based on the fact that a measurement carries a certain probability of error with it, and thus the belief state evolves as a function of that assumption. If that assumption is incorrect, the belief state will be altered in a way that misrepresents the agent's true state, and so it is likely to choose a suboptimal action and take longer to reach the goal.

Also apparent from Figure 7.7 is that if the agent's model of the sensor matches



(a) Uniform policy

(b) Overconfident policy



(c) Triangle policy

(d) Under-confident policy

Figure 7.7: Grid world: here we analyse the effects of using a sensor which does not behave consistently in the face of non-stationary data. We fix (A), the policy, and test every pairwise combination of (B) and (C), the sensor's true model and the agent's model of the sensor. Along the horizontal axis lies the true sensor model: the measurements are generated using these models. The vertical axis shows the average number of steps to termination, and each curve shows an agent expecting measurements from a sensor corresponding to the legend. Error bars represent the standard error over 10,000 independent tests over a single 10×10 map.

the true sensor model, it is best for them to be the *under-confident* sensor, and worst for them to be the *uniform* sensor. This is the introspective case we will examine in more detail in the next experiment. All non-matched pairings could be the result of the use of non-introspective classifiers.

Thirdly, the four graphs are almost identical, so the choice of policy seems not to affect the number of steps required to reach the goal.

Note that the blue *uniform* curve is approximately horizontal in all the graphs. Each agent can be thought of as receiving two pieces of information from any single measurement: whether $z < 0.5$ (the square is black, or not) and; within that interval, the degree of confidence in that assertion (a.k.a. the error rate). An agent modelling the sensor as *uniform* is blind to the second piece of information, seeing all measurements $z < 0.5$ as equally informative (because $f_{1,\text{uniform}}(0 < z < 0.5) = 1.5$), and the same for $z > 0.5$. Therefore, because each sensor has an equal chance of showing $z < 0.5$ as $z > 0.5$, the *uniform* agent cannot distinguish between the true sensor models. This results in an invariance to the true sensor model and hence a near-horizontal curve.

This also explains why the purple *under-confident* curves are the steepest. The agent or decision-making system which models the sensor as *under-confident* makes the strongest assumption about the information content of a measurement. If it is correct about the sensor's true model, it stands much to gain. However, if the true sensor model is *uniform*, it will have too much faith in measurements near $z = 0$ and $z = 1$, and bias the belief distribution away from the true state.

By expecting the worst and modelling the sensor as the less-informative *uniform* model, the agent guarantees a baseline of performance: it will reach the exit in approximately 28 steps. This is the safe choice, because it bounds the worst possible performance. However, it is possible to solve the problem more efficiently if we correctly model the sensor as more informative. By assuming a greater level of

information, we stand to gain performance, but if the sensor becomes less informative than we are expecting, we risk losing by the same amount. This is analogous to using a classical decision-making system with non-introspective classifiers. Equation 6.1 assumes introspection by using the term $p(C_i)$. If the classifier measurement does not behave like a probability, the decision-making process is assuming information which it does not have. This is what leads to catastrophic decisions in Chapter 6.

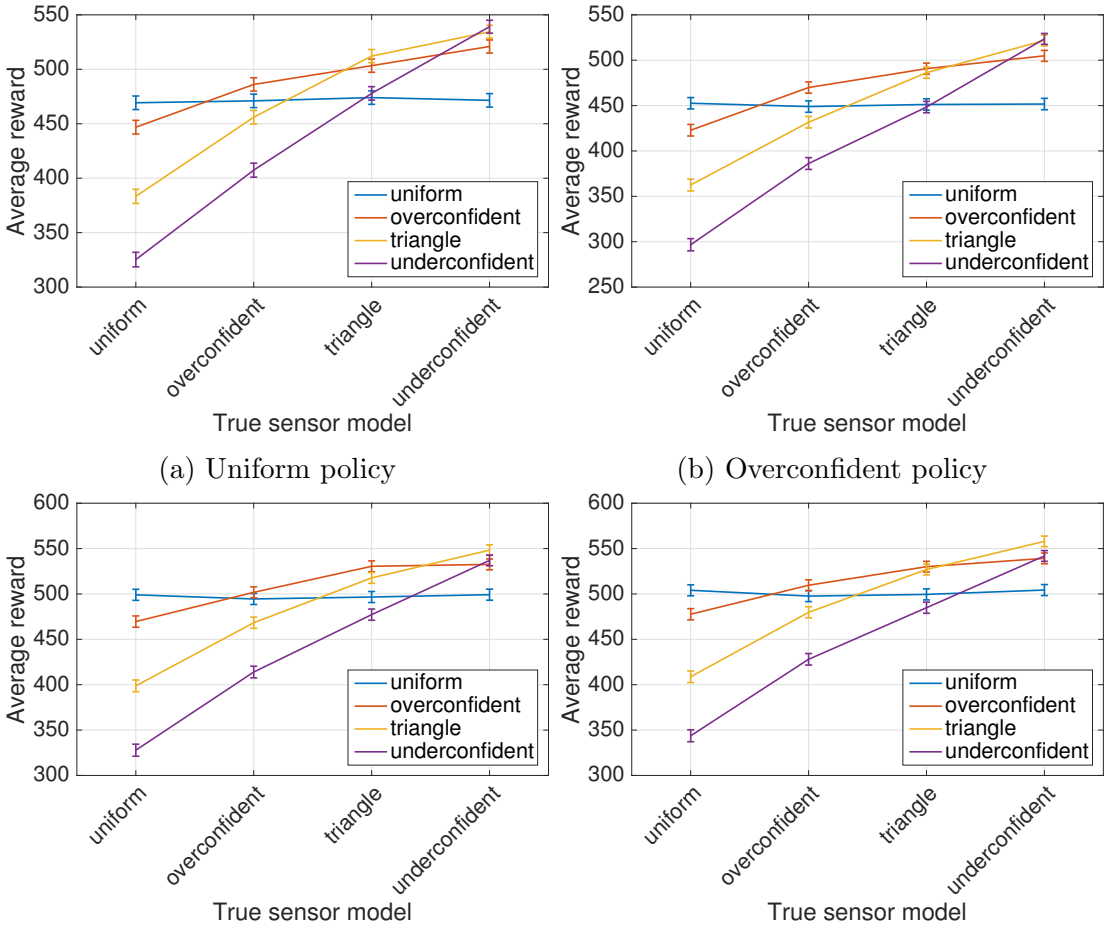
Wumpus

The results for the wumpus scenario are shown in Figure 7.8. Note that a better result corresponds to being towards the top of the vertical axis, unlike the grid world figure. Once more, the blue *uniform* curves are very close to horizontal, and the *under-confident* curves are the steepest. Unlike in the grid world results, (i) the choice of which model to use for policy generation affects the rewards, (ii) the tendency of the best choice of the agent's model of the sensor being to match the true sensor model is less clear, and (iii) we stand to lose more than we gain by modelling the sensor as informative, given the available choices.

(i) We see that there is a tendency towards more informative policy models, yielding higher rewards rewards. Here, the *triangle* and *under-confident* policies yield higher rewards than the *uniform* and *overconfident* policies. We will explore this more explicitly in the next experiment in Section 7.5.2.

(ii) The agent that correctly models its sensor performs very well, but not always best. When the true sensor is *under-confident*, we achieve similar performance by modelling it as *overconfident*, *triangle*, or *under-confident*. We suggest that this is related to (iii).

(iii) We see that we stand more to lose by incorrectly expecting a more informative sensor than we do to gain by correctly expecting it. By examining the average number of steps to completion in Figure 7.9, we see the opposite tendency: that we



(a) Uniform policy

(b) Overconfident policy

(c) Triangle policy

(d) Under-confident policy

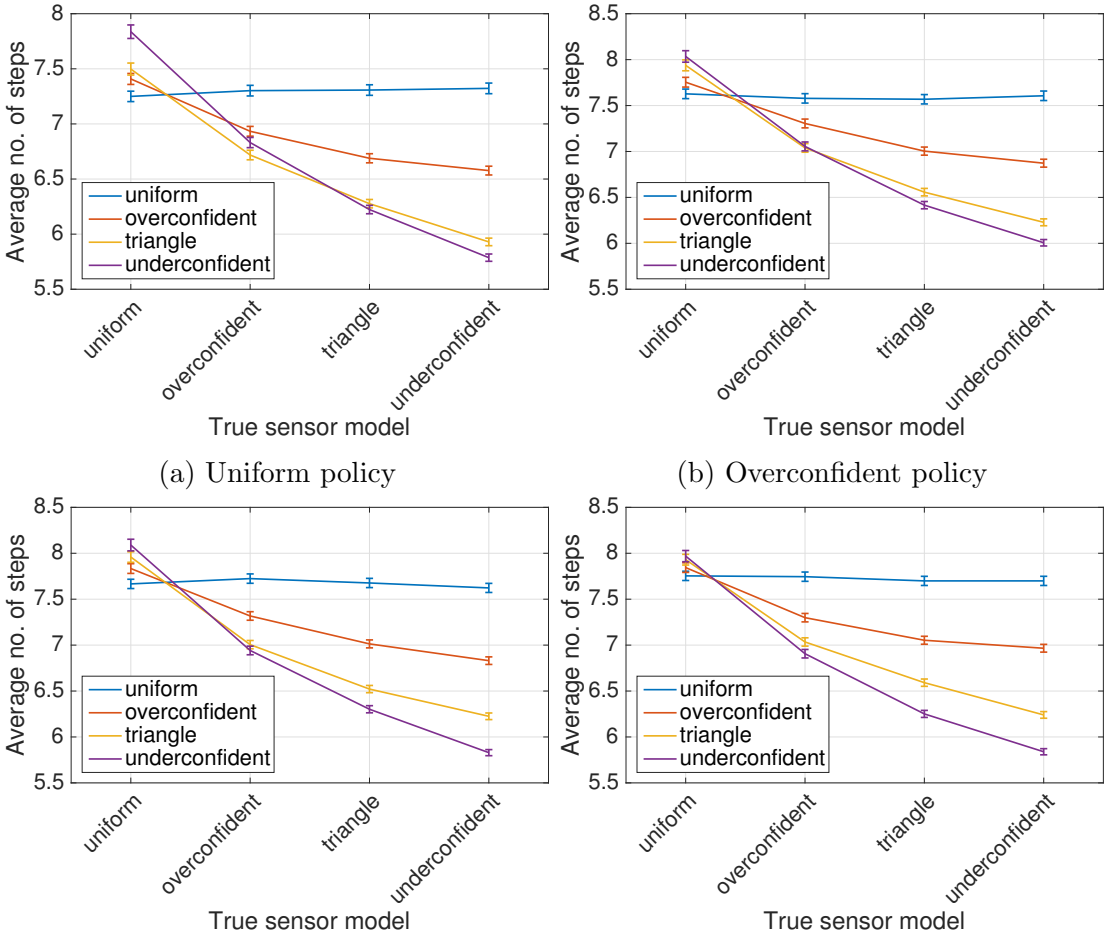
Figure 7.8: Wumpus, reward: we investigate the effects of using a sensor which is not consistent in the face of non-stationary data. We fix (A), the policy and test every pairwise combination of (B) and (C), the sensor's true model and the agent's model of the sensor, showing the average reward. Along the horizontal axis lies the true sensor model, and each curve shows an agent expecting measurements from a sensor corresponding to the legend. Error bars represent the standard error over 10,000 independent tests over all 20 possible 2×3 maps.

stand to gain more in terms of speed of solution by modelling the sensor as introspective than we stand to lose if that assumption is incorrect. We propose that this is the key to understanding the relationship between the agent's expectation and the truth about the model. By modelling the sensor as informative, the agent sometimes uses the measurements it receives to greatly increase 'peakiness' in the belief — more than it does when it models the sensor as non-informative. In this way, it can become overly confident about the true state, and thus rather than gather more information, it explores the world aggressively. This aggressive behaviour results in quick terminations, either by picking up the gold quickly, or by straying into the wumpus and being eaten. Thus, the average number of steps is much lower, but the risk is greater and the average reward is smaller.

Examining the grid world results in Figure 7.7, the intersection of the *uniform* and *under-confident* curves is approximately in the centre of each figure. This is not true in Figures 7.8 and 7.9. We attribute this difference to the asymmetry in the termination outcomes: it is hard to terminate with +1000 reward because the gold needs to be *picked up*, whereas terminating with −1000 is much easier because the agent simply needs to stray into the wrong square. It is this imbalance which reduces the possible rewards towards the right-hand side in Figure 7.8. If the goal is to maximise reward and we are aware of a lack of classifier introspection, perhaps it would be best to assume the minimum of information, rather than overestimate it as we do in Chapter 6.

7.5.2 Consistent and Appropriate Sensors

In this experiment we examine a subset of the results from those in Section 7.5.1, notably in the case that the sensor (B) remains consistent with the decision making system's expectations (C) despite possibly non-stationary data. This is the case where our classifiers are more introspective, because their behaviour remains consis-



(a) Uniform policy

(b) Overconfident policy

(c) Triangle policy

(d) Under-confident policy

Figure 7.9: Wumpus, number of steps: we investigate the effects of using a sensor which is not consistent in the face of non-stationary data. We fix (A), the policy and test every pairwise combination of (B) and (C), the sensor's true model and the agent's model of the sensor, showing the average reward. Along the horizontal axis lies the true sensor model, and each curve shows an agent expecting measurements from a sensor corresponding to the legend. Error bars represent the standard error over 10,000 independent tests over all 20 possible 2×3 maps.

tent, and the decision-making system is capable of making the best possible decisions given that sensor.

We also wish to determine the importance of the choice of (A) in terms of policy quality. The outcome of this experiment indicates whether the information content is relevant to policy generation, or whether one can simply generate a policy using any classifier and then use an introspective classifier at run-time. We expect the choice of sensor model for the policy generation to steer the approximation towards belief states often seen by that kind of sensor. Therefore, the value function and hence the policy is likely to be closer to the optimal policy for that chosen sensor. In most applications it is intractable to explore the whole belief space, so most POMDP solvers seek to explore the *reachable* region of the space (as described in Appendix E.2). The observation distribution is used to determine which areas of the belief space are reachable, and hence better approximate the optimal policy in those regions. The analysis of this experiment will aid us to understand the importance of the direction of this search and whether the resulting policy is effective in generalising to new regions of the belief space.

Grid World

For this experiment the agent has two sensors, the size of the map is 10×10, and the policy training time is two hours using a single 2.5GHz core. In Figure 7.10 we show the average number of steps for the agent to reach the goal, where the horizontal axis denotes (A), the sensor model used to generate the policy, and each curve represents the sensor model used for *both* (B) and (C), the true sensor model and the agent's model of the sensor.

Firstly, the gradient of the curves is close to 0, indicating that the sensor used to generate the policy is almost irrelevant to the behaviour of the agent. This indicates that the policy allows the agent to generalise well to new sensor types,

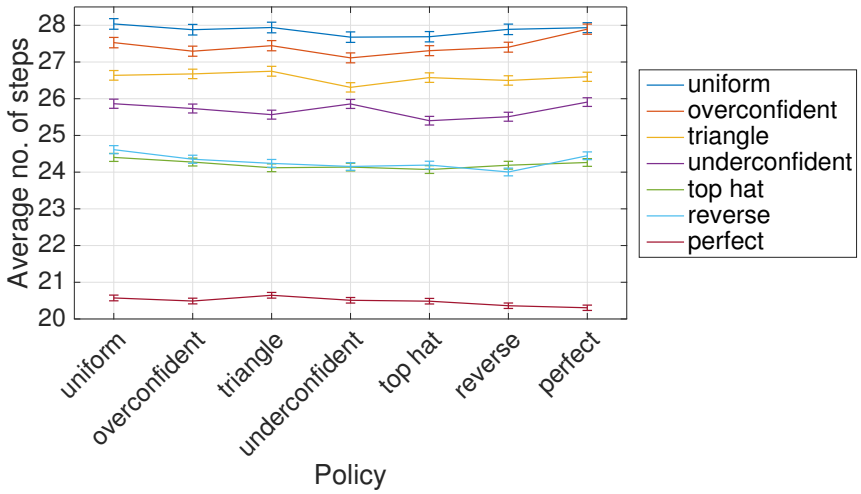


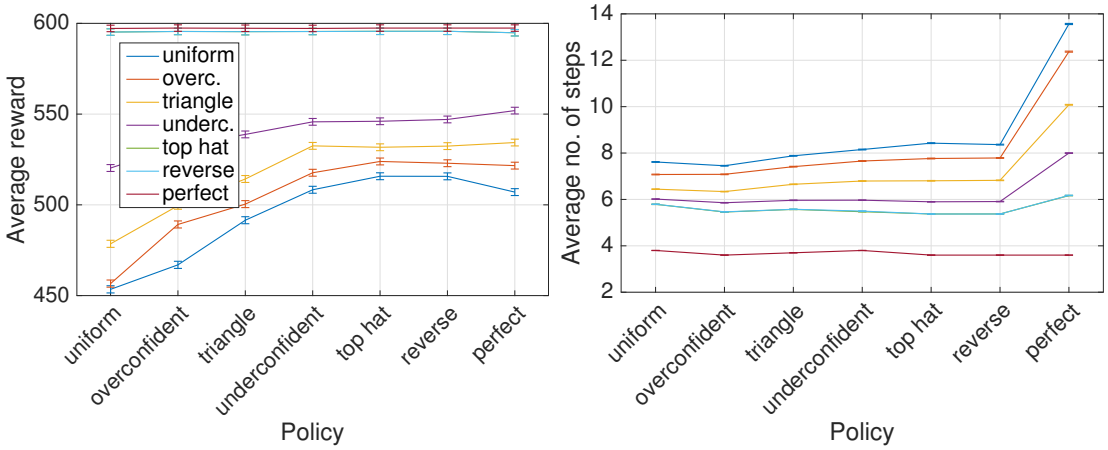
Figure 7.10: Grid world: introspective consistency in sensing. We show the average number of steps required to attain the goal for each pairwise combination of (i) classifier used to generate the policy (the horizontal axis), and (ii) the classifier used by the agent as it traverses the grid world towards the goal (each coloured line). A smaller number of steps is more desirable. For all classifiers the number of steps is invariant to the classifier model used to generate the policy. For this experiment the agent has two sensors, the size of the map is 10×10, the policy training time is two hours using a single 2.5GHz core, and the lines and error bars show the means and standard errors of 10^4 tests over the same map.

perhaps because all idealised classifiers explore a similar area of the belief space in this scenario. Secondly, the speed of the agent's solution is in line with the predictions made with regards to the entropy in Section 7.4. The agents using more informative classifiers reach the goal faster, implying that the information content of the measurement is key to performance. Thus, information content is important but only at run-time for this scenario, rather than during policy generation.

Wumpus

The average reward is shown in Figure 7.11a. Similarly to the grid world scenario, the horizontal axis denotes (A), the sensor model used to generate the policy, and each curve represents the sensor model used for *both* (B) and (C), the true sensor model and the agent's model of the sensor.

Firstly, note the ordering: the more informative classifiers yield a higher average



(a) (b)

Figure 7.11: Wumpus: introspective consistency in sensing. (a) the average reward at termination, and (b) the average no. of steps to termination. The horizontal axis shows (A), the sensor used to generate the policy, and each trace shows (B) and (C), the true sensor model and the agent's model of the sensor. Note that termination could either be as a result of grabbing the gold (+1000 reward) or being eaten by the wumpus (−1000 reward). Also note that the *top hat* and *reverse* curves are so similar that they appear as one. The map is 3×3, the lines and error bars show the means and standard errors of 10^4 runs over the same map, and the policy training time is two hours using a 2.5GHz core.

reward. It is important for a classifier to be consistent across non-stationary test data, and also for it to be informative regarding the class. As in Chapter 6, the introspective *top hat* classifier produces the best decisions.

Secondly, the agents carrying the *top hat* and *reverse* classifiers are as one line, and very close to the *perfect* classifier. This proximity indicates that the three classifiers are almost as informative as the *perfect* classifier, although the vertical difference between them is caused by the increased number steps required by the *top hat* and *reverse* classifiers, as shown in Figure 7.11b. We suggest that because their measurements are sometimes completely uninformative, these two agents *grab* more often in order to receive more measurements. These three agents are also relatively agnostic to the sensor model used to generate the policy, like in the grid world scenario. The remaining four agents, however, are not. The *under-confident*, *top hat*, *reverse*, and *perfect* policies are the best. Since the policies were generated for fixed time, it is possible that the policies for these classifiers reached more training iterations due to the relative sparsity of the belief states, and thus the policy is a better approximation of the optimal policy. We propose that the reason this effect is not seen for the grid world experiments in this section is that the state space here is very much greater, and so in the two hours of training time the resulting grid world policy is much closer to the optimal policy than in the wumpus scenario.

Although less key, it is interesting to note the number of steps to termination (resulting from either picking up the gold, or being eaten by the wumpus) in Figure 7.11b. The choice of model for the policy generation seems to bear little importance on the average number of steps to termination for all except the *perfect* policy, where it causes an increase. This increase in the number of steps corresponds to a divergence in average reward, where the *uniform* agent performs worse and the *under-confident* agent performs better. We propose that this policy results in the agent performing many more *grab* actions. For the *uniform* agent, this increases the

length of a run without any significant effect on the reward. For the more informative agents, however, their belief state improves and they make correspondingly better decisions.

7.5.3 Changing the Size of the World

In this section we examine the effects on the agent's speed (and quality) of solution with world size. An increase in world size leads to an increased number of necessary moves, and given that the classifiers give varying quantities of information per measurement, we expect the less informative classifiers to have a more uniform belief than the more informative classifiers. This reduction in 'peakiness' of the belief distribution should lead to a greater ambiguity of action choice, and thus an increase in number of steps to completion.

Like the experiment in Section 7.5.2, we vary (B) and (C) together, but always using the *uniform* model for (A). We have shown that the choice of model used for policy generation is unimportant for the grid world and does not change the ranking of the results in the wumpus world, so we consider using a single policy to be sufficient to characterise the effects of world size.

Grid world

In Figure 7.12 we show the number of steps to termination as we increase the world size. We generate square worlds with the following side lengths: {6, 8, 10, 12, 14, 16, 18, 20, 30, 40}, generate a policy for each using the *uniform* model, and then apply the agents 10,000 times to each map. For this scenario we do not expect the difference between classifiers to be large, because a specific sequence of uninformative measurements can give a lot of information regarding the agent's current position.

We see that the agents sporting more informative classifiers reach the goal in fewer steps, and that as the world size increases, the benefit of a more informative

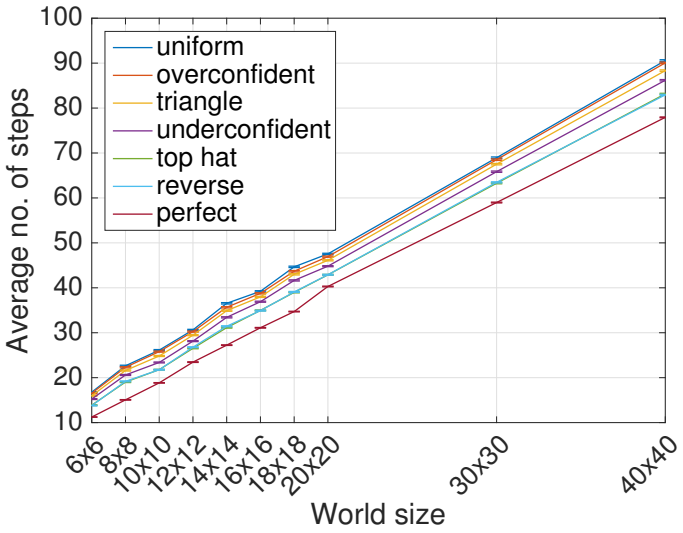


Figure 7.12: Grid world: we show that the larger the map, the greater the effect of the introspective qualities. The differences between the classifiers is 6 steps for a 6×6 world, but 13 for a 40×40 world. We also show that the *top hat* and *reverse* classifiers perform equivalently. All of these are using the policy generated using the uniform classifier with three sensors. Error bars represent the standard error over 10,000 independent tests over the same map.

classifier also increases. The difference in the number of steps required by the *uniform* and *perfect* agents is approximately 6 in a 6×6 world, but 13 in a 40×40 world. This increase is consistent with our prediction.

Wumpus

In Figure 7.13a we show the average reward gained by the agents over various world sizes. We show fewer world sizes than for the grid world because the order of computational complexity is much larger here. The grid world has one state per possible agent position, which even for a 100×100 world is a modest 10^4 states. The wumpus scenario, however, considers a *configuration* of the map to be a state, so a 4×4 map containing the agent, gold, and a wumpus has approximately $3^{4\times4} \approx 43 \times 10^6$ states. Because the problem complexity here is larger than for the grid world, and the agent has a significant risk of incurring a large negative reward if it has a poorly informed state estimate, we expect a larger discrepancy between the

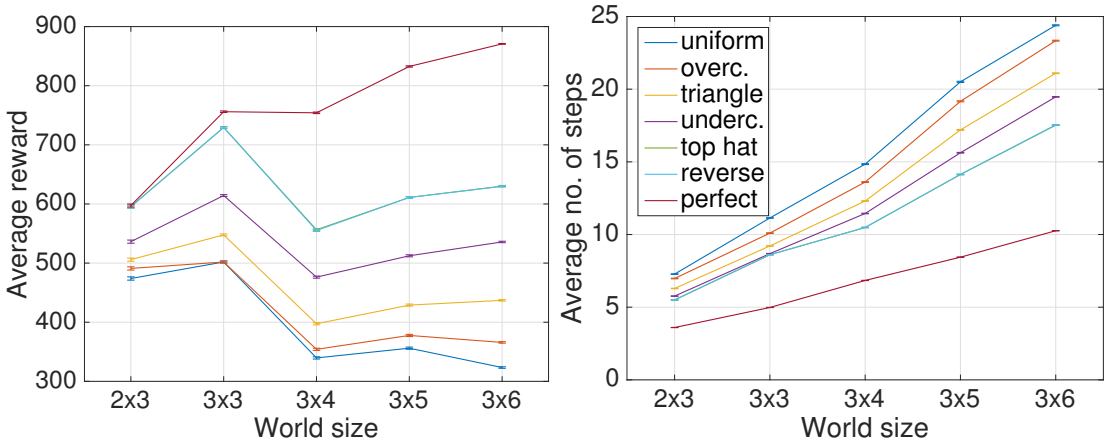
classifiers.

Firstly, there is a divergence between the agents as the world size increases, which is the effect we predicted. Secondly, we see that the average reward does not change smoothly as the world size is increased. We propose that this results from not testing the agents on every possible map at each world size. There are approximately 272 valid 3×6 maps, and testing 5,000 times on each one would be very computationally expensive. The particular choice of map has a large effect on the resulting reward, and so using a subset of 50 of the possible maps will introduce significant variance. Nevertheless, the difference in reward between the most and least introspective agents is approximately 110 for the 2×3 world, but 550 for the 3×6 world. This is indeed larger than the differences in the grid world, as expected. Lastly, we see the same ordering and divergence as in the grid world scenario, and the *top hat* and *reverse* agents perform equivalently to each other.

7.5.4 Changing the Number of Sensors

Here we vary the number of sensors and examine the average number of steps for the agent to reach the goal in the grid world. In the wumpus world each of the sensors is fixed for the scenario formulation, so it would not be useful to add or remove any. Here, we perform 10,000 independent tests for each matched pairing of (B) and (C) on a randomly generated 10×10 grid.

The results are shown in Figure 7.14. As with the previous experiments, the ordering is preserved: more informative classifiers solve the map faster. The negative gradient indicates that increasing the number of sensors allows an agent to reach the goal faster. This is due to a better location estimate (or a less uniform belief) and results in correspondingly better actions, leading to a shorter path to the goal. The added value of increasing from one to two sensors is greater than or equal to that of increasing from four to five. Giving the agent more information is beneficial,



(a) (b)

Figure 7.13: Wumpus: we show (a) the average reward, and (b) the average number of steps to termination (which could be either the positive or the negative outcome) of the agents as a function of world size. The larger the map, the greater the divergence between the classifiers and therefore the greater the effect of the introspective differences. Note that the number of steps to termination not only increase, but the classifiers diverge as the world size increases. We show that the larger the map, the greater the effect of the introspective qualities. Also note that the *top hat* and *reverse* curves are so similar that they appear as one. Error bars represent the standard error over 5,000 independent tests over up to 50 maps.

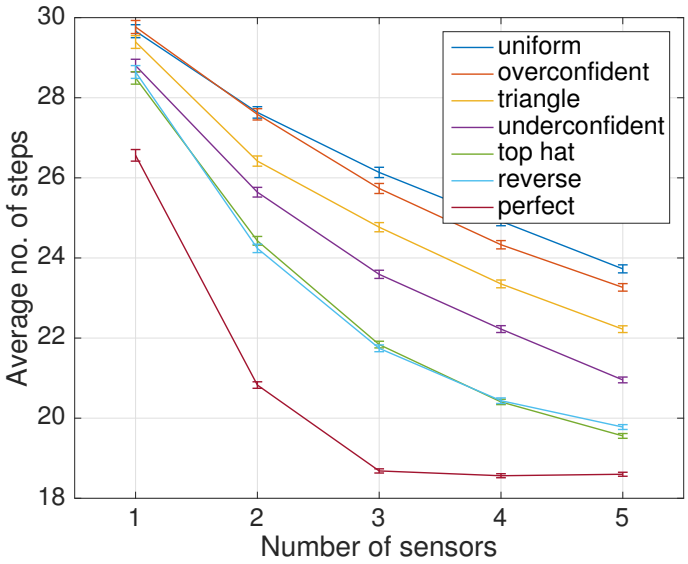


Figure 7.14: We show that the more sensors are available to the agent, the greater the effect of the introspective qualities. We also show that the *top hat* and *reverse* classifiers perform equivalently. All of these are using the policy generated for two hours on a single 800MHz core using the uniform classifier in a 10×10 map. Error bars represent the standard error over 10,000 independent tests over the same map.

so adding a *perfect* sensor is more valuable than adding a *uniform* sensor. However, we see diminishing returns in the number of sensors, as evidenced by the gradients of the curves becoming less negative as the number of sensors increases. The *perfect* agent reaches steady state at three sensors, at which point it knows exactly where it is in the world at almost every time step, and thus always chooses the best action that the policy will allow.

7.6 Conclusions

In this chapter we have examined the effects of information content and classifier consistency when confronted by non-stationary data. We have shown that using a sensor that changes characteristics unexpectedly will result in poor decision-making. In the wumpus world, assuming a higher level of information than is available is worse than the converse due to the greater difficulty in achieving the positive reward

than the negative rewards. This is what we do when applying overconfident class estimates to classical decision-making systems. We have shown that when we have a consistent, introspective classifier, the information content of the measurements determines an agent's effectiveness. Finally, we demonstrated diminishing returns when adding additional sensors, and a that the difference between the classifiers was amplified by increased world size and scenario complexity.

We have found that because the mapping from z to the class is explicitly stated, unlike in classical decision making, it is not required for mistakes to be made with high uncertainty. This is shown by the equivalent performance of the *top hat* and *reverse* classifiers. Instead of a correlation between correctness and confidence, the POMDP framework requires any consistent mapping between z and class. This is because the POMDP is unaware that the observations might be probabilities. The information content of a particular classifier as measured by entropy determines the efficiency with which an agent solves various problem scenarios. However, we emphasise that introspection will induce good decision making performance in sequential decision making, and indeed the element of consistent behaviour is paramount.

Chapter 8

Conclusions and Future Work

In this chapter we first present our conclusions in Section 8.1, before outlining a number of avenues for future work in Section 8.2.

8.1 Conclusions

The following points outline the conclusions and contributions of this thesis:

- **Introspection.** We introduce and name the concept of introspection, a characteristic that we argue is required of a classifier for it to make good decisions in robotics. We motivate this in the context of making mission-critical decisions, where the degree of uncertainty with which a classification is made is key to avoiding catastrophic outcomes. Two examples are semantic mapping, and a "robot crossing the road" scenario, where the robot must decide whether it is safe to cross, risking a collision. In Section 4.1 we propose that the 'ideal' imperfect classifier (the top hat classifier), which makes correct decisions when it is confident, and only makes mistakes with high uncertainty. This consistent correlation between confidence and correctness gives it the important introspective quality. This requirement is rooted within the fact that our classifiers

operate on non-stationary data, where the trained model is unlikely to generalise well to all future test data. In order to benchmark a number of real classifiers, we derive several idealised classifiers that have varying degrees of introspection.

- **A treatment of distance.** We propose that one way to invoke this introspective behaviour in a classifier is via a good treatment of distance. We argue that the data which are unlike the training data are likely to be misclassified, and so a classifier ought to respond to these with high uncertainty. This in turn results in a more cautious robot behaviour. This 'unlikeness' can be characterised by an increasing distance between the training data and the test datum. We examine a number of commonly-used classification frameworks and determine that two factors are relevant to this treatment of distance. The first and foremost is whether a classifier is single-discriminant or multi-discriminant. This label relates to whether, at test time, the test datum is evaluated by a single discriminant, or several which span the hypothesis space. We determine that multi-discriminant classifiers offer a mechanism of determining whether a test datum is far away from the training data, by virtue of a predictive variance: if all the discriminants agree on a classification, it is likely to be near the training data; if there is discord, it is likely to be further away and should be treated with suspicion. The multi-discriminant classifiers are determined to be the Gaussian process classifiers and the random forest, while the single-discriminant classifiers are the support vector machines and LogitBoost. The second factor of import is whether the classifier uses a linear or non-linear kernel. In low-dimensional feature spaces (two or three dimensions), classifiers using a linear kernel are less capable of determining distance between training and test data. However, this effect dwindles as feature dimensionality increases, leading to the number of discriminants becoming the dominant

factor.

- **Examining high-uncertainty decisions.** In Section 6.1 we investigate the uncertainty of classifications which are made erroneously, and conclude that the multi-discriminant classifiers are more uncertain than the single-discriminant classifiers. This is consistent across the data sets, although the margins are small. We predict that this should result in an increased competency in active learning, where classifiers choose high-uncertainty queries which are then labelled by an oracle and added to their training set. Comparing a single-discriminant and a multi-discriminant classifier, we show that indeed the multi-discriminant IVM chooses queries which are more beneficial than the single-discriminant SVM in terms of increasing classification performance.

- **Examining high-confidence decisions.** In Section 6.3 we apply the classifiers' outputs to a classical decision-making system, simulating a "robot crossing the road" scenario. We test their adherence to the expectation that as the cost increases, a classifier which makes good decisions will avoid the risky decision to *go* unless it is very confident that the way is clear. We show that as we increase the relative cost of a false negative error, no real classifier avoids incurring large cost across all three data sets. This highlights the general lack of introspective capacity of the classifiers. We benchmark the introspective failings of the real classifiers against the idealised classifiers.

- **Examining sequential decisions.** We analyse the abilities of various agents when they must make a sequence of decisions, each receiving measurements from a particular idealised classifier. We discuss the importance of the information content in the measurements, and note that our 'top hat' classifier maximises this information content. Indeed, the more informative classifiers lead their respective agents to success faster than the less informative classifiers.

Furthermore, we conclude that when the characteristics of the sensor change unexpectedly, as is likely the case in non-stationary data streams and/or if using overconfident classifiers, there is no safe assumption for the model. Any discrepancy here will result in sub-optimal behaviour, and the best strategy is to choose a classifier which will behave consistently in changing data streams. It may be helpful to track the statistics of the incoming data in order to gain awareness of when they have become non-stationary. Generative models can do this by calculating the probability that a set of data comes from the same distribution as the training data.

- **Final impressions** In summary, we have made progress towards a series of tests which analyse the ability of a classifier to make appropriate decisions in various settings. We have defined the ideal classifier behaviour as a benchmark for real classifiers to strive for, and proposed the importance of multi-discriminant classifiers that offer predictive variance. Of the real classifiers benchmarked in this thesis, there is no clear introspective winner. The search for the introspective classifier is still on.

8.2 Future Work

The work presented in this thesis leads in many directions, and in this section we outline a number of lines of research which would further contribute to the understanding and leverage of introspection in robotics.

8.2.1 Introspection

1. **Additional classification frameworks: deep neural networks.** A huge amount of attention is being paid to deep neural networks [Mnih et al., 2013] [Krizhevsky et al., 2012] [Simonyan and Zisserman, 2015], and it would be very

interesting to evaluate the introspective capacity of a deep net to the classification frameworks already examined in this thesis. We share the opinion of Gal and Ghahramani [2015] that the use of dropout in the learning phase of the network is a form of model averaging, and could result in introspective behaviour. Dropout is a technique used to reduce the risk of overfitting, whereby a random subset of the computational units are dropped from the optimisation procedures, resulting in a level of network redundancy [Srivastava et al., 2014]. However, at test time it is possible to employ the same technique on test data, evaluating several different subsets of the network units to generate multiple class decisions. This is akin to sampling from the hypothesis space, and the collection of measurements can be used to calculate the predictive variance we consider to be key to introspection.

2. **Additional classification frameworks: calibrated AdaBoost.** As remarked in Blair et al. [2014], the LogitBoost classifier we employ in this thesis has a sigmoid to map scores to a score in $[0, 1]$, but the sigmoid takes two parameters, scale and offset, which are not learned in LogitBoost. Learning these parameters via Platt's method [Platt, 1999] or more appropriately, the improved algorithm proposed by Lin et al. [2007] will provide a better mapping. This method is already applied to the SVMs as part of the LIBSVM implementation Chang and Lin [2011], and is likely to yield a less over-confident classifier than LogitBoost. Note that this classifier is still single-discriminant and linear, so given the observations in this thesis we would expect it to perform similarly to the linear SVM in terms of introspection.

3. **Additional classification frameworks: generative models.** All the classifiers considered in this thesis are discriminative. That is, the learned models serve to discriminate between classes, but they do not explicitly model the dis-

tributions of the training classes themselves. The reason for this choice is that discriminative models typically outperform generative models in classification tasks, with the exception of cases with very few training data [Ng and Jordan, 2002]. However, we partially motivate our work through the non-stationarity of real data, and generative models can be used to determine whether the training data are representative of the test data. Generative classifiers include Gaussian Mixture Models (GMM) [Rasmussen, 2000], Kernel Density Estimation [Bishop, 2006], Naive Bayes [Ng and Jordan, 2002].

4. **Additional classification frameworks: different idealised classifiers.** In Chapter 4 we introduced a range of idealised classifiers, designing their error functions and deriving appropriate probability density functions. In doing so, we made a number of assumptions. The effects of relaxing those assumptions would be of interest. For instance, the distribution of $f(z)$ is uniform, but by examination of Figures D.1 and D.2 we see that this is not the case for the real classifiers. Secondly, the error functions and density functions are all symmetrical around $z = 0.5$, which is a reasonable approximation to most of the density functions, but not all. The Random Forest, in particular, is very asymmetrical. Thirdly, all the idealised classifiers have a total expected error rate of 0.25. We could vary the total error rate along with the entropy of the density functions to investigate the relative importance of those terms.

5. **Introspection in Big Data.** In this thesis we have concerned ourselves with relatively small numbers of training data (from 100 to 750). There are many semantic mapping and autonomous operation tasks for which labelled data are scarce. However, labelling data is cheap compared to making catastrophic mistakes in autonomous tasks, so in some situations a user might have access to larger bodies of labelled training data. A similar investigation with a hundred

or a thousand times the amount of training data would be valuable.

6. **The effects of class imbalance.** One variable we have not investigated in this thesis is the imbalance in the number of training data for each class. In the third-class and classification experiments of Chapter 5 the number of data is balanced in both training and testing. In the detection case, the training data have a ratio of 1:2, and the test data have a ratio of approximately 1:10. We justify the choices in Section 5.7, but we do not examine the effects of these choices. A further set of experiments which investigate the importance of these ratios could inform how it is best to train classifiers in the future.

7. **Third class relevance in detection.** In Section 5.6 we show a clear difference between the multi-discriminant classifiers and the single-discriminant classifiers in terms of introspective capacity, but the differences do not seem to exhibit themselves so strongly in the detection experiments of Section 5.7. Perhaps the data sets for the detection experiments are particular in their stationarity, or perhaps the data imbalance or parameters are too different to be comparable, but a thorough investigation of what aspects of the data bring out such behaviour is required.

8. **More ideal introspective behaviours.** An additional ideal behaviour would be that the overall test uncertainty should decrease as the classification model approaches the optimal model given full training information. In other words, a classifier with very few data to characterise complex classes would ideally be uncertain about most test instances, and as more data are added, the proportion of test classification with greater confidence should increase. We have investigated this in part with synthetic data in Section 5.6.1, but a more detailed investigation into the progress of uncertainty of real classifiers with increasing training data would be valuable.

9. **Bias and variance in multi-discriminant classifiers.** The concepts of bias and variance in machine learning relate to the ability for an ensemble of learners to achieve their target value. An ensemble of learners in which each individual gives the same measurement for a given test datum have no variance, and an ensemble which always gives the correct measurement has no bias. There is a parallel between this and our analysis of single and multi-discriminant classifiers. We argue that predictive variance ought to be small near to training data, and large far away from training data. An investigation of this parallel would be of great interest.

10. **A designer feature for introspection.** Underlying our proposition that an appropriate treatment of distance is a way to achieve introspection is the expectation that we are more likely to correctly classify data nearby training examples. This is not guaranteed for all feature types. It might be possible to design a feature which lends itself to increasing the distance between training data and misclassified test data. One way to do this might be to learn the feature mapping from the data, like a neural network There may be a benefit in tailoring a feature to a particular classifier model.

11. **Varying the number of discriminants.** We argue that multi-discriminant classifiers are more introspective than single discriminant classifiers. It would be interesting to sample k discriminants from the version or hypothesis space and show that introspective quality increases with k. This could possibly be achieved in GPC by sampling from the predictive distribution $f(\mathbf{x}_*)$, or in the case of the Random Forest, by changing the number of trees.

8.2.2 Semantic Mapping

1. **Evaluating Map Confidence.** In this thesis we have concerned ourselves with the uncertainty of single classifications. How should we extend this to the uncertainty of a whole image, or of a section of a semantic map? If we could consistently and accurately judge map uncertainty, we could automatically determine whether a robot is capable of making safe decisions within a certain region. This is a crucial component to *autonomy on demand*, a common principle in autonomous driving in which the vehicle offers autonomous operation in some situations but not others.

8.2.3 Active Learning

1. **What data yield high-uncertainty classifications?** For a well-calibrated non-linear kernel method such as the IVM, there are two situations which yield high uncertainty. Either the point lies directly on the decision boundary, or it lies very far from the training data. These two very different situations yield the same high-uncertainty response, but intuitively their effect on active learning will be very different. Labelling such points and retraining will have varying bearing on the resulting model. Preliminary investigations have shown that data lying on the decision boundary have very little effect in changing the model when labelled and retrained. The more distant points, however, can move the model more and allow it generalise better. How can we determine which is which? The Gaussian Process regressor, before the sigmoid is applied, yields a predictive variance which could be used to determine whether the point is close to the training data or not. This predictive variance could be used as a criterion for active learning with a Gaussian Process classifier.

2. **Testing active learning with explicitly non-stationary data sets.** In

the experiment in Section 6.2 we carry out an active learning experiment on the same data sets used in the detection experiment in Section 5.7. The results from the detection experiment show that there is not a great deal of difference between the behaviours of the classifiers in that situation, which predicts a small effect in active learning. However, if we were to carry out an active learning experiment on an explicitly non-stationary data set, for instance those in either of the third-class experiments from Section 5.6, we are likely to find a larger difference between the IVM and the SVM. A half-way solution could be to use a detection data set, but with a non-stationary positive class. For instance, training on a set of road signs versus background, and testing on a different set of road signs vs background.

8.2.4 Classical Decision Making

1. **Testing high-cost decision-making in explicitly non-stationary data sets.** Similarly to paragraph 2 in Section 8.2.3, we might expect the more introspective classifiers following the third-class experiments in Section 5.6 to outperform the others in a high-cost decision making scenario if the data set is more explicitly non-stationary. The previous point in this section would also serve to inform us about this to a greater extent.

2. **Decision-making based on ϵ-bounds.** In the high-confidence decision making experiment we determine the costs first and then analyse the decisions. We consider this a *problem-centric* method to determine the threshold T on the measurement which results in either *waiting* or *going*. In Section 6.3.2 we propose setting the threshold by considering a specific classifier's behaviour, such that the probability of a particular outcome is bounded to some predetermined value ϵ based on a training set. This is a *classifier-centric* method

of choosing the threshold. This is not likely to affect the consistency of the classifiers, but for stationary data sets it may result in better decision-making than determining the cost matrix.

8.2.5 Sequential Decision Making

1. **Closeness to the optimal policy.** Because we follow the authors advice and terminate the optimisation for generating the policy after a fixed period of time [Kurniawati et al., 2008], the resulting policy may or may not be close to the optimal policy. A more detailed evaluation of the role of the length of time before termination and the quality of the resulting policy would be of value, particularly in problem scenarios which are computationally expensive.

2. **Extending the problem set.** We have considered two problem scenarios which we believe represent the body of standard problems, but it would be valuable to confirm our findings in a variety of settings.

3. **Making the wumpus scenario more efficient.** The wumpus problem we have used in our experiments in Chapter 7 is very simple in that there is only ever one wumpus. The original problem as set by Russell and Norvig [2003] allows for more than one wumpus and also a number of *pits*, which also terminate the episode with a large negative cost, and squares adjacent to a pit are *breezy*. Indeed the agent has a sensor for breeziness. In some formulations of the problem, the agent also carries a single arrow, which it can fire in one of the four cardinal directions, and if the arrow passes through a wumpus it will die and *scream*, clearing the way for the agent. There would also be a *scream* sensor on board the agent. We use a simplified scenario here because already the state space is very large and generating (and writing to disk) the observation and state transition distribution functions takes a long time (in

the order of hours in a 3×5 grid. There will be ways to lessen the file sizes and computational burdens, enabling experiments on more complex versions of the scenario and over larger sizes.

4. **Practical scenario.** Throughout the thesis we compare the real classifiers and the idealised classifiers, but in Chapter 7 we consider only the latter. Evaluating the information gain from real classification frameworks on real data using a POMDP to make decisions could inform classifier choice in practical problems.

5. **Classifiers without full support in** $[0, 1]$**.** The support of a function f on the interval $[0, 1]$ is the set of values z at which $f(z) \neq 0$. In Section 7.5.1 we evaluate the performance of an agent when its model of the sensor is not necessarily the true sensor model. This creates a deliberate disparity between the actual value of an observation and the agent's perception of its value, similarly to a non-introspective classifier being tested on a third class or non-stationary data set; it can be unreasonably overconfident or under-confident. However, we cannot test on the *top hat*, *reverse*, or *perfect* classifiers because the lack of full support in $z \in [0, 1]$. If the agent's model of the sensor is *perfect* but the true model is *uniform*, the sensor can give an erroneous measurement, but the agent interprets it as being correct. Certain combinations of moves and erroneous measurements will result in a belief update which cannot account for a particular measurement given a particular belief, resulting in a perceived logical impossibility. This occurs in map-building: a robot running a SLAM (Simultaneous Localisation and Mapping) system with loop closure can detect a loop which is not consistent with the local odometry. Typically a bundle-adjustment on the entire set of poses of the robot and its landmarks will be performed, using the loop-closure as a constraint [Sibley et al., 2010]. This

warps the existing poses to make up for the accumulated error in odometry. The POMDP framework in the form described in Appendix E is not capable of revisiting previous decisions, owing to the Markov assumption. One might argue that our robots need to be able to revisit past decisions in order to compensate for previous errors.

Appendices

Appendix A

Road Graph Generation Algorithms

In this appendix we present the two algorithms used in Section 2.3.1 to create a road network from a collection of ordered vehicle poses. The aim is to reproduce the skeleton structure of the driving lanes which have been manually driven at least once.

Algorithm 1: Creating and simplifying the adjacency matrix required for Algorithm 2.

Data: vehicle poses $\mathcal{X} \in \mathcal{R}^3$, connecting distance d, number of pruning loops P

Result: Symmetric adjacency matrix A

`// Connect all nodes within a radius` d `of each other`

forall the *pairs of poses* $(x_i, x_j) \in \mathcal{X}$ **do**
  if $\|x_i - x_j\| \leq d$ **then** $A_{ij} \leftarrow 1$
  else
    $A_{ij} \leftarrow 0$; $A_{ji} \leftarrow 0$ `// Symmetric` A

`// Disconnect temporally subsequent nodes within` d

forall the *poses* $x_i \in \mathcal{X}$ **do**
  $n \leftarrow 1$
  while $\|x_i - x_{i+n}\| \leq d$ `// Exit if assertion fails`
  do
    $A_{i,i+n} \leftarrow 0$; $A_{i+n,i} \leftarrow 0$
    $n \leftarrow n + 1$

`// Connect nodes in the order they were driven`

forall the *poses* $x_i \in \mathcal{X}$ **do**
  $A_{i,i+1} \leftarrow 1$; $A_{i+1,i} \leftarrow 1$

`// Prune cliques larger than 2, replacing them by the node nearest to the clique centre`

for $p \leftarrow 1$ **to** P **do**
  `// Produce an ordered list of maximal cliques`
  $C \leftarrow$ `findMaximalCliques`(A)
  $C \leftarrow$ `sortDescending`(C)
  $R = \emptyset$ `// The set of nodes marked for removal`
  forall the *cliques* $c \in C$ *such that* $|c| > 2$ **do**
    $x_\mu \leftarrow$ `mean`(x_c)
    $x_b \leftarrow$ `returnNearestNode`$(x_\mu, \mathcal{X})$
    $N \leftarrow$ `returnNeighbours`(c, A) `// N includes indices` $i \notin c$ `but adjacent to at least one node in` c
    `// Skip cliques containing nodes marked for removal`
    if $N \cap R = \emptyset$ **then**
      $A_{b,N} \leftarrow 1$; $A_{N,b} \leftarrow 1$
      `// Find the dead-end nodes`
      $D \leftarrow \{i \in N$ for which `degree`$_A(i) = 1\}$
      `// Mark these nodes for removal`
      $R \leftarrow R \cup c \cup D \setminus \{b\}$
  $A \leftarrow$ `removeRowsAndCols`(R, A) `// remove the rows and columns of` A `which refer to the nodes in` R

Algorithm 2: Iteratively exploring the adjacency matrix from Algorithm 1 to discover the road network and its intersections.

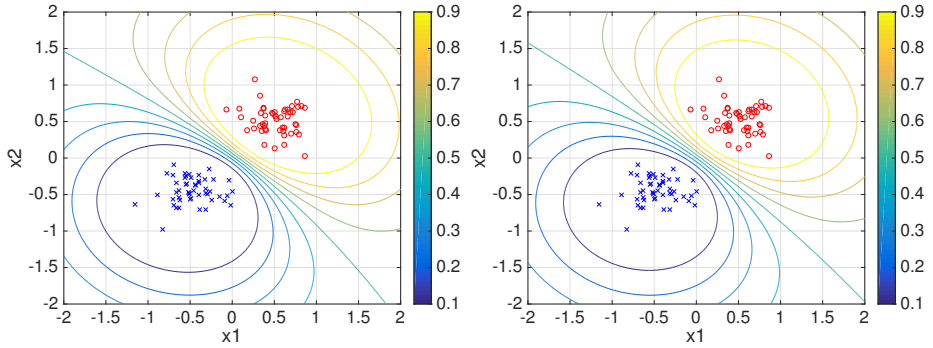
Data: Symmetric adjacency matrix A with M rows and columns
Result: Road network R, intersections I^*

$v \leftarrow \emptyset$ // Current node
$T \leftarrow \emptyset$ // Termination nodes
$R \leftarrow \emptyset$ // Road network
$I^* \leftarrow \{i \in \{1, \ldots, M\}$ for which $\texttt{degree}_A(i) > 2\}$
while A *nonempty* **do**
  // Intersection nodes
  $I \leftarrow \{i \in \{1, \ldots, M\}$ for which $\texttt{degree}_A(i) > 2\}$
  // Dead-end nodes
  $D \leftarrow \{i \in \{1, \ldots, M\}$ for which $\texttt{degree}_A(i) = 1\}$
  if $D \neq \emptyset$ **then**
    $v \leftarrow$ some $d \in D$ // Start at a dead-end node
    $T \leftarrow I$ // Terminate at an intersection node
  else
    if $I \neq \emptyset$ **then**
      $v \leftarrow$ some $i \in I$
      $T \leftarrow I \setminus \{v\}$
    else
      // Remaining road network is only one loop
      $v \leftarrow 1$ // Start anywhere
      $T \leftarrow \emptyset$
  $L \leftarrow v$ // Lane
  $O \leftarrow \texttt{returnNeighbours}(v, A)$ // Options
  while $(i \notin T) \wedge (O \neq \emptyset)$ **do**
    $v \leftarrow$ some $o \in O$
    $L \leftarrow (L, v)$ // add v to the vector L
    $O \leftarrow \texttt{returnNeighbours}(v, A) \setminus L$ // with L as a set
  $R \leftarrow R \cup \{L\}$ // R is a set of vectors
  if $L_1 \in I$ **then**
    $L \leftarrow L \setminus L_1$ // don't include start node
  if $L_{|L|} \in I$ **then**
    $L \leftarrow L \setminus L_{|L|}$ // don't include terminal node
  $A \leftarrow \texttt{removeRowsAndCols}(L, A)$

Appendix B

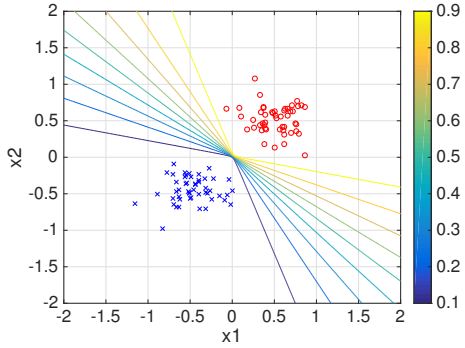
Classifier Probability Contours

In this appendix we train each classifier on two-dimensional synthetic data, and show the contours of the test probabilities. In Figure B.1 we show the contours for classifiers trained on linearly separable data, and in Figure B.2 we show the same for non-linearly-separable data. Note that these may not extrapolate to higher-dimensional spaces, and serve only as a simple illustration for low-dimensional space.

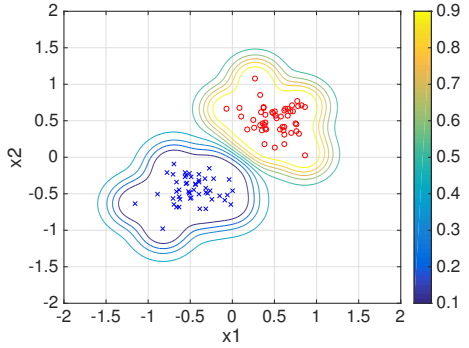


(a) IVM

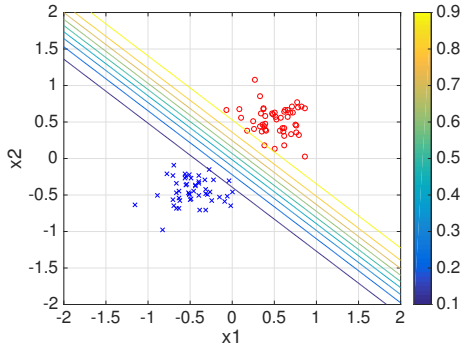
(b) Non-linear GPC



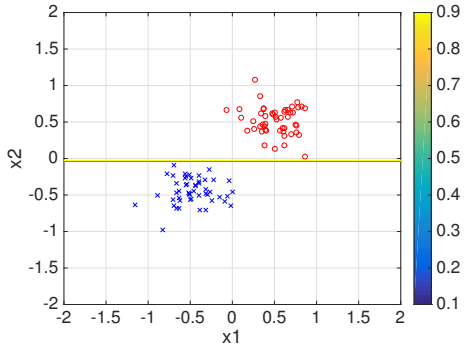
(c) Linear GPC



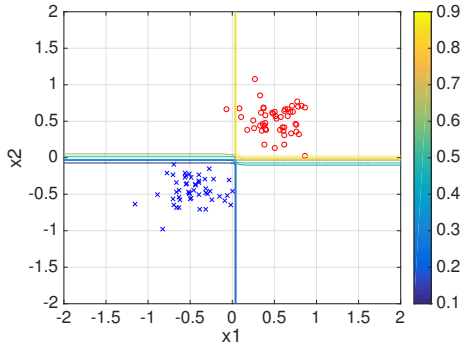
(d) Non-linear SVM



(e) Linear SVM

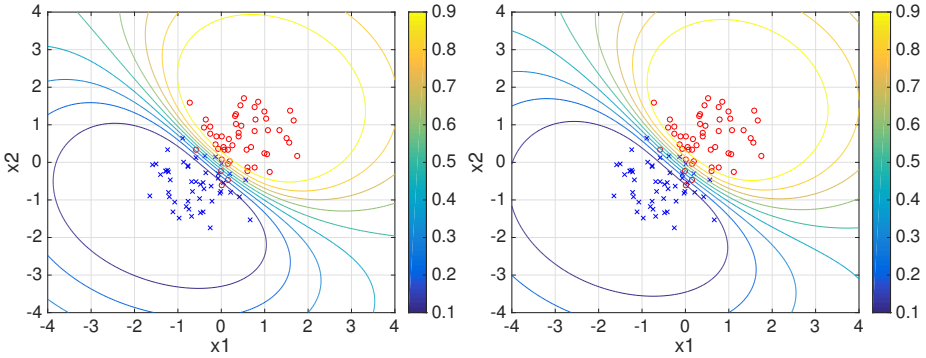


(f) LogitBoost



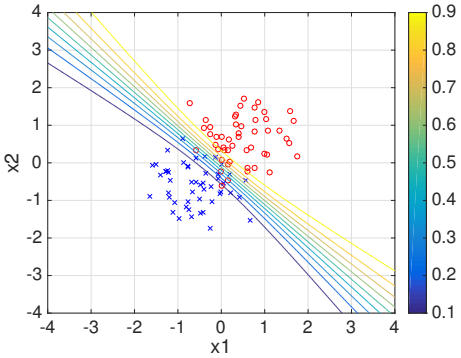
(g) Random Forests

Figure B.1: Separable: the probability contours for each classifier on two-dimensional data which are linearly separable. (Best viewed in colour.)

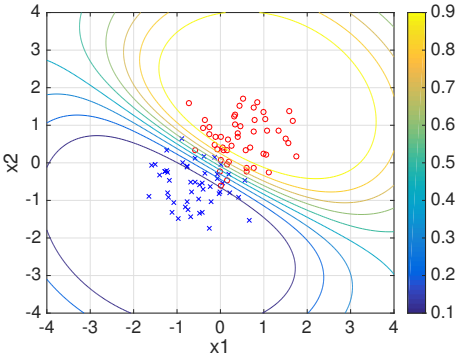


(a) IVM

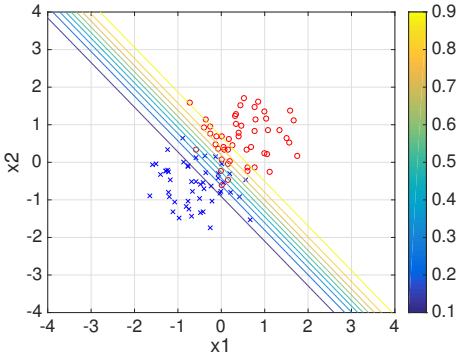
(b) Non-linear GPC



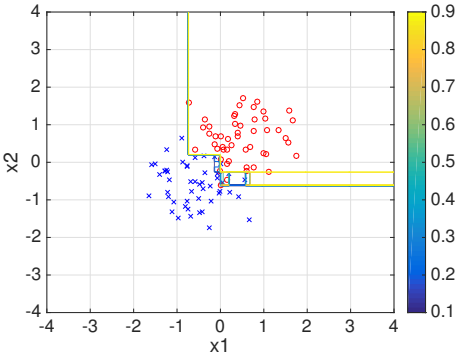
(c) Linear GPC



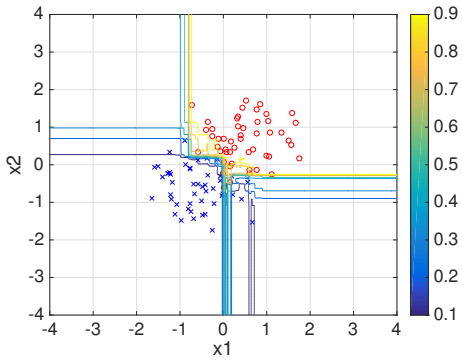
(d) Non-linear SVM



(e) Linear SVM



(f) LogitBoost



(g) Random Forests

Figure B.2: Non-separable: the probability contours for each classifier on two-dimensional data which are not linearly separable. (Best viewed in colour.)

Appendix C

Idealised Classifier Error Functions

Subject to the assumptions in Section 4.3.1 we wish to construct a number of simple examples error functions with which to benchmark real classifiers. For simplicity we consider scenarios in which the two classes are equiprobable, and so the error functions are symmetrical. In particular, we wish to explore the effects which arise from varying the extent to which mistakes occur around the decision boundary $z = 0.5$, while maintaining the same expected error rate of 0.25 for all classifiers. We also add a *perfect* classifier which makes no errors, to contextualise the others.

The following define the simple example error functions.

$$\textit{uniform}\colon\ E(z) = 0.25 \tag{C.1}$$

$$\textit{overconfident}\colon\ E(z) = \begin{cases} 0.25((4z-1)^3+1), & \text{if } z < 0.5 \\ 0.25((-4z+3)^3+1), & \text{if } z \geq 0.5 \end{cases} \tag{C.2}$$

$$\textit{triangle}\colon\ E(z) = \begin{cases} z, & \text{if } z < 0.5 \\ 1-z, & \text{if } z \geq 0.5 \end{cases} \tag{C.3}$$

$$\textit{under-confident}\colon\ E(z) = \begin{cases} 0.25(1+\sqrt[3]{4z-1}), & \text{if } z < 0.5 \\ 0.25(1+\sqrt[3]{-4z+3}), & \text{if } z \geq 0.5 \end{cases} \tag{C.4}$$

$$\textit{top hat}\colon\ E(z) = \begin{cases} 0, & \text{if } z < 0.25 \\ 0.5, & \text{if } 0.25 \leq z < 0.75 \\ 0, & \text{if } z \geq 0.75 \end{cases} \tag{C.5}$$

$$\textit{reverse}\colon\ E(z) = \begin{cases} 0.5, & \text{if } z < 0.25 \\ 0, & \text{if } 0.25 \leq z < 0.75 \\ 0.5, & \text{if } z \geq 0.75 \end{cases} \tag{C.6}$$

$$\textit{perfect}\colon\ E(z) = 0. \tag{C.7}$$

Appendix D

Empirical Probability Density Functions of Real Classifiers

In this appendix we show the empirical distribution functions of real classifiers trained and tested on the Daimler Pedestrian data set. Figures D.1 and D.2 show the distribution of the measurements z for the background class (left columns), and the pedestrian class (right columns).

It is clear that with the exception of the Random Forests, all classifiers behave very similarly in terms of measurement distribution, with the majority of the classifications being made with great confidence, near $z = 0$ and $z = 1$. The Random Forest classifier behaves very differently to the rest, exhibiting much less confidence in both classes, but especially the pedestrian class, which is extraordinarily uncertain.

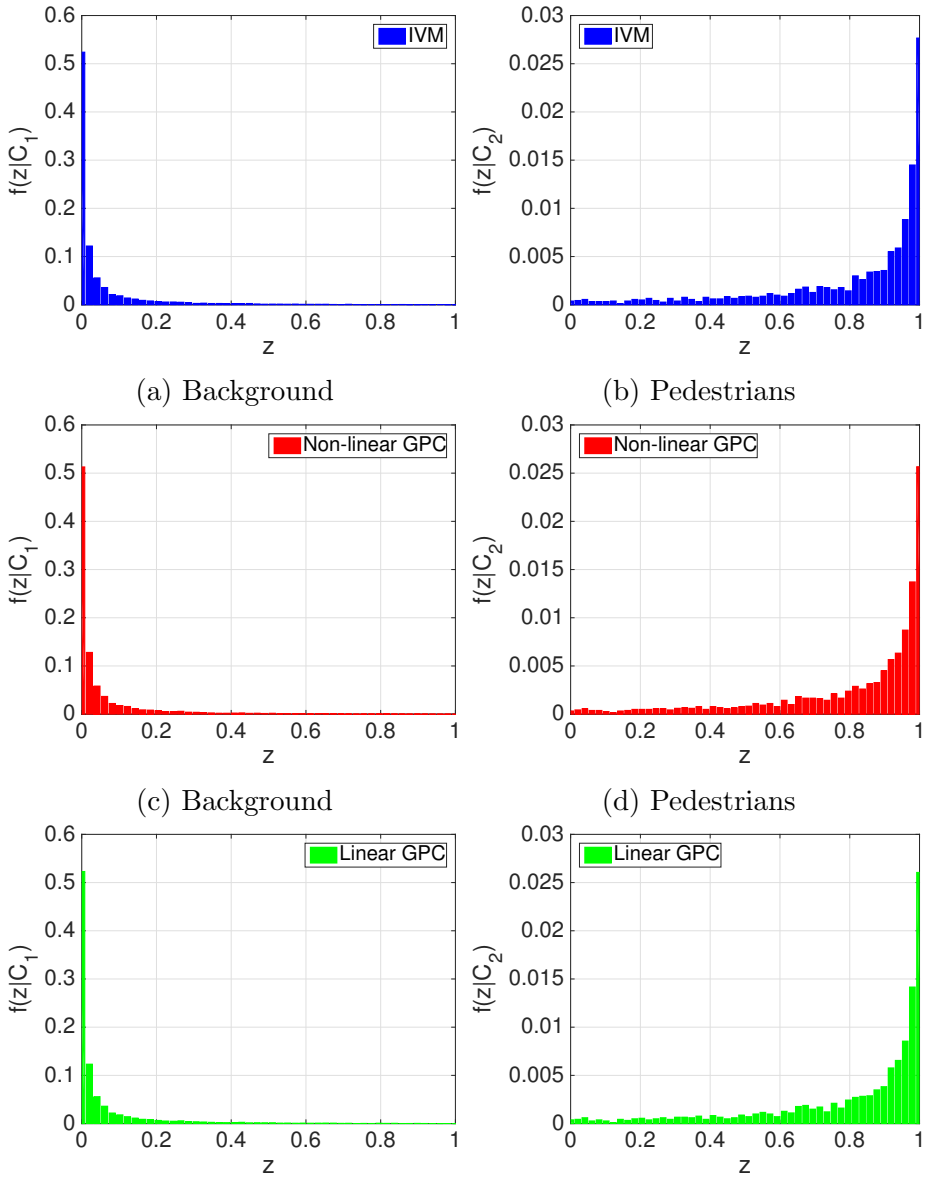
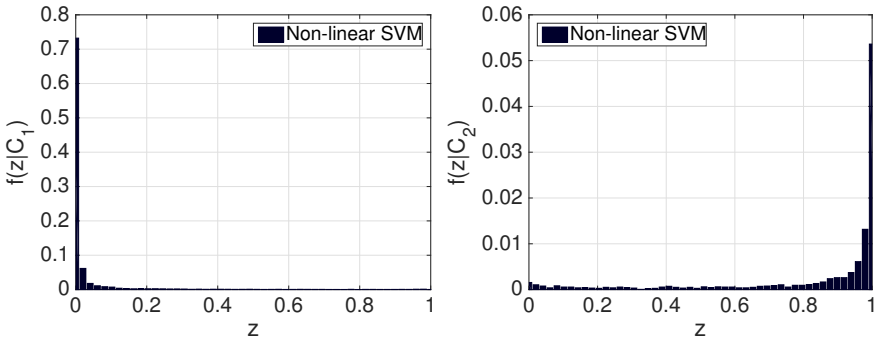
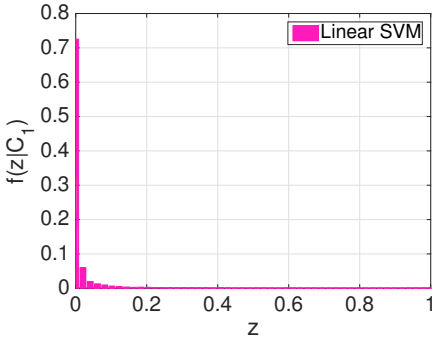


Figure D.1: The empirical probability density functions of the classifiers on a single run from the DP data set. The density functions for C_1 (background) show 16,000 samples and the density functions for C_2 show 2,000 samples, each across 50 equally-sized bins. See Figure D.2 for the other classifiers.

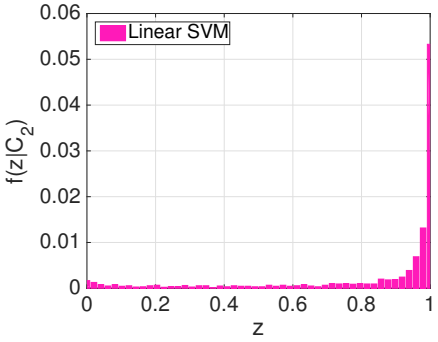


(a) Background

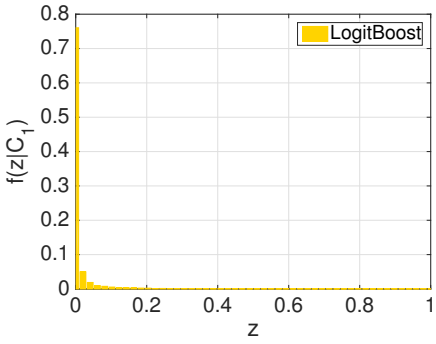
(b) Pedestrians



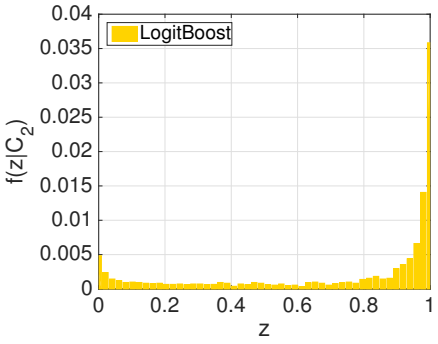
(c) Background



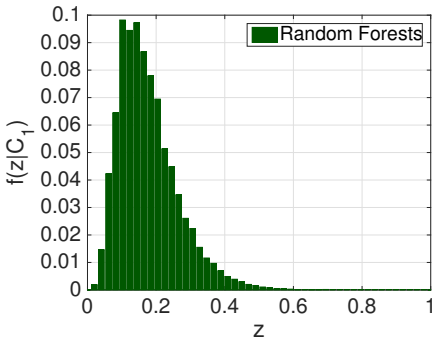
(d) Pedestrians



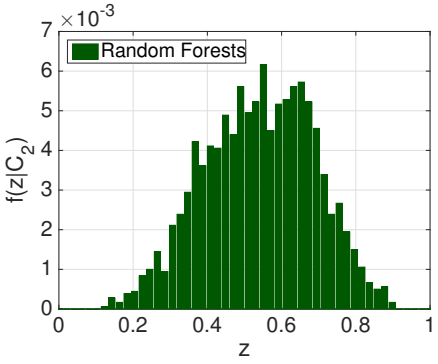
(e) Background



(f) Pedestrians



(g) Background



(h) Pedestrians

Figure D.2: The empirical probability density functions of the classifiers on a single run from the DP data set. The density functions for C_1 (background) show 16,000 samples and the density functions for C_2 show 2,000 samples, each across 50 equally-sized bins. See Figure D.1 for the other classifiers.

Appendix E

MDPs and POMDPs

In this appendix we describe the workings of two sequential decision making models, starting with the Markov Decision Process (MDP) and then adding the complexity of stochastic sensor measurements with the Partially Observable MDP (POMDP). These processes can be used to model a wide body of problems.

E.1 Markov Decision Process (MDP)

A Markov Decision Process can be defined by the following [Sutton and Barto, 1998]:

- a set of *states* $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$ in which the agent can be,
- a set of *actions* $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ that the agent can choose to perform,
- the *state transition distribution function* $p(s' \mid s, a)$ which describes the likelihood of ending up in state s' if the agent starts in state s and performs action a, and
- an *expected immediate reward function* $r(s, a, s')$ which defines the immediate reward given to the agent for performing action a in state s, and ending in state s'.

We will restrict ourselves to discrete time planning, where the state can change at each time step t. The general problem is to find a suitable *policy* π which maps state to action:

$$\pi : s_t, s_{t-1}, \ldots, a_{t-1}, a_{t-2}, \cdots \rightarrow a. \tag{E.1}$$

In the MDP, the agent starts in a known state s_0, chooses some action a_0 and then is told which resulting state $s'_0 = s_1$ it now finds itself. The state is fully observable. We denote the *reward* at time t as R_t. The goal is to maximise the total reward.

Central to the MDP is the assumption of the Markov property, being that the state transition distribution depends only on the chosen action and the current state, not any previous states or actions. Applying the Markov property to our generalised planning problem allows us to restrict ourselves to policies

$$\pi : s \rightarrow a \tag{E.2}$$

that only depend upon the current state. Thus at any time t, the agent looks up $\pi(s_t)$ and performs that action. Choosing a policy is not as simple as choosing the action which maximises the immediate reward, because that greedy choice may land the agent in a state for which all future rewards are smaller than the alternative, and thus the total reward over the course of its life are very likely to be suboptimal. If the state transition distribution is deterministic we can sum up the immediate reward over all future states, but this could be infinite, and thus make it impossible to compare policies. If the state transition distribution is stochastic, we can instead calculate an expected total reward, but again, that could be infinite. So how do we compare policies yielding infinite rewards? Two common approaches are to discount future reward by a constant factor $\gamma \in [0, 1)$, or to use a finite horizon, where we total the expected future reward for a fixed number of time-steps into the future [Sutton and Barto, 1998]. Both of these methods result in the rewards becoming

finite, and therefore comparable, in infinite-horizon tasks (where the number of time steps is infinite) or indefinite-horizon tasks (where the number of time steps *may* be infinite, but is generally finite). Discounting future reward is the approach used by the framework we use for this work (the SARSOP point-based POMDP solver, under the APPL toolkit by Kurniawati et al. [2008]).

The *total discounted future reward* at time t is called the *return*, G_t, defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{E.3}$$

Now we can define the *state-value function* v for a particular policy π, which tells us the expected return that the agent receives if it follows that policy from each state, thus:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]. \tag{E.4}$$

Note that $v_\pi(s)$ depends upon s but not on t. One convenient property of this value function is that it is recursive, relating the value of being in a state s to the value of being in all possible next states s':

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] && \text{(E.5)} \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] && \text{(E.6)} \\ &= \mathbb{E}_\pi\left[R_{t+1} + \sum_{k=0}^{\infty} \gamma^{k+1} R_{t+k+2} \middle| S_t = s\right]. && \text{(E.7)} \end{aligned}$$

To calculate $\mathbb{E}_\pi[R_{t+1}]$ we must sum the immediate rewards, starting in state s, taking action a, and leading to the new state s', weighted by the likelihood of the triplet (s, a, s') occurring. That is,

$$\mathbb{E}_\pi[R_{t+1}] = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \cdot r(s, a, s'), \tag{E.8}$$

which we can substitute into (E.7) to obtain Bellman's equation [Sutton and Barto, 1998]:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \left[r(s, a, s') + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s \right] \right] \tag{E.9}$$

$$= \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v_\pi(s') \Big] \tag{E.10}$$

in the case where the policy is stochastic. From now on we assume the policy to be deterministic. This may be expressed in the form

$$v_\pi(s) = \underbrace{r\big(s, \pi(s)\big)}_{\text{immediate reward}} + \overbrace{\gamma \sum_{s' \in \mathcal{S}} p\big(s' \mid s, \pi(s)\big) \cdot v_\pi(s')}^{\text{discounted future reward}}, \tag{E.11}$$

where $r(s, a)$ is the immediate reward of being in state s and taking action a, so that

$$r(s, a) = \sum_{s' \in \mathcal{S}} p(s' \mid s, a) r(s, a, s'). \tag{E.12}$$

Now that we have defined the value at a state for a given policy, we are in a position to compare the values of different policies. If we have two policies π and π', we write $\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s)\ \forall s \in \mathcal{S}$. It is a principal result that there exist one or more policies π_* satisfying $\pi_* \geq \pi$ for all policies π [Sutton and Barto, 1998], and any such π_* is called an *optimal policy*. All optimal policies have the same state-value function, called the *optimal state-value function*, v_*, which satisfies *Bellman's optimality equation*,

$$v_*(s) = \max_\pi \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \Big[r(s, a, s') + \gamma v_*(s') \Big]. \tag{E.13}$$

Finding the optimal policy using *value iteration*

Given the optimal state-value function v_*, the optimal policy may be found by

$$\pi_* = \arg\max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s', s, a) v_*(s') \right). \tag{E.14}$$

One way to compute $\pi_*(s)$ is to solve (E.13) for $v_*(s)$, but this can be very difficult when there are a substantial number of states and actions [Shani et al., 2013]. Another way of calculating $v_*(s)$ is to use the process of *value iteration*, as follows. Let $v(s)$ be the value function associated with policy π. If you take action a and subsequently follow policy π, the expected return is

$$\mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v(s'). \tag{E.15}$$

If we choose a greedily, such that this is a maximum, we obtain a new value function $v'(s)$ given by

$$v'(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v(s') \right). \tag{E.16}$$

We write $v' = J(v)$ where J is a function, or an operator, which maps state-value functions to state-value functions. Hauskrecht [2000] explains that J has a property of being a *contraction*, and this implies that successive value-iteration converges to the optimal value function. In practice we cannot run value iteration forever, so typically we continue until the difference in value between two successive iterations is less than some predetermined small $\epsilon(1-\gamma)/\gamma$, by which point the distance to the optimal value function is bounded by ϵ. This bound is shown not to increase with subsequent iterations.

While the MDP is extremely effective in a number of problems (e.g. Dean et al. [1993], Laroche et al. [1999]), it assumes that we have full observability over the state, even if the transition function or policy are stochastic. In practice, our robots

will often not have this; real sensors are noisy, and so the state could be more realistically modelled as only partially observable. The need for agents which can generate useful policies in the absence of full observability has prompted work into partially observable MDPs, or POMDPs, which is described in the following section and inherits much from the theory of MDPs presented above.

E.2 Partially Observable Markov Decision Processes (POMDP)

The POMDP can be seen as an extension of the MDP, removing the assumption that the agent's state is fully observable [Pineau et al., 2006]. Instead, the agent maintains a probability distribution over possible states, called the *belief* or *belief state*, $b(s)$. At each time step, the agent receives some *measurement* or *observation* z which it uses to update its belief state.

A POMDP model is defined by the four components from the MDP in Appendix E.1, plus three more:

- a list of possible observations $\mathcal{Z} = \{z_1, z_2, \ldots, z_{|\mathcal{Z}|}\}$, and
- the *observation probability distribution* $p(z \mid a, s')$. This describes the likelihood of seeing observation z if the agent is in state s' having performed action a. Lastly,
- the agent's initial belief state.

Whereas in the MDP the number of states can be finite, the belief can be any probability distribution over the states, and there are infinitely many such distributions. We refer to this space of possible probability distributions as the *belief space*. As a sequence of observations are made, the current belief state of the agent evolves

within the belief space, and as it does so it maintains the Markov property. That is, the current belief state is sufficient to determine the future belief states. Thus, given the initial belief state, the agent chooses an action, receives an observation, and updates its belief accordingly. This allows us to draw an equivalence between a POMDP and a continuous-space MDP, with the belief space acting as the state space of the MDP. As a result, we can make use of the theory described in the previous section.

POMDP Equations

Here we present the POMDP adaptations of the MDP equations discussed in Appendix E.1 [Pineau et al., 2006].

The value function update is defined as

$$v'(b) = \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s,a) b(s) + \gamma \sum_{z \in \mathcal{Z}} p(z \mid a, b) v\big(\tau(b,a,z)\big) \right], \tag{E.17}$$

where $\tau(b,a,z)$ is the new belief state b' having taken action a and received measurement z. It is the element-wise application of

$$\tau(b,a,z) \to b'(s' \mid b,a,z) = p(z \mid a, s') \frac{\sum_s p(s' \mid a, s) b(s)}{p(z \mid b, a)}, \tag{E.18}$$

over all $s' \in \mathcal{S}$. An element $b'(s')$ is the belief that the agent has transitioned to state s'. In practice we do not compute the denominator $p(z \mid b, a)$ explicitly and instead normalise the numerator of (E.18) such that $\sum_{s'} b'(s' \mid b, a, z) = 1$.

The policy given a value function v is therefore

$$\pi'(b) = \arg\max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s,a) b(s) + \gamma \sum_{z \in \mathcal{Z}} p(z \mid a, b) v\big(\tau(b,a,z)\big) \right]. \tag{E.19}$$

Further discussion can be found in Shani et al. [2013]. Next, we discuss the

implications of the fact that the belief space is infinite.

Vector representation of the value function, and *exact value iteration*

The state-value function of the MDP may be represented by a table with an entry for each state s. In contrast, a belief state $b(s)$ is continuous and so $v_\pi(b)$ lies in an $(|\mathcal{S}| - 1)$-dimensional space. The difficulty is therefore that the calculation of the value function at any step is not guaranteed to be possible in finite time or to be representable in finite memory. However, Smallwood and Sondik [1973] show that the value function can both be computed in finite time and represented in finite memory when expressed in terms of a convex, piecewise-linear interpolation. Specifically, they show that, after t iterations, the value function can be represented in terms of a finite set of vectors $\Gamma_t = \{\alpha_1, \alpha_2, \dots\}$ by the expression

$$v_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in \mathcal{S}} \alpha(s) b(s), \tag{E.20}$$

where $\alpha(s)$ is the s^{th} element of the vector α. This is illustrated in Figure E.1. In some cases a hyperplane defined by α can be dominated by one or more other hyperplanes, for instance the purple α_4 in the figure, in which case it can be removed from Γ_t. This representation gives rise to an iterative process called *exact value iteration*, and proceeds from step to step via an operation called a *backup* [Sondik, 1978, Cassandra et al., 1997].

The key idea of a backup is as follows [Pineau et al., 2006]. We update Γ_{t-1} by considering the immediate reward (or value) and discounted future value of each $\alpha_i \in \Gamma_{t-1}$, given some action a and observation z (similarly to (E.11)). Let $\Gamma_t^{a,*}$ be the set containing the single function $r(s, a)$ (defined in (E.12)), which we write in the form

$$\Gamma_t^{a,*} \leftarrow \alpha^{a,*}(s) = r(s, a). \tag{E.21}$$

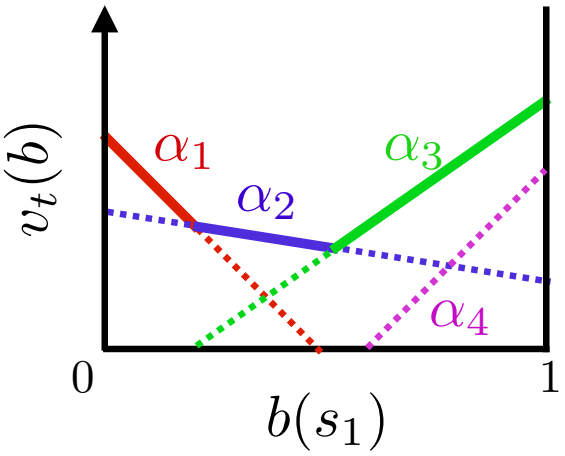


Figure E.1: Representing the value function v at iteration t by a piecewise-linear convex function $\Gamma_t = \{\alpha_1, \alpha_2, ...\}$. The value of a state $v(s)$ is the maximum over all linear functions evaluated at that point s, shown by the bold coloured lines. The dotted sections are redundant because they do not contribute to the maximum.

Similarly, we define the set $\Gamma_t^{a,z}$ by

$$\Gamma_t^{a,z} \leftarrow \alpha_i^{a,z}(s) = \gamma \sum_{s' \in \mathcal{S}} p(z \mid a, s') p(s' \mid s, a) \alpha_i(s'), \tag{E.22}$$

for $\alpha_i \in \Gamma_{t-1}$. Let

$$\Gamma_t^a = \Gamma_t^{a,*} + \Gamma_t^{a,z_1} \oplus \Gamma_t^{a,z_2} \oplus \ldots, \tag{E.23}$$

where $\oplus$ is the *cross-sum* operator, given by

$$A \oplus B = \{a_i + b_j : a_i \in A,\ b_j \in B\}. \tag{E.24}$$

Finally,

$$\Gamma_t = \bigcup_{a \in \mathcal{A}} \Gamma_t^a. \tag{E.25}$$

As t grows, the size of Γ_t grows exponentially, and so calculating the exact value iteration by this method becomes infeasible except in the case of very small problems.

Approximating the belief space, and *point-based value iteration*

Since exact value iteration is generally infeasible, an acceptable strategy is to select a finite set of belief points to represent the infinite belief space [Shani et al., 2013]. A naive approach could be to sample a grid of points over the space, but in most problems there is a section of the space to which no sequence of actions and observations can lead. This results in computational effort being wasted on estimating the value of these *unreachable* areas of the belief space. Therefore, today's point-based value iteration algorithms sample from the reachable parts of the belief space, exploring from the initial belief under arbitrary sequences of actions.

Throughout the experiments in this chapter we use a point-based value iteration solver called SARSOP, under the APPL toolkit by Kurniawati et al. [2008]. The key idea here is to maintain upper and lower bounds on the optimal value function, and through iteration to shrink the distance between these bounds until either they are within some predetermined value ϵ or a predetermined time limit elapses. In the results presented in Chapter 7, SARSOP typically terminates at the time limit.

Bibliography

The V-Charge project website. `http://www.v-charge.eu/`, 2015. Accessed: 2015-09-28.

The AdaptiVe project website. `https://www.adaptive-ip.eu/`, 2016. Accessed: 2016-01-27.

D. Abhishek, N. J. V. Raymond, and J. S. Luuk. Predicting Face Recognition Performance Using Image Quality. *CoRR*, abs/1510.07119, 2015.

A. Alessandrini, A. Cattivera, C. Holguin, and D. Stam. CityMobil2: Challenges and Opportunities of Fully Automated Mobility. In *Road Vehicle Automation*, pages 169–184. Springer, New York, NY, USA, 2014.

M. Aly. Real Time Detection of Lane Markers in Urban Streets. In *Intelligent Vehicles Symposium*, pages 7–12. IEEE, 2008.

D. Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1988.

A. Atrash, R. Kaplow, J. Villemure, R. West, H. Yamani, and J. Pineau. Development and Validation of a Robust Speech Interface for Improved Human-Robot Interaction. *International Journal of Social Robotics*, 1(4):345–356, 2009.

D. Barnes, W. Maddern, and I. Posner. Exploiting 3D Semantic Scene Priors for Online Traffic Light Interpretation. In *Intelligent Vehicles Symposium*, pages 573–578. IEEE, 2015.

Y. Bazi and F. Melgani. Gaussian Process Approach to Remote Sensing Image Classification. *Geoscience and Remote Sensing*, 48(1):186–197, 2010.

Ro. Benenson, R. Timofte, and L. Van Gool. Stixels Estimation Without Depth Map Computation. In *International Conference on Computer Vision Workshops*, pages 2010–2017. IEEE, 2011.

L.-P. Berczi, I. Posner, and T. D. Barfoot. Learning to Assess Terrain from Human Demonstration Using an Introspective Gaussian Process Classifier. In *International Conference on Robotics and Automation*, 2015.

L. F. Bertuccelli and J. P. How. Robust Markov Decision Processes using Sigma Point Sampling. In *American Control Conference*, pages 5003–5008. IEEE, 2008.

C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, New York, NY, USA, 2006.

C. G. Blair, J. Thompson, and N. M. Robertson. Introspective Classification for Pedestrian Detection. In *Conference on Sensor Signal Processing for Defence*. IEEE, 2014.

E. P. Blasch, A. Lakhotia, and G. Seetharaman. Unmanned Vehicles Come of Age: The DARPA Grand Challenge. *Computer*, 39(12):26–29, 2006.

M. Boshra and B. Bhanu. Predicting Performance of Object Recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 22(9):956–969, 2000.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

E. Brynjolfsson, Y. Hu, and M. D. Smith. Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers. *Management Science*, 49(11):1580–1596, 2003.

C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Y.-D. Cai, K.-Y. Feng, W.-C. Lu, and K.-C. Chou. Using LogitBoost Classifier to Predict Protein Structural Classes. *Journal of Theoretical Biology*, 238(1): 172–176, 2006.

A. R. Cassandra, M. L. Littman, and N. L. Zhang. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. *The Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61, 1997.

C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 4(1):129–145, 1996.

T. M. Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *Electronic Computers*, EC-14(3):326–334, 1965.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, USA, 2nd edition, 2006.

C. Cusano, G. Ciocca, and R. Schettini. Image Annotation Using SVM. In *Electronic Imaging*, pages 330–338. International Society for Optics and Photonics, 2003.

D. R. Cutler, T. C. Edwards Jr, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, and J. J. Lawler. Random Forests for Classification in Ecology. *Ecology*, 88(11): 2783–2792, 2007.

N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893. IEEE, 2005.

A. P. Dawid. The Well-Calibrated Bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

T. L. Dean, L. P. Kaelbling, J. Kirman, and A. E. Nicholson. Planning With Deadlines in Stochastic Domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, volume 93, pages 574–579. AAAI, 1993.

M. H. DeGroot and S. E. Fienberg. The Comparison and Evaluation of Forecasters. *The Statistician*, pages 12–22, 1983.

M. Dettling and P. Bühlmann. Boosting for Tumor Classification with Gene Expression Data. *Bioinformatics*, 19(9):1061–1069, 2003.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, NY, USA, 1986.

C. Dima, M. Hebert, and A. Stentz. Enabling Learning from Large Datasets: Applying Active Learning to Mobile Robotics. In *International Conference on Robotics and Automation*, volume 1, pages 108–114. IEEE, 2004.

C. Dubout and F. Fleuret. Exact Acceleration of Linear Object Detectors. In *Computer Vision*, pages 301–311. Springer, New York, NY, USA, 2012.

H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping: Part I. *Robotics & Automation Magazine*, 13(2):99–110, 2006.

D. Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge, 2014.

L. El Ghaoui and A. Nilim. Robust Solutions to Markov Decision Problems with Uncertain Transition Matrices. *Operations Research*, 53(5), 2005.

E. E. Elattar. Day-ahead Price Forecasting of Electricity Markets Based on Local Informative Vector Machine. *IET Generation, Transmission & Distribution*, 7 (10):1063–1071, 2013.

M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-Cue Pedestrian Classification with Partial Occlusion Handling. In *Conference on Computer Vision and Pattern Recognition*, pages 990–997. IEEE, 2010.

M. Enzweiler, M. Hummel, D. Pfeiffer, and U. Franke. Efficient Stixel-Based Object Recognition. In *Intelligent Vehicles Symposium*, pages 1066–1071. IEEE, 2012.

E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie. Mismatch String Kernels for SVM Protein Classification. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2002.

N. Fairfield and C. Urmson. Traffic Light Mapping and Detection. In *International Conference on Robotics and Automation*, pages 5421–5426. IEEE, 2011.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective Sampling using the Query by Committee Algorithm. *Journal of Machine Learning*, 28(2):133–168, 1997.

J. Friedman, T. J. Hastie, and R. J. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 28:2000, 1998.

A. Furda and L. Vlacic. Enabling Safe Autonomous Driving in Real-world City Traffic Using Multiple Criteria Decision Making. *Intelligent Transportation Systems Magazine*, 3(1):4–17, 2011.

P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, Timpner. J., S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, Stellmacher. M., H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart. Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project. In *Intelligent Vehicles Symposium*, pages 809–816, Gold Coast, Australia, 2013. IEEE.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. 2015. URL `http://arxiv.org/abs/1506.02142`.

D. Garrett, D. A. Peterson, C. W. Anderson, and M. H. Thaut. Comparison of Linear, Nonlinear, and Feature Selection Methods for EEG Signal Classification. *Neural Systems and Rehabilitation Engineering*, 11(2):141–144, 2003.

A. Geiger, P. Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

D. Gerónimo, A. D. Sappa, D. Ponsa, and A. M. López. 2D–3D-based On-board Pedestrian Detection System. *Computer Vision and Image Understanding*, 114 (5):583–595, 2010.

P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson. Random Forests for Land Cover Classification. *Pattern Recognition Letters*, 27(4):294–300, 2006.

J. González, M. Osborne, and N. D. Lawrence. GLASSES: Relieving The Myopia Of Bayesian Optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. JMLR, 2016.

H. Grimmett, R. Paul, R. Triebel, and I. Posner. Knowing When We Don't Know: Introspective Classification for Mission-Critical Decision Making. In *International Conference on Robotics and Automation*, pages 4351–4358. IEEE, 2013.

H. Grimmett, M. Buerki, L. Paz, P. Piniés, P. Furgale, I. Posner, and P. Newman. Integrating Metric and Semantic Maps for Vision-Only Automated Parking. In *Proceedings of the International Conference on Robotics and Automation*, Seattle, WA, USA, 2015a.

H. Grimmett, R. Triebel, R. Paul, and I. Posner. Introspective Classification for Robot Perception. *International Journal of Robotics Research*, 2015b. Corrigendum-ibid. Grimmett et al. [To appear 2016].

H. Grimmett, R. Triebel, R. Paul, and I. Posner. Corrigendum: Introspective Classification for Robot Perception. *International Journal of Robotics Research*, To appear 2016.

P. Grother and E. Tabassi. Performance of Biometric Quality Measures. *Pattern Analysis and Machine Intelligence*, 29(4):531–543, 2007.

R. Harb, X. Yan, E. Radwan, and X. Su. Exploring Precrash Maneuvers Using Classification Trees and Random Forests. *Accident Analysis & Prevention*, 41(1): 98–107, 2009.

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC, 1990.

M. Hauskrecht. Value-Function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

M. Hauskrecht and H. Fraser. Planning Treatment of Ischemic Heart Disease with Partially Observable Markov Decision Processes. *Artificial Intelligence in Medicine*, 18(3):221–244, 2000.

J. Hoey and P. Poupart. Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *International Joint Conference on Artificial Intelligence*, pages 1332–1338, 2005.

A. Holub, P. Perona, and M. C. Burl. Entropy-Based Active Learning for Object Recognition. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

T. Hospedales, S. Gong, and T. Xiang. Finding Rare Classes: Active Learning with Generative and Discriminative Models. *Transactions on Knowledge and Data Engineering*, 25(2):374–386, 2013.

K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez. Grasping POMDPs. In *International Conference on Robotics and Automation*, pages 4685–4692. IEEE, 2007.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. *A Practical Guide to Support Vector Classification*. 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

A. Huang and S. Teller. Probabilistic Lane Estimation using Basis Curves. In *Robotics: Science and Systems VI*, Zaragoza, Spain, 2010.

J. Huang, X. Shao, and H. Wechsler. Face Pose Discrimination using Support Vector Machines (SVM). In *International Conference on Pattern Recognition*, volume 1, pages 154–156. IEEE, 1998.

A. Ibisch, S. Houben, M. Schlipsing, R. Kesten, P. Reimche, F. Schuller, and H. Altinger. Towards Highly Automated Driving in a Parking Garage: General Object Localization and Tracking using an Environment-embedded Camera System. In *Intelligent Vehicles Symposium Proceedings*, pages 426–431. IEEE, 2014.

R. Jaulmes, J. Pineau, and D. Precup. *Active Learning in Partially Observable Markov Decision Processes*. Springer, New York, NY, USA, 2005.

A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class Active Learning for Image Classification. In *Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos. Scalable Active Learning for Multiclass Image Classification. *Pattern Analysis and Machine Intelligence*, 34 (11):2259–2273, 2012.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1):99–134, 1998.

Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. Hundelshausen, et al. Team AnnieWAY's Autonomous System for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9): 615–639, 2008.

A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian Processes for Object Categorization. *International Journal of Computer Vision*, 88(2):169–188, 2010.

S. S. Keerthi and C.-J. Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15(7):1667–1689, 2003.

A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the Damage of Dataset Bias. In *Computer Vision–ECCV 2012*, pages 158–171. Springer, New York, NY, USA, 2012.

S. Kotsiantis, E. Athanasopoulou, and P. Pintelas. Logitboost of Multinomial Bayesian Classifier for Text Classification. *International Review on Computers and Software*, 1(3):243–250, 2006.

A. Krizhevsky, I. Sutskever, and G. Hinton. PImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

J. Kulick, M. Toussaint, T. Lang, and M. Lopes. Active Learning for Teaching a Robot Grounded Relational Symbols. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1451–1457, Menlo Park, California, 2013. AAAI Press.

H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, 2008.

P. Laroche, F. Charpillet, and R. Schott. Mobile Robotics Planning using Abstract Markov Decision Processes. In *International Conference on Tools with Artificial Intelligence*, pages 299–306. IEEE, 1999.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK, 2006.

N. D. Lawrence. MLTOOLS, 2012. URL `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/mltools/`.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. *Advances in Neural Information Processing Systems*, 15:609–616, 2002.

N. D. Lawrence, J. C. Platt, and M. I. Jordan. Extensions of the Informative Vector Machine. In *First International Conference on Deterministic and Statistical Methods in Machine Learning*, pages 56–87, New York, NY, USA, 2005. Springer.

D. D. Lewis and W. A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, New York, NY, USA, 1994. Springer.

L. Li, M. L. Littman, and T. J. Walsh. Knows What It Knows: A Framework for Self-aware Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 568–575, New York, NY, USA, 2008. ACM.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A Note on Platt's Probabilistic Outputs for Support Vector Machines. *Journal of Machine Learning*, 68(3):267–276, 2007.

B. Lu, D. Gu, H. Hu, and K. McDonald-Maier. Sparse Gaussian Process for Spatial Function Estimation with Mobile Sensor Networks. In *Emerging Security Technologies (EST), 2012 Third International Conference on*, pages 145–148. IEEE, 2012.

D. J. C. MacKay. Comparison of Approximate Methods for Handling Hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.

D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

S. Marsland, U. Nehmzow, and J. Shapiro. On-line Novelty Detection for Autonomous Mobile Robots. *Robotics and Autonomous Systems*, 51(2):191–206, 2005.

A. McCallum and K. Nigam. Employing EM in Pool-Based Active Learning for Text Classification. In *International Conference on Machine Learning*, pages 350–358, 1998.

D. Meger, P. E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe. Curious George: An Attentive Semantic Robot. *Robotics and Autonomous Systems*, 56(6):503–511, 2008.

T. P. Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, Burlington, MA, USA, 2001. Morgan Kaufmann Publishers Inc.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. 2013. URL `http://arXiv.org/abs/1312.5602`.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, USA, 2012.

A. Ng and M. Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. *Advances in Neural Information Processing Systems*, 14:841, 2002.

A. Niculescu-Mizil and R. Caruana. Predicting Good Probabilities with Supervised

Learning. In *The 22nd International Conference on Machine Learning*, pages 625–632, New York, NY, USA, 2005. ACM.

Robotics Centre of Mines ParisTech. Traffic Lights Recognition (TLR) Data Set. 2010. URL `http://www.lara.prd.fr/benchmarks/trafficlightsrecognition`.

F. A. Oliehoek, J. F. P. Kooij, N. Vlassis, et al. The Cross-Entropy Method for Policy Search in Decentralized POMDPs. *Informatica*, 32(4):341–357, 2008.

E. Osuna, R. Freund, and F. Girosi. Training Support Vector Machines: an Application to Face Detection. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136. IEEE, 1997.

S. G. Pauker and J. P. Kassirer. The Threshold Approach to Clinical Decision Making. *New England Journal of Medicine*, 302(20):1109–1117, 1980.

R. Paul, R. Triebel, D. Rus, and P. Newman. Semantic Categorization of Outdoor Scenes with Uncertainty Estimates using Multi-Class Gaussian Process Classification. In *Intelligent Robots and Systems*, pages 2404–2410. IEEE, 2012.

J. Pearl. *Do We Need Higher-order Probabilities And, If So, what Do They Mean?* UCLA, Computer Science Department, 1987.

J. Pineau, G. Gordon, and S. Thrun. Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, pages 335–380, 2006.

C. Plagemann, D. Fox, and W. Burgard. Efficient Failure Detection on Mobile Robots Using Particle Filters with Gaussian Process Proposals. In *International Joint Conference on Artificial Intelligence*, pages 2185–2190, 2007.

J. C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances In Large Margin Classifiers*, pages 61–74, Cambridge, MA, USA, 1999. The MIT Press.

D. M. Powers. Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *International Journal of Machine Learning Technology*, 2:37–63, 2011.

C. E. Rasmussen. The Infinite Gaussian Mixture Model. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Neural Information Processing Systems 12*, pages 554–560. The MIT Press, 2000.

C. E. Rasmussen and H. Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. *The MIT Press*, 2006.

S. Rosenthal, M. M. Veloso, and A. K. Dey. Task Behavior and Interaction Planning for a Mobile Service Robot that Occasionally Requires Help. In *Automated Action Planning for Autonomous Mobile Robots*, 2011.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, London, UK, 2nd edition, 2003.

A. Sayedi, M. Zadimoghaddam, and A. Blum. Trading off Mistakes and Don't-Know Predictions. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2092–2100. Curran Associates, Inc., 2010.

F. Schroff, A. Criminisi, and A. Zisserman. Object Class Segmentation using Random Forests. In *British Machine Vision Conference*, pages 1–10, 2008.

C. Schüldt, I. Laptev, and B. Caputo. Recognizing Human Actions: a Local SVM Approach. In *International Conference on Pattern Recognition*, volume 3, pages 32–36. IEEE, 2004.

M. Schulze, G. Nocker, and K. Bohm. PReVENT: A European Program to Improve Active Safety. In *International Conference on Intelligent Transportation Systems Telecommunications, France*, 2005.

Y-W Seo, N. D. Ratliff, and C. Urmson. Self-Supervised Aerial Image Analysis for Extracting Parking Lot Structure. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1837–1842, Menlo Park, California, USA, 2009. AAAI Press.

G. Shani, J. Pineau, and R. Kaplow. A Survey of Point-Based POMDP Solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics: Science and Systems V*, page paper 23, 2009.

G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-Scale Outdoor Navigation using Adaptive Relative Bundle Adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

R. D. Smallwood and E. J. Sondik. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research*, 21(5):1071–1088, 1973.

T. Smith and R. Simmons. Heuristic Search Value Iteration for POMDPs. In *The 20th Conference on Uncertainty in Artificial Intelligence*, Uncertainty in Artificial Intelligence, pages 520–527, Arlington, VA, USA, 2004. AUAI Press.

E. J. Sondik. The Optimal Control of Partially Observable Markov Processes. Technical report, DTIC Document, 1971.

E. J. Sondik. The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs. *Operations Research*, 26(2):282–304, 1978.

M. Spaan and N. Vlassis. A Point-Based POMDP Algorithm for Robot Planning. In *International Conference on Robotics and Automation*, volume 3, pages 2399–2404. IEEE, 2004.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Networks*, 32:323–332, 2012.

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. The MIT Press, Cambridge, MA, USA, 1998.

E. Tabassi, C. Wilson, and C. Watson. NIST Fingerprint Image Quality. *NIST Internal Report 7151*, pages 34–36, 2004.

S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy. Toward Information Theoretic Human-Robot Dialog. In *Robotics: Science and Systems*, page 409, Cambridge, MA, USA, 2013. The MIT Press.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

A. Torralba and A. A. Efros. Unbiased Look at Dataset Bias. In *Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE, 2011.

A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *Pattern Analysis and Machince Intelligence*, 29(5):854–869, 2007.

R. Triebel, H. Grimmett, R. Paul, and I Posner. Driven Learning for Driving: How Introspection Improves Semantic Mapping. In *International Symposium on Robotics Research*, Singapore, 2013.

C. Urmson, A. Anhalt, J. A. Bagnell, C. R. Baker, R. E. Bittner, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, H. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y-W Seo, R. Simmons, S. Singh, J. M. Snider, S. Stentz, W. L. Whittaker, and J. Ziglar. Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge. Technical report, Robotics Institute, Pittsburgh, PA, USA, April 2007.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M.N. Clark, J. Dolan,

D. Duggins, T. Galatali, C. Geyer, et al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

V. N. Vapnik and V. Vapnik. *Statistical Learning Theory*, volume 1. Wiley, New York, NY, USA, 1998.

A. Vedaldi and B. Fulkerson. VLFeat – An Open and Portable Library of Computer Vision Algorithms. In *ACM International Conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM.

J. Velez, G. Hemann, A. S. Huang, I. Posner, and N. Roy. Planning to Perceive: Exploiting Mobility for Robust Object Detection. In F. Bacchus, C. Domshlak, S. Edelkamp, and M. Helmert, editors, *The International Conference on Automated Planning and Scheduling*, pages 266–273, Menlo Park, California, 2011. The AAAI Press.

G. Wachman, R. Khardon, P. Protopapas, and C. R. Alcock. Kernels for Periodic Time Series Arising in Astronomy. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 489–505, New York, NY, USA, 2009. Springer.

J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep Learning via Semi-supervised Embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, New York, NY, USA, 2012.

C. K. I. Williams and D. Barber. Bayesian Classification with Gaussian Processes. *Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *Evolutionary Computation*, 1(1):67–82, 1997.

D. H. Wolpert and W. G. Macready. Coevolutionary Free Lunches. *Evolutionary Computation*, 9(6):721–735, 2005.

F. Zaklouta, B. Stanciulescu, and O. Hamdoun. Traffic Sign Classification Using KD Trees and Random Forests. In *International Joint Conference on Neural Networks*, pages 2151–2155. IEEE, 2011.

P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting Failures of Vision Systems. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3566–3573. IEEE, 2014.

W. Zhang, X. Stella, and Y. S. Teng. Power SVM: Generalization with Exemplar Classification Uncertainty. In *Conference on Computer Vision and Pattern Recognition*, pages 2144–2151. IEEE, 2012.

J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. Making Bertha Drive An Autonomous Journey on a Historic Route. *Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.

B. Ziólko, S. Manandhar, R. C. Wilson, and M. Ziólko. Logitboost WEKA Classifier Speech Segmentation. In *International Conference on Multimedia and Expo*, pages 1297–1300. IEEE, 2008.