Sequential Forecasting and Decision Making in Dynamic and Incomplete Environments

Seung Min LEE
St. Edmund Hall

Robotics Research Group
Department of Engineering Science
University of Oxford

Hilary Term 2009
Sequential Forecasting and Decision Making in Dynamic and Incomplete Environments

Abstract

In many real-world data analysis problems observations arrive sequentially in time and it is required to perform inference on-line. Sequential learning provides us with techniques to fuse information, learn policies, analyse risks, forecast outcomes and make decisions in such a way that a current model is updated as new information becomes available. This framework of sequential learning is particularly important because the environment we are faced with changes over time and is often partially missing. It is therefore necessary to develop methods that perform consistently on such occasions. This thesis focuses on the problem of sequential forecasting, decision and action making in dynamic and incomplete environments. To solve this problem we adopt a dynamic model framework; we discuss five sub-classes of the dynamic model: (1) dynamic linear models, (2) dynamic non-linear models, (3) dynamic generalised linear models, (4) Markov decision processes and (5) delayed reward, reinforcement learning, models. We apply these models to real-world dynamic forecasting and decision making problems, in which information is incompletely known and/or observed.
# Contents

1 Introduction 1
   1.1 Motivation .................................................. 1
   1.2 Problem Formulation ........................................ 4
   1.3 Thesis Overview ............................................ 7

2 Dynamic Linear Models 9
   2.1 Theoretical Background .................................... 9
   2.2 The Kalman Filter ........................................... 16
   2.3 Smoothing and Forecasting ................................ 22
   2.4 Non-linear Kalman Filters ................................ 25

3 Dynamic Generalised Linear Models 36
   3.1 Generalised Linear Models .................................. 36
   3.2 Exponential Family Dynamic Models ....................... 38

4 Making a Sequence of Optimal Decisions 45
   4.1 Markov Decision Processes ................................ 45
   4.2 Reinforcement Learning ..................................... 50
   4.3 Decision Making on Continuous Spaces Using Radial Basis Functions ........ 56
   4.4 Finding a Path in Non-stationary Environments ............ 66

5 Multivariate Time Series Forecasting in Incomplete Environments 72
   5.1 Learning Dynamic Linear Models ............................. 73
   5.2 Multivariate Time Series Models ............................ 77
   5.3 Results ..................................................... 83

6 Sequential Classification Using Dynamic Models 99
   6.1 Dynamic Logistic Regression ............................... 100
   6.2 Dynamic Classification using the Dynamic Binomial Model ... 106
   6.3 Making use of Past Information ............................. 110
   6.4 Results ..................................................... 111
# Conclusions and Future Work

7.1 Conclusions .................................................. 129
7.2 Future Extensions ............................................. 132

## Derivations

A.1 Derivation of Equation (3.31) .................................. 135
A.2 Derivation of Equation (6.48): Conjugate prior of a binomial .... 135
A.3 Derivation of Equation (6.53): One-step forecast distribution of a binomial 137
# List of Figures

1.1 The structure of a distributed and decentralised information network system. 3

4.1 A diagram of a Markov decision process. 47

4.2 The mountain-car task. 61

4.3 Five radial Gaussian functions over the position and velocity spaces, with the basis function width of 0.1 (position) and 0.01 (velocity). 62

4.4 Value functions $\max_{a_t} Q(w_t, a_t)$ estimated after 1 (first row), 10 (second row), 100 (third row) and 1,000 (fourth row) episodes for RBF-S (left column), RBF-A (middle column) and Tile Coding (right column). The estimated value functions of RBF-S and RBF-A rightly reflect the characteristics of the mountain-car task: they show high values (dark red) for high position and high velocity, and low values (dark blue) for middle position and low velocity. Meanwhile, the estimated value function of Tile Coding does not have these characteristics. 64

4.5 (Top row) The number of steps taken to the goal over 500 episodes. (Middle row) Value functions estimated after 500 episodes. (Bottom row) Policy functions: full power forward (red), full power reverse (blue), zero power (yellow). The policy function of Tile Coding does not reflect the characteristics of the mountain-car problem. 65

4.6 Value functions (top row) and policy functions (bottom row) obtained after 500 episodes for RBF-A (left column) and cRBF-A (right column) when the car’s starting position and velocity are randomly selected in every episode. 67

4.7 A shortest path problem. We are interested in finding a route from ‘S’ to ‘G’ after discovering three hidden areas coloured grey. 68

4.8 Paths found after 2,000 episodes by the naive approach (left) and the novel approach(right). Since the novel approach consider the change of the environment, it successfully found the shorted path. 70

5.1 Annual surface temperature anomaly relative to 1951-1980 mean, based on surface air measurements at meteorological stations and ship and satellite measurements of sea surface temperature. 84
5.2 Predicted values for annual surface temperature anomaly by a DMAR model of an AR order: (a) $p = 1$, (b) $p = 5$, (c) $p = 10$ and (d) $p = 20$. Observed values are represented by a black line, and predictions by a red line. Shaded areas depict a 95% confidence region of the predictions. DMAR models with a higher AR order predicted temperature forecasts showing an upward trend.

5.3 Predicted values for annual surface temperature anomaly by a MLT model with a data window of size (a) $n = 50$ and (c) $n = 80$. (b) and (d) depict time plots of estimated local trends for MLT($n = 50$) and MLT($n = 80$) respectively; that is, each represents an estimate annual change rate of temperature. A MLT model considering a larger number of past data (corresponding (c) and (d)) appears to estimate more realistic annual temperature changes.

5.4 Predicted values for annual surface temperature anomaly by a GP model with a covariance function having (a) one squared exponential term (b) two squared exponential terms. A GP model with two squared exponential terms (one for local disturbances and the other for a trend) found a mildly increasing trend.

5.5 The Bramble Bank weather station and associated web site (see www.bramblemet.co.uk). This figure is provided by Osborne et al. (2008c).

5.6 (a) Air temperature data collected in real time by a network of weather sensors located at Chichester Bar (red), Southampton Dockhead (green) and Bramble Pile (blue). (b) True air temperature readings recovered afterwards. Five grey regions are of our particular interest as these are periods when missing values regularly occurred.

5.7 Predicted values for three sensors by a DMAR model of AR order (a) $p = 1$ and (b) $p = 15$. Pink areas represent a 95% confidence region of the predictions. A DMAR model with a higher AR order performed better particularly in the second interest region. However, both models failed to predict a daily cyclical pattern of air temperature as shown in the third and fourth grey areas.

5.8 Predicted values for three sensors by a MLT model. Despite partial missing values being well-predicted, the MLT model, which is based on a first-order Markov model, failed to capture a daily cyclical pattern of air temperature.

5.9 Predicted values for three sensors by a GP model. Since the GP model incorporated prior knowledge about the data such as a daily cyclical pattern into a covariance function, it successfully predicted missing values even when all sensors were missing.

5.10 A time series having a daily periodic cycle.
5.11 Predicted values for three sensors by (a) the DMAR\((p = 1)\), (b) the DMAR\((p = 15)\) and (c) the MLT when cycle-augmented observations are used. By allowing the models to be informed of a daily cyclical pattern, we obtained much improved performances, particularly for the DMAR\((p = 15)\) model. It predicted very well the periodic pattern of the air temperature even when all sensors were missing.

6.1 Moderated posterior class probability for different sizes of the variance of activation.

6.2 Two Gaussian distributions rotating in a circular fashion over time.

6.3 Three data sets of two rotating Gaussians with different Bayes error.

6.4 Missing label experiments with variation in fractions of observed labels from 0\% to 100\% for the data sets with (a) 0\%, (b) 4\% and (c) 22\% Bayes errors. The classifiers maintained their classification performances with up to 70\% of labels missing.

6.5 Mountain fire scenario: a fire starts in a wooded mountain with five villages (denoted by red dots), which are in potential danger according to local weather conditions.

6.6 (Left panel) A 3-D plot of the three input variables and the class variable. (Right panel) A time plot of the class variable. The true labels alternate between ‘danger’ and ‘no danger’ at irregular intervals as the weather variables change. Note that we deal with a single village represented by a light blue ring in Figure 6.5.

6.7 Missing label experiment on the mountain fire data with variation in fractions of observed labels from 0\% to 100\%. The classification performances of the models did not worsen in proportion to the number of unobserved labels.

6.8 Active label requesting on the mountain fire data when 50\% of total labels is requested by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots. The grey regions represent time steps when the classifiers requested a label.

6.9 A time plot of (a) true inputs, (b) given inputs (70\% missing) and (c) predicted inputs. A dynamic multivariate autoregressive model of AR order 2 was used.

6.10 Missing input experiment on the mountain fire data with variation in fractions of observed inputs from 0\% to 100\%. The DLR-E model, in particular, performed very well with only 40\% of inputs available.
6.11 Classification predictions on the EEG data by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots.

6.12 Missing label experiment on the EEG data with variation in fractions of observed labels from 0% to 100%. The classification performances of the models remained high even with a small fraction of labelled data points.

6.13 Active label requesting on the EEG data when 50% of total labels is requested by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots. The grey regions represent time steps when the classifiers requested a label.

6.14 Missing input experiment on the EEG data with variation in fractions of observed inputs from 0% to 100%. The DLR-E model outperformed the DLR-U and DBM models: its performance reaches close to optimal even with a small fraction of input data points.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Three dynamic non-linear models used to test non-linear filters.</td>
<td>35</td>
</tr>
<tr>
<td>2.2</td>
<td>Root mean square error of the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the Chebyshev Kalman filter (CKF) on the three dynamic non-linear models.</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>The number of steps taken for the car to reach the goal position when the car’s starting position and velocity are commonly fixed to 0 in every episode.</td>
<td>62</td>
</tr>
<tr>
<td>4.2</td>
<td>The number of steps taken for the car to reach the goal position when the car’s starting position and velocity are commonly fixed to 0 in every episode. When the action space is assumed to be continuous (cRBF-A), we found a better policy that took the car to the goal in 73 steps.</td>
<td>66</td>
</tr>
<tr>
<td>5.1</td>
<td>Root mean square error of predictions by DMAR, MLT and GP models.</td>
<td>97</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison between three multivariate forecasting models.</td>
<td>98</td>
</tr>
<tr>
<td>6.1</td>
<td>Proportion of correct classification. (Top) The current input vector $h_t$ was used by three classifiers, namely DLR-E, DLR-U and DBM. (Middle) The classifiers considered augmented inputs, $i_{t,p} = [h_t^T, h_{t-1}^T, \ldots, h_{t-p}^T]^T$. Here $i_{t,(4/6/3)}$ denotes that the value of $p$ for the data sets with 0%, 4% and 22% Bayes errors is 4, 6 and 3 respectively. (Bottom) Another augmented input vector, $d_{t,p} = [h_t^T, h_{t-1}^T, \ldots, h_{t-p}^T, y_{t-1}, \ldots, y_{t-p}]^T$.</td>
<td>114</td>
</tr>
<tr>
<td>6.2</td>
<td>Proportion of correct classification on the mountain fire data. (Top) The current input vector $h_t$ was used by four classifiers. Three adaptive models outperformed a static logistic regression. (Bottom) Adaptive classifiers used an augmented input vector, $d_{t,1} = [h_t^T, h_{t-1}^T, y_{t-1}]^T$.</td>
<td>118</td>
</tr>
<tr>
<td>6.3</td>
<td>Proportion of correct classification on the mountain fire data when a classifier actively requests labels. Despite requesting only half of labels, the classifiers maintained high classification accuracy.</td>
<td>119</td>
</tr>
</tbody>
</table>
6.4 Proportion of correct classification on the EEG data. (Top) The current input vector $h_t$ was used by one static and three dynamic classifiers. The adaptive models outperformed the static model. (Bottom) Adaptive classifiers used an augmented input vector, $d_{t,1} = [h_t^T, h_{t-1}^T, y_{t-1}]^T$. The dynamic classifiers with this augmented input classified labels almost perfectly.

6.5 Proportion of correct classification on the EEG data when a classifier actively requests labels. Despite requesting only half of labels, the classifiers maintained high classification accuracy.
### Notation

Throughout this thesis, the following notations are adopted unless otherwise indicated.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>Scalar random variable</td>
</tr>
<tr>
<td>$x$</td>
<td>Scalar realisation</td>
</tr>
<tr>
<td>$\mathbf{x}$</td>
<td>Vector random variable</td>
</tr>
<tr>
<td>$\mathbf{x}$</td>
<td>Vector realisation</td>
</tr>
<tr>
<td>$X$</td>
<td>Matrix</td>
</tr>
<tr>
<td>$\mathbf{x}^T, X^T$</td>
<td>Transpose of a vector or a matrix</td>
</tr>
<tr>
<td>$p_x(x)$</td>
<td>Probability density (or mass) function of $x$; subscript $x$ is omitted where obvious</td>
</tr>
<tr>
<td>$E(x)$</td>
<td>Expectation of $x$</td>
</tr>
<tr>
<td>$x \sim N(\alpha, \beta)$</td>
<td>Denotes $x$ has a Gaussian distribution with mean $\alpha$ and variance $\beta$</td>
</tr>
<tr>
<td>$x \sim (\alpha, \beta)$</td>
<td>Denotes $x$ has mean $\alpha$ and variance $\beta$ without specifying the probability density of $x$</td>
</tr>
<tr>
<td>$t$</td>
<td>Discrete time index</td>
</tr>
<tr>
<td>$y_t$</td>
<td>Observation variable of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$w_t$</td>
<td>State (or parameter) variable of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$H_t$</td>
<td>Observation matrix of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$F_t$</td>
<td>State evolution matrix of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$n_t$</td>
<td>Observation noise variable of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$v_t$</td>
<td>State noise variable of a dynamic model at time $t$</td>
</tr>
<tr>
<td>$R_t$</td>
<td>Covariance of observation noise process at time $t$</td>
</tr>
<tr>
<td>$Q_t$</td>
<td>Covariance of state noise process at time $t$</td>
</tr>
<tr>
<td>$D_t$</td>
<td>Set of observations up to time $t$, i.e. ${y_1, \ldots, y_t}$</td>
</tr>
<tr>
<td>$K_t$</td>
<td>Kalman gain at time $t$</td>
</tr>
<tr>
<td>$\hat{w}_{t</td>
<td>t}$</td>
</tr>
<tr>
<td>$P_{t</td>
<td>t}$</td>
</tr>
<tr>
<td>$\hat{w}_{t</td>
<td>t-1}$</td>
</tr>
<tr>
<td>$P_{t</td>
<td>t-1}$</td>
</tr>
<tr>
<td>$\hat{y}_{t</td>
<td>t-1}$</td>
</tr>
<tr>
<td>$S_{t</td>
<td>t-1}$</td>
</tr>
</tbody>
</table>
Acknowledgements

First and foremost I would like to thank my supervisor, Professor Stephen Roberts, for his support and guidance, without which all the achievements I have made during the past three years would have been impossible. For help in many different forms, I would like to thank all my friends; to name a few, Mike Osborne, Fr. Yoon, Fr. Kim, Fr. Kang and Joo-Young Park. Last, but by no means the least I would like to thank my mother and brother for keeping faith in me.

This work was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership.
Chapter 1

Introduction

1.1 Motivation

A powerful earthquake, which measured at 7.9 on the Richter scale, hit Sichuan province of China at 14:28 local time on 12th May 2008. As of 2nd July 2008, official figures state that 69,195 are confirmed dead and 374,176 injured. In addition, 18,403 are listed as missing and millions of people are left homeless. The economic loss from the earthquake is substantial; some experts estimate it to be over US$75 billion, thereby it potentially reduces the country’s economic growth for the rest of the year by 0.15 to 0.2 percent (Xinhua News Agency, 2008; The Epoch Times, 2008).

On 29th August 2005, Hurricane Katrina made landfall in southeast Louisiana. Its 125 mph wind makes it the third most intense United States land-falling hurricane on record. Eighty percent of the city of New Orleans was flooded, which occurred following several levée breaches. The effect of the hurricane is reported over an area of 90,000 square miles, nearly the size of the UK. An estimated 1,353 people died, 275,000 homes were damaged or destroyed, and over half a million people were left homeless. The total economic losses are estimated to be greater than US$100 billion. The disaster is regarded as one of the worst to hit the United States since the Great Depression of 1930s (Ravilious, 2006; Johnson, 2006).

A Richter magnitude 9.0 earthquake struck off the West coast of the Indonesian Island of Sumatra on 26th December 2004, which is the largest earthquake to occur in the world in 40 years. The earthquake generated a mega tsunami that devastated the coast of Indonesia, Sri
1.1 Motivation

Lanka, India, and eastern Africa with waves up to 48.9m high. The catastrophic damage and loss of life inflicted by this tsunami and the resulting flooding were staggering with estimated direct fatalities upward of 265,000, tens of thousands reported missing and over a million left homeless (Kawata et al., 2005).

We face these kinds of large-scale natural disasters with increasing frequency, arguably due to global climate change. Not only natural catastrophes devastate our communities, but also civil disorders and terror attacks are nowadays major threats to our peaceful daily lives. As a result, coping with them has become increasingly important. There is abundant information about such disastrous events and we have technology, facilities, governmental and non-governmental units designed to manage these disasters. A question that arises regarding this information, technology and any associated skills, knowledge and resources is how to collect information selectively that is relevant to the disasters, organise all available resources efficiently and make appropriate and timely decisions.

An information network system allows us to deal with disaster situations in a systematic way. A distributed and decentralised intelligent system is illustrated in Figure 1.1. The system consists of a number of independent agents. Each agent is capable of processing information, reasoning and taking actions. It is also able to communicate with other neighbouring agents. Note that there exists no central agent where information fusion or decisions are globally made. These are main characteristics of a distributed and decentralised intelligent multi-agent system.

As a real-world application of such an information network system, we consider the monitoring of a hurricane in the United States. First, a weather forecast centre collects information about the hurricane from different sources. These include hurricane formation data from weather satellites, some information gathered by ‘hurricane hunters’, atmospheric pressure and wind speed data monitored by specially reinforced US Air Force aeroplanes and scientists’ predictions using mathematical models before a hurricane season starts. On the basis of such information, the weather centre makes predictions regarding the hurricane’s possible directions and its potential impacts, and alerts relevant organisations that include
1.1 Motivation

Figure 1.1: The structure of a distributed and decentralised information network system.

Federal emergency management agencies, local authorities, officers and media. Given the predictions and warnings, they communicate and take appropriate actions. While playing its own role, each organisation shares newly gained information and collaborates with others so as to make effective use of limited resources; for example, it prioritises where those resources are needed most (Ravilious, 2006).

As stated above, an individual agent possesses the capability to fuse information, analyse risks, forecast outcomes and make decisions. It does so in an attempt to achieve its own goals. There are, however, a large number of agents that are involved in an information network system. They chase their own goals, which may clash with each other. For instance, suppose that a powerful earthquake hits a densely populated city. Buildings and roads have collapsed, and numerous places are on fire. Fire engines and ambulances are in operation all over the city trying to save civilians. The major goal of a fire engine or an ambulance would be to reach a place in need of help as quickly as possible. When two fire engines
are available near to a building on fire, what could happen is that they both go to the site so as to achieve their own individual goal of saving lives. However, this would be a waste of resources that could be used elsewhere in need. To avoid such situations it is necessary to structure interactions and coordinate the individual agents in a way that each agent is incentivised to meet a desirable global behaviour. Conclusively, problems associated with the information network system can be categorised as either to do with individual agents or to do with multiple agents.

Solving problems of multiple agents is important because it determines macroscopic behaviour of the overall system. However, this fact does not undermine the importance of the issues concerning individual agents as the whole system operates on the basis of their reasoning, forecasts and decisions. For instance, an agent combines information from different sources and makes decisions such as providing appropriate and timely warnings. The importance of an individual agent lies in the fact that the more timely and precise the information is, the more effective actions other agents that receive this information can take, and accordingly more lives and property can be saved. This thesis primarily concerned with the problems of how to make individual agents reason and act in a coherent, consistent and intelligent manner.

1.2 Problem Formulation

Challenges that individual agents face in reality are manyfold. Firstly, information available tends to be noisy due to random disturbances. Secondly, the information is uncertain in a sense that there are unknown or hidden factors that affect it. Thirdly, it can be provided by multiple sources; it is not independent but correlated. Fourthly, our environments are likely to change over time; the information is dynamic. Lastly, it is often incomplete; some events are only partially observable or can be completely missing.

This thesis aims to develop models and techniques for individual agents that form a distributed and decentralised intelligent system. For the information network system to deal with the characteristics of the environments described above, we focus on the framework
A dynamic linear model consists of two recursive equations. The first equation relates an observation variable $y_t$ to a state variable $w_t$:

$$y_t = H_t w_t + n_t.$$  \hspace{1cm} (1.1)

Here, the index $t$ represents that the variables are associated with time $t$. $H_t$ is an input matrix and is referred to as the observation matrix; $n_t$ is a (zero-mean) random error term having a Gaussian distribution. The second equation is associated with the evolution of the state variable:

$$w_t = F_t w_{t-1} + v_t,$$  \hspace{1cm} (1.2)

where $F_t$ governs the linear evolution of the state variable and is referred to as the transition matrix; $v_t$ is a Gaussian diffusion term.

This dynamic linear model has been widely used, to some degree, due to its mathematical elegance and computational simplicity. Moreover, an important feature of the model is that it allows us to analyse data sequentially as a new data point becomes available; it is useful particularly in time series analysis. However, the Gaussian assumption of the model limits its flexibility. For non-Gaussian data we can consider a dynamic generalised linear model, which retains the mathematical and computational properties of the dynamic linear model. Using these dynamic models we are able to fuse information, forecast outcomes and make inferences in an on-line manner. In addition to data fusion, forecasting and inference problems, we consider here sequential planning and acting based on the information fused and the inferences made. This problem can be formulated in a general dynamic model framework. For example, the state variable may be affected by a deterministic action (or control) input $a_t$ according to

$$w_t = F_t w_{t-1} + G_t a_{t-1} + v_t,$$  \hspace{1cm} (1.3)

where $G_t$ is a (known) matrix and is referred to as a control policy. An optimal action can be found in such a way that the sum of rewards that a system is expected to receive over time is maximised, or equivalently, the sum of costs is minimised.
1.2 Problem Formulation

The environments surrounding the individual agent are often incomplete; for instance, it receives information sparsely, or the information is only partially observable. The agents are required nevertheless to reason, forecast and act. The constraints of such circumstances provide difficult challenges to solve. We are primarily interested in studying and developing models and techniques that allow the individual agent to deal with such incomplete environments. Conclusively, the main theme of this thesis will be the sequential forecasting and decision making in uncertain and dynamic environments.

1.2.1 ALADDIN Project

The ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project is a multi-disciplinary research project funded by a BAE Systems and EPSRC strategic partnership. It involves a number of research groups from Imperial College London, University of Southampton, University of Bristol and University of Oxford. The project aims to develop mechanisms, architectures and techniques to deal with the dynamic and uncertain nature of distributed and decentralised intelligent systems, and take informed actions. Disaster management is the chosen application domain.

The project incorporates four separate themes: ‘Individual Actors’, ‘Multiple Actors’, ‘DDIS (Decentralised Data and Information Systems) Architectures’ and ‘Applications’; the first of these is the primary focus of the Oxford side. Key issues involved in this theme include how to fuse noisy, uncertain, incomplete and dynamic information in limited bandwidth systems, how to make efficient use of available information, and how to make optimal decisions or inferences. Of particular interest is to develop sequential methods because disaster management systems are required to run in real time. As mentioned earlier, in this thesis we consider the problem of the sequential forecasting and decision making in uncertain and dynamic environments. Hence, techniques presented here are designed to fit well into the individual agent’s tasks of the ALADDIN project.
1.3 Thesis Overview

The following chapters examine sequential models and techniques for forecasting and making decisions in noisy, uncertain, correlated, dynamic and incomplete environments.

Chapters 2 and 3 establish the mathematical background of dynamic models, on which other models and techniques to be dealt with in later chapters are based. Chapter 2 covers the dynamic linear model; the theoretical background of the model is first discussed, and then we demonstrate an optimal recursive linear filter known as the Kalman filter. Non-linear extensions of the Kalman filter are also covered; we propose using the Chebyshev approximation as a non-linear filter. The subject of Chapter 3 is a dynamic generalised linear model, in which we show that not only the Gaussian distribution but also other distributions in the exponential family can be dealt with within the dynamic model framework.

Chapters 4, 5 and 6 deal with how the dynamic models are employed to forecast and make decisions in practical applications. Sequential decision making methods receive special attention in Chapter 4. Reinforcement learning, a collection of model-free decision making techniques, is a typical example. This has attracted wide attention in recent years. We propose an efficient algorithm using non-linear basis function models, which handles continuous spaces; and we propose a modification of the traditional reinforcement learning technique, which is suitable for non-stationary settings. Chapter 5 covers the multivariate forecasting problem, in which a number of sensors are correlated and their values are only partially observable or missing. We discuss three different models and their performances are compared after applying them to real-world weather condition data. Chapter 6 deals with the sequential dynamic classification problem, in which we classify binary labels in real time. We study dynamic logistic regression models using non-linear Kalman and unscented filters; and we propose using a dynamic binominal model as an alternative dynamic classifier. It is common that labels are sparsely observed and/or inputs are not available. We discuss a method that allows a classification model to actively request labels. In addition, we show that the forecasting models detailed in Chapter 5 can be successfully combined with the dynamic classifiers. Chapter 7 concludes the thesis with discussions regarding future research.
1.3 Thesis Overview

directions.
Chapter 2

Dynamic Linear Models

In this chapter we discuss dynamic linear models, on which more general dynamic models are founded. The Kalman filter (Kalman, 1960; Kalman and Bucy, 1961) – the term ‘filter’ means a causal data processing algorithm – provides a recursive estimation method and has been widely used since its invention in the early sixties. West and Harrison (1997) argue that the Kalman filter is designed to solve a particular form of the dynamic linear model, in which all components of the model are completely known, and go on to insist that “Bayesian forecasting is not Kalman filtering.” Although we agree with their argument, we give special attention to the Kalman filter in this chapter, not only because it is the first attempt to solve a dynamic linear model, but also because it is important to understand the original interest and motivation underlying the filter. We first present the theoretical background of the Kalman filter, namely probability and estimation theory. Then, we discuss the extension of the Kalman filter to dynamic non-linear systems. In addition, we propose a novel non-linear Kalman filter using Chebyshev polynomials.

2.1 Theoretical Background

2.1.1 Review of Probability Theory

Jazwinski (1970) gives an excellent introduction to probability theory. He summarises a “problem solving” procedure in three distinct steps:

1. determination of (prior) probabilities of certain events,
2.1 Theoretical Background

2. operation on these probabilities with certain rules to obtain probabilities of other events,

3. making physical predictions on the basis of the probabilities obtained in Step 2.

Step 1 involves a debate regarding how to assign probabilities; two main perspectives on this matter are known as the ‘relative frequency’ approach and the ‘Bayesian’ approach. For this interesting topic, refer to Berkson (1977); Bernardo and Smith (1994). Statistics is a study of dealing with Step 3; the field of statistics considers inferring something about the probability of an event. The probability theory of our interest deals only with Step 2. We assume that readers are familiar with basic concepts of probability including the probability axioms (Kolmogorov, 1956), therefore we only cover the aspects of the probability directly related to methods presented in subsequent chapters.

A random variable is a real-valued function whose values depend on the outcome of an event. Sums, products and functions of random variables are also random variables. The probability that a random variable \( x \) is less than or equal to its realisation \( x \) can be expressed as

\[
\Pr(x \leq x) \triangleq F_x(x).
\]  

(2.1)

\( F_x(x) \) is called the (probability) distribution function of the random variable \( x \). Note that \( F_x(-\infty) = 0 \) and \( F_x(+\infty) = 1 \). If \( F_x(x) \) is differentiable with respect to \( x \), then \( f_x(x) = \frac{d}{dx}F_x(x) \) is called the (probability) density function of \( x \). Its counterpart for a discrete random variable is called the (probability) mass function. It is important to note that the distribution function completely describes the properties of a random variable.

The expectation – mean, average and first moment are commonly used synonyms – of a random variable \( x \) is defined by

\[
\E(x) \triangleq \int xp_x(x)dx.
\]  

(2.2)

In addition,

\[
\E(x^n) \triangleq \int x^np_x(x)dx
\]  

(2.3)
2.1 Theoretical Background

defines the \( n \)-th moment of \( x \). An important instance of this is the second moment \( E(x^2) \), which is called the \textit{mean square} value of \( x \). The \( n \)-th central moment of \( x \) is defined by

\[
E[(x - E(x))^n] \triangleq \int (x - E(x))^n p_x(x) dx. \tag{2.4}
\]

The second central moment is known as the \textit{variance} of \( x \), which is denoted by \( \text{Var}(x) \); the \textit{standard deviation} of \( x \) is defined by the square root of the variance.

We introduce some concepts that arise in multivariate cases – the term ‘variate’ means a random variable and was first used by Karl Pearson in 1909 (Upton and Cook, 2002). Random variables \( x_1, x_2, \ldots, x_n \) are said to be \textit{jointly distributed} if they are defined on the same probability space. As a single random variable is characterised by its distribution function, they are characterised by their \textit{joint distribution function},

\[
F_{x_1,\ldots,x_n}(x_1,\ldots,x_n) \triangleq \Pr(x_1 \leq x_1, \ldots, x_n \leq x_n), \tag{2.5}
\]

or by their \textit{joint density function},

\[
p_{x_1,\ldots,x_n}(x_1,\ldots,x_n) = \frac{\partial^n}{\partial x_1 \cdots \partial x_n} F_{x_1,\ldots,x_n}(x_1,\ldots,x_n). \tag{2.6}
\]

The \textit{marginal density function} of \( x_1,\ldots,x_m \) \((m < n)\) is computed from the joint density function:

\[
p_{x_1,\ldots,x_m}(x_1,\ldots,x_m) = \int \cdots \int p_{x_1,\ldots,x_m,x_{m+1},\ldots,x_n}(x_1,\ldots,x_m,x_{m+1},\ldots,x_n) dx_{m+1} \cdots dx_n. \tag{2.7}
\]

As with the single variable case, we can define joint moments and joint central moments of two random variables, say \( x_k \) and \( x_l \), which have the forms \( E(x_k^\alpha x_l^\beta) \) and \( E((x_k - E(x_k))^\alpha(x_l - E(x_l))^\beta) \) respectively. Of particular importance is the second joint central moment \((i.e. \alpha = \beta = 1)\), which is called the \textit{covariance} of \( x_k \) and \( x_l \); it is defined by

\[
\text{Cov}(x_k, x_l) \triangleq E[(x_k - E[x_k])(x_l - E[x_l])]. \tag{2.8}
\]

Closely related to the covariance is the \textit{correlation coefficient}, which is defined by

\[
\rho(x_k, x_l) \triangleq \frac{\text{Cov}(x_k, x_l)}{\sqrt{\text{Var}(x_k)} \sqrt{\text{Var}(x_l)}}. \tag{2.9}
\]
We now discuss two very important concepts in probability theory, namely independence and uncorrelatedness. We focus on the distinction between them, which sometimes causes some confusion. Two jointly distributed random variables, say $x_1$ and $x_2$, are said to be \textit{independent} if the following condition is satisfied:

$$
px_{1,2}(x_1, x_2) = px_1(x_1)px_2(x_2).
$$

On the other hand, the two jointly distributed random variables, $x_1$ and $x_2$, are said to be \textit{uncorrelated} if the covariance of the variables is zero;

$$
\text{Cov}(x_1, x_2) = 0.
$$

An important property about the relationship between the independence and the uncorrelatedness is that if two jointly distributed random variables are independent, then they are also uncorrelated. However, the converse of the property is in general not true. In other words, independence is a stronger condition than uncorrelatedness. If two random variables have a Gaussian distribution, then no correlation implies independence (for proof see e.g. page 84 of Rice (1995)). Since random variables are assumed to be Gaussian in many cases, the terms independence and uncorrelatedness are often interchangeably used. Orthogonality is a similar concept; two random variables $y_1$ and $y_2$ are said to be \textit{orthogonal} if

$$
\mathbb{E}(y_1y_2) = 0.
$$

An uncorrelated random variables can be changed into orthogonal random variables. In other words, if $x_1$ and $x_2$ are uncorrelated, then new random variables, $y_1 = x_1 - \mathbb{E}(x_1)$ and $y_2 = x_2 - \mathbb{E}(x_2)$, are orthogonal.

The \textit{conditional density function} $p_{x|z}(x|z)$ of $x$ given $z = z$ is defined by

$$
p_{x|z}(x|z) \triangleq \frac{p_{x,z}(x,z)}{p_z(z)} = \frac{p_{x,z}(x,z)}{\int p_{x,z}(x,z) \, dx}.
$$

Manipulating Equation (2.13) we have the following equation:

$$
p_{x|z}(x|z) = \frac{p_{x|z}(z|x)p_x(x)}{p_z(z)}.
$$

(2.14)
This is famously known as Bayes’ theorem. Since the conditional random variable $x$ given $z$ is itself a random variable, it is possible to consider its expectation. The conditional expectation of $x$ given $z$ is defined by

$$E_x(x|z) \triangleq \int x p_{x|z}(x|z)dx.$$  

(2.15)

Note that $E_x(x|z)$ is a random variable that is a function of $z$, and so we can take the outer expectation with respect to the distribution of $z$:

$$E(x) = E_z[E_x(x|z)].$$  

(2.16)

In other words, the expectation with respect to $z$ of the random variable $E_x(x|z)$ is $E(x)$; this indicates that, roughly speaking, computing the average of $x$ is equivalent to computing its conditional expectation for each given $z$ and then averaging over $z$’s. We can also consider the variance of the random variable $E_x(x|z)$:

$$\text{Var}(x) = \text{Var}_z[E_x(x|z)] + E_z[\text{Var}_x(x|z)].$$  

(2.17)

For proof, see e.g. pages 137–138 of Rice (1995).

**Gaussian Random Variables**

Amongst many widely-used distribution functions such as Poisson, binomial, gamma, etc, we give the Gaussian distribution special attention because of its popularity and importance. Originally de Moivre developed the distribution as an approximation to the binomial distribution, and it was subsequently used by Laplace and Gauss to study measurement error in the analysis of astronomical data (Havil, 2003).

There are several reasons that the Gaussian distribution enjoys such huge popularity. Firstly, from the engineering perspective, many physical processes are approximately Gaussian. This is associated with the central limit theorem.

**Theorem 1 (Central Limit Theorem)** Let $x_1, x_2, \ldots, x_n$ be independent random variables each having the same distribution with mean $\mu$ and variance $\sigma^2$. For a large value of $n$, the distribution of the sample mean is approximately Gaussian with mean $\mu$ and variance $\frac{1}{n}\sigma^2$. 


2.1 Theoretical Background

For the proof of the theorem, the reader is referred to page 169 of Rice (1995). When a large number of small independent random effects are superimposed, regardless of their individual distribution, the distribution of the sum of these effects is approximately Gaussian. As a result, a Gaussian component is often a good model for noise in physical devices (Jazwinski, 1970). If a random variable $x$ is Gaussian, the density function of $y = g(x)$ for a given linear function $g$ is also Gaussian. This is the second reason that the Gaussian distribution is widely used. Lastly, the mathematical properties of the Gaussian distribution make it easy and efficient to work with; a Gaussian assumption allows us to deal with many problems in an analytical way.

2.1.2 Review of Estimation Theory

We start with presenting a short history of estimation theory, focusing on least squares estimation. For an overview of its extensive history refer to Harter (1974). Least squares estimation is one of two main estimation methods, the other being maximum likelihood estimation, which has been widely used since its invention by R. A. Fisher in the early 20th century. Although Gauss anticipated the maximum likelihood method, he rejected it in favour of the idea of the least squares method (Sorenson, 1970).

Estimation theory was originally established in 1806 with the first publication on least squares estimation by Legendre. Despite Legendre’s first publication, the origin of the least squares method is accredited to a then 18 year old named Carl Friedrich Gauss, who claimed to have used the idea since 1795. The debate over the priority of the idea between Gauss and Legendre is an enjoyable read. For interested readers, refer to Plackett (1972); Stigler (1981).

What follows in this subsection is largely based on Sorenson (1970). Estimation theory was developed by astronomers studying the motion of planets and comets. They collected observation data and attempted to correct errors so as to get the best possible information from the observations. Gauss reasoned that the astronomical observations are nothing more than approximations to the truth, and thus the aim of all computation associated with the determination of unknown quantities must be to approximate true values as closely as practicable.
2.1 Theoretical Background

He suggested that “the most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum.” (Gauss, 1963)

To avoid any confusion, we give the definitions of the terms ‘estimation’, ‘estimator’ and ‘estimate’ (Upton and Cook, 2002).

- **Estimation**: the process of using a mathematical formula to calculate from observed data a value for an unknown parameter
- **Estimator**: a statistic, which is a function of the set of random variables corresponding to a set of observations, used to estimate a parameter
- **Estimate**: the realised value of an estimator for a particular sample of data

Suppose that a parameter vector $\mathbf{w}$ is related to an observation vector $\mathbf{y}$ according to

$$y_t = H_t \mathbf{w} + n_t \quad \text{for } t = 1, 2, \ldots, n, \quad (2.18)$$

where $H_t$ represents a matrix of given inputs and $n_t$ a variable for measurement errors. We denote by $\hat{\mathbf{w}}$ the estimate of $\mathbf{w}$ computed with a set of $n$ observations $\{y_1, \ldots, y_n\}$. The residual, which is the difference between an observation and an estimate, is defined by

$$r_t \triangleq y_t - H_t \hat{\mathbf{w}}. \quad (2.19)$$

The “most probable value” of $\mathbf{w}$ mentioned by Gauss, which is known as the least squares (LS) estimate, is the value that minimises the sum of the squares of the residuals:

$$\hat{\mathbf{w}}(\text{LS}) = \operatorname*{arg\,min}_{\mathbf{w}} \sum_{t=1}^{n} r_t^T S_t r_t, \quad (2.20)$$

where $S_t$ represents what Gauss describes as “numbers that measure the degree of precision.”

There is another important estimation method known as the minimum mean square error estimation. It was independently developed by Kolmogorov in 1941 and Wiener in 1942 (Levinson, 1947). As we will see shortly, the problem treated by Kolmogorov and Wiener
The Kalman Filter

is significantly different from Gauss’ problem in which a parameter vector is regarded as a vector of unknown constant values. Meanwhile, Kolmogorov and Wiener discuss the linear least squares estimation for stochastic processes.

To assist understanding we consider the following observation and parameter (or state) processes. The observation process is described by

\[ y_t = H_t w_t + n_t \quad \text{for } t = 1, 2, \ldots, n, \quad (2.21) \]

where \( n_t \) is a white-noise sequence. The state vector \( w \) evolves according to

\[ w_t = F_t w_{t-1} + v_t, \quad (2.22) \]

where \( F_t \) is known dynamics of the state evolution and \( v_t \) is a random noise term. The Wiener-Kolmogorov’s problem is different from that of Gauss in that it considers the stochastic evolution of the state vector. As its name suggests, the minimum mean square error (MMSE) estimate is the estimate of \( w_t \) that minimises the mean-square error:

\[ \hat{w}_t(\text{MMSE}) = \arg\min_{\hat{w}_t} E \left[ (w_t - \hat{w}_t)^T (w_t - \hat{w}_t) \right]. \quad (2.23) \]

A significant development in the estimation theory is a recursive approach, in which we compute the minimum mean-square estimate \( \hat{w}_t \) when a new observation \( y_t \) is available. In doing so we can gain significant economies in terms of processing computation and data storage capacity. The recursive method was designed by different researchers under the terms ‘stepwise’, ‘stagewise’, ‘sequential’, ‘progressive’ or ‘on-line’. The minimum mean square error estimation and the recursive approach are important because they form the basis of the Kalman filter, which is the subject of the following section.

2.2 The Kalman Filter

2.2.1 Background

The main aim of the estimation theory is to search for an optimal (or best) estimator; optimality is in general considered synonymous with the prediction precision. In fact, the prediction
2.2 The Kalman Filter

Precision is not the only factor to consider in the choice of an optimal estimator; other factors involve computational speed, memory capacity required, etc. However, we limit our discussion to the narrow meaning of optimality. Hence, finding an optimal estimator is equivalent to finding an estimator that produces the best prediction accuracy (or the least prediction error).

We denote an estimate error by $e_t$, which is the difference between the true value of a parameter $w_t$ and its estimated value $\hat{w}_t$. Suppose that $\delta(e_t)$ is a non-negative, convex function. For a function $L$ to be a loss function, it must satisfy the following properties:

$$L(e_t) = 0 \quad \text{if} \quad e_t = 0$$

$$L(e_t) \geq L(e_t') \quad \text{if} \quad \delta(e_t) \geq \delta(e_t') \geq 0 \quad \text{(non-decreasing)} \quad (2.24)$$

Here an example of $\delta$ is $\delta(e_t) = ||e_t||_2 = (e_t^T e_t)^{1/2}$. Moreover, an example of the loss function $L$ is $L(e_t) = e_t^T K e_t$, where $K$ is a positive-semidefinite matrix. The risk function $R$ is defined as the expected value of a loss function over the range of admissible values of the error:

$$R(w_t) \triangleq E[L(e_t)] = E[L(w_t - \hat{w}_t)]. \quad (2.25)$$

We aim to find the estimate $\hat{w}_t$ of $w_t$ that minimises the expected loss, i.e. the risk $R(w_t)$.

Assume that we have a conditional density function of $w_t$ at time $t$ given an observation random process $D_t = \{y_1, y_2, \ldots, y_t\}$, which is denoted by $p(w_t|D_t)$. A random process is defined as a time-ordered sequence of random variables. We can rewrite the risk function $R(w_t)$ as follows:

$$R(w_t) = E[E(L(w_t - \hat{w}_t)|D_t)]. \quad (2.26)$$

Note that the first expectation term of this equation does not depend on $\hat{w}_t$, but only on the observation random process $D_t$. As a result, it turns out that minimising $R(w_t)$ with respect to $\hat{w}_t$ is equivalent to minimising $E[L(w_t - \hat{w}_t)|D_t]$.

**Theorem 2** (Sherman, 1958) Assume that a loss function $L$ satisfies the properties listed in Equation (2.24), and the conditional distribution function $p(w_t|D_t)$ is
2.2 The Kalman Filter

1. symmetric about the mean \( \bar{w}_t = E(w_t | D_t) \).

2. unimodal.

Then the optimal estimate \( \hat{w}_t^* \) which minimises the risk function is the conditional expectation; that is,

\[
\hat{w}_t^* = E(w_t | D_t)
\] (2.27)

This theorem is of great importance as it provides conditions required for an optimal estimate. For example, a Gaussian distribution satisfies the requirements of Theorem 2. Hence, if the conditional density, \( p(w_t | D_t) \), is Gaussian, an optimal estimate of \( w_t \) is simply the conditional expectation given in Equation (2.27), regardless of the form of a loss function in use.

Consider that we use the square loss function defined by \( L(e_t) = e_t^T e_t \). In this case, the estimate that minimises the risk function \( R(w_t) = E[L(w - \hat{w}_t)] \) is referred to as the minimum mean square error estimate.

**Theorem 3 (Sherman, 1958) If the square loss function is used, then Theorem 2 is true without the properties of the conditional distribution.**

This theorem tells us that the minimum mean square error is the conditional mean. In other words, if the square loss function is used, then the conditional expectation, \( E(w_t | D_t) \) yields the optimal estimate of \( w_t \), without any assumption on the form of the conditional distribution; the Gaussian assumption is not necessary. This result is the principal reason why the minimum mean square error estimator is almost universally employed for estimation problems (Deutsch, 1965). However, since an optimal estimator generated from Theorem 3 is not guaranteed to be linear in the case of non-Gaussian processes (Sherman, 1958), it is common to assume random processes to be Gaussian.

### 2.2.2 Optimal Recursive Linear Filter

Maybeck (1979) defines the Kalman filter as an optimal recursive linear filter. This definition describes everything associated with the Kalman filter. Firstly, *Optimal* means that the filter
is based on the minimum mean square error estimation, in which an optimal estimator is the conditional expectation given a set of observations. Secondly, *Recursive* indicates that the estimation is performed sequentially each time a new observation becomes available. Lastly, *Linear* means that the estimator is a linear combination of observations.

Consider a dynamic linear model in which a multivariate observation $y$ is related to a state (or parameter) variable $w$ by

$$ y_t = H_t w_t + n_t, \quad (2.28) $$

and the state variable is described by

$$ w_t = F_t w_{t-1} + v_t. \quad (2.29) $$

Here, $n_t$ and $v_t$ represent independent Gaussian noise sequences with zero mean vectors and covariance matrices $R_t$ and $Q_t$ respectively; they are assumed to be uncorrelated. We refer to $H_t$ and $F_t$ as the *observation matrix* and the *evolution matrix* respectively.

Given a set of observations, $D_t = \{y_1, \ldots, y_t\}$, an optimal estimate of the state $w_t$ is determined as that which minimises the mean square error; that is,

$$ \hat{w}^*_t|_t = \arg\min_{\hat{w}_t} \mathbb{E}\left[ (w_t - \hat{w}^*_t|_t)^T (w_t - \hat{w}^*_t|_t) \right]. \quad (2.30) $$

As detailed in Equation (2.26), the above equation is equivalent to

$$ \hat{w}^*_t|_t = \arg\min_{\hat{w}_t} \mathbb{E}\left[ (w_t - \hat{w}^*_t|_t)^T (w_t - \hat{w}^*_t|_t) \bigg| D_t \right]. \quad (2.31) $$

Moreover, according to Theorem 3 the optimal estimate is simply the conditional expectation of $w_t$ given $D_t$:

$$ \hat{w}^*_t|_t = \mathbb{E}(w_t|D_t). \quad (2.32) $$

Note that the conditional mean is an unbiased estimate. The estimated covariance is defined by

$$ P^*_t|_t \triangleq \mathbb{E}\left[ (w_t - \hat{w}^*_t|_t)(w_t - \hat{w}^*_t|_t)^T \bigg| D_t \right]. \quad (2.33) $$

Since

$$ \mathbb{E}(P^*_t|_t) = \mathbb{E}\left[ (w_t - \hat{w}^*_t|_t)(w_t - \hat{w}^*_t|_t)^T \right], \quad (2.34) $$
we have
\[
\text{tr} \left( E(\mathbf{P}_t|t) \right) = E \left[ (\mathbf{w}_t - \hat{\mathbf{w}}^*_t)^T (\mathbf{w}_t - \hat{\mathbf{w}}^*_t) \right],
\]
where ‘tr’ denotes the trace of a matrix. Hence, \( \mathbf{P}_{t|t} \) provides a measure of the goodness of the estimate; the knowledge of \( \mathbf{P}_{t|t} \) is as important as that of \( \hat{\mathbf{w}}^*_t \) (Jazwinski, 1970). From now on the superscript * is deleted from the expression of the optimal estimate for brevity of notation.

We define an optimal estimate of \( \mathbf{w}_t \) given observations up to \( t - 1 \), i.e. \( \mathbf{D}_{t-1} \), by
\[
\hat{\mathbf{w}}_{t|t-1} \triangleq E(\mathbf{w}_t|\mathbf{D}_{t-1}).
\]
We refer to it as a prior estimate of \( \mathbf{w}_t \) and it can be rewritten using the state process in Equation (2.29):
\[
\hat{\mathbf{w}}_{t|t-1} = E(F_t \mathbf{w}_{t-1} + \mathbf{v}_t|\mathbf{D}_{t-1})
\]
\[
= F_t E(\mathbf{w}_{t-1}|\mathbf{D}_{t-1}) + E[\mathbf{v}_t|\mathbf{D}_{t-1}]
\]
\[
= F_t \hat{\mathbf{w}}_{t-1|t-1}.
\]
(2.37)

A prior estimate of the covariance, denoted by \( \mathbf{P}_{t|t-1} \), is
\[
\mathbf{P}_{t|t-1} \triangleq \text{Cov}(\mathbf{w}_t|\mathbf{D}_{t-1})
\]
\[
= E \left[ (\mathbf{w}_t - \hat{\mathbf{w}}_{t|t-1})(\mathbf{w}_t - \hat{\mathbf{w}}_{t|t-1})^T \right]
\]
\[
= F_t \mathbf{P}_{t-1|t-1} F_t^T + \mathbf{Q}_t.
\]
(2.40)

An optimal estimate \( \hat{\mathbf{w}}_{t|t} \) can be computed as a linear combination of the prior estimate \( \hat{\mathbf{w}}_{t|t-1} \) and a new observation \( \mathbf{y}_t \) as follows:
\[
\hat{\mathbf{w}}_{t|t} = \mathbf{K}_t' \hat{\mathbf{w}}_{t|t-1} + \mathbf{K}_t \mathbf{y}_t,
\]
(2.41)
where \( \mathbf{K}_t' \) and \( \mathbf{K}_t \) are weighting (or gain) matrices. Using the unbiased condition of \( \hat{\mathbf{w}}_{t|t} \) and the observation process in Equation (2.28) we can find that \( \mathbf{K}_t' = \mathbf{I} - \mathbf{K}_t \mathbf{H}_t \), thereby Equation (2.41) is rewritten as
\[
\hat{\mathbf{w}}_{t|t} = \hat{\mathbf{w}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{w}}_{t|t-1}).
\]
(2.42)
Note that $H_t \hat{w}_{t|t-1}$ is an expected measurement of $y_t$, which is denoted by $\hat{y}_{t|t-1}$. A residual of the expected measurement (i.e. $y_t - H_t \hat{w}_{t|t-1}$) reflects an error in $\hat{w}_{t|t-1}$. In addition, we can write the covariance in $\hat{w}_{t|t}$ as follows:

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} (I - K_t H_t)^T + K_t R_t K_t^T. \tag{2.43}$$

To complete a recursive linear algorithm for an optimal estimate of $w_t$, we need to determine the gain matrix $K_t$. Since $\hat{w}_{t|t}$ is an optimal estimate, $K_t$ should be determined in a way to minimise the mean square error of the estimate; that is, to minimise the following risk function:

$$R(w_t) = E[(w_t - \hat{w}_{t|t})^T (w_t - \hat{w}_{t|t})]$$

$$= \text{tr} \left( E[(w_t - \hat{w}_{t|t})(w_t - \hat{w}_{t|t})^T] \right)$$

$$= \text{tr}(P_{t|t}). \tag{2.44}$$

Differentiating this equation with respect to $K_t$ and setting it equal to zero gives

$$K_t = P_{t|t-1} H_t^T [H_t P_{t|t-1} H_t^T + R_t]^{-1}. \tag{2.45}$$

It is generally referred to as the Kalman gain. For more details of this derivation, refer to Durrant-Whyte (2001).

Summarising the discussion above, the Kalman filter is commonly represented as two distinct steps, namely prediction and update steps:

**Prediction Step**

$$\hat{w}_{t|t-1} = F_t \hat{w}_{t-1|t-1} \tag{2.46}$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \tag{2.47}$$

**Update Step**

$$K_t = P_{t|t-1} H_t^T [H_t P_{t|t-1} H_t^T + R_t]^{-1} \tag{2.48}$$

$$\hat{w}_{t|t} = \hat{w}_{t|t-1} + K_t (y_t - H_t \hat{w}_{t|t-1}) \tag{2.49}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{2.50}$$
2.3 Smoothing and Forecasting

Although Kalman originally reached a recursive solution for a dynamic linear model using the idea of orthogonal projections (Kalman, 1960), his approach is essentially equivalent to the sequential estimation procedure detailed above. There are other perspectives and approaches to the Kalman filter: Ho and Lee (1964) made the first attempt to treat the Kalman filter in a Bayesian perspective; Kailath (1968) paid special attention to a residual, which was named an innovation, as the residual can be thought of as new information brought by a new observation.

2.3 Smoothing and Forecasting

There are, roughly speaking, three inference and prediction processes in the dynamic linear model: filtering, smoothing and forecasting. An optimal recursive linear filter, the Kalman filter, has been so far covered. The other two processes, smoothing and forecasting, are the subjects of this section. The following probability densities define concisely what we discuss shortly.

- State Filtering: \( p(w_t|D_t) \)
- State Smoothing: \( p(w_{t-k}|D_t) \) for \( k \geq 1 \)
- Observation Forecasting: \( p(y_{t+k}|D_t) \) for \( k \geq 1 \)

2.3.1 The Kalman-Rauch Smoother

Filtering is concerned with the inference of a state at time \( t \) given all observations up to time \( t \). Meanwhile, smoothing considers the picture of what happened in the past. In other words, smoothing infers past states given all observations up to the present.

Rauch et al. (1965) proposed a retrospective estimation technique known as the Kalman-Rauch smoother. It consists of two steps: one is a forward step and the other a backward step. Suppose that we have a set of observations up to time \( t = T \), i.e. \( D_T = \{y_1, \ldots, y_T\} \). In the forward step a process runs forward in time from \( t = 1 \) to \( t = T \). In other words, we compute
2.3 Smoothing and Forecasting

the posterior mean and covariance of the state variable, \( \hat{w}_{t|t} \) and \( P_{t|t} \) (for \( t = 1, 2, \ldots, T \)), via the filtering algorithm detailed in the previous subsection.

In the backward step we infer the past states (for \( t = T, \ldots, 1 \)) on the basis of the forward filtered estimates of the states. The backward equations are given by

\[
\hat{w}_{t-1|T} = \hat{w}_{t-1|t-1} + J_{t-1} (\hat{w}_{t|T} - \hat{w}_{t|t-1}),
\]

\[
P_{t-1|T} = P_{t-1|t-1} + J_{t-1} (P_{t|T} - P_{t|t-1}) J_{t-1}^T,
\]

where

\[
J_{t-1} = P_{t-1|t-1} F_t^T (P_{t|t-1})^{-1}.
\]

If we recursively run the backward equations, the smoothed estimates of the states, \( \hat{w}_{t|T} \) and \( P_{t|T} \), can be computed from \( t = T - 1 \) to \( t = 1 \).

Additionally, the covariance \( P_{t,t-1|T} \) between two states adjacent in time given \( D_T \) can be computed using the backward equations. For \( t = T, T - 1, \ldots, 2 \),

\[
P_{t-1,t-2|T} = P_{t-1|t-1} J_{t-2} + J_{t-1} (P_{t,t-1|T} - F_t P_{t-1|t-1}) J_{t-2}^T,
\]

where

\[
P_{T,T-1|T} = (I - K_T H_T^T) F_T P_{T-1|T-1}.
\]

2.3.2 Observation Forecasting

In the state filtering process we compute the one-step ahead forecast distribution of the observation variable, \( p(y_t|D_{t-1}) \), which is represented by two moments of the distribution:

\[
\hat{y}_{t|t-1} = E(y_t|D_{t-1}) = H_t \hat{w}_{t|t-1},
\]

\[
S_{t|t-1} = \text{Cov}(y_t|D_{t-1}) = H_t P_{t|t-1} H_t^T + R_t.
\]

We are often interested in forecasting not only a one-step ahead observation but also a multiple-step ahead observation with observations up to time \( t \); we would like to obtain

\[
p(y_{t+k}|D_t), \quad \text{for} \quad k = 1, 2, \ldots.
\]
According to the observation process of the dynamic linear model given in Equation (2.28), the observation variable at \( t + k \) can be represented as follows:

\[
y_{t+k} = H_{t+k} w_{t+k} + n_{t+k}. \tag{2.59}
\]

Hence, computing the forecast distribution of Equation (2.58) is closely related to inferring the distribution of future states, \( p(w_{t+k}|D_t) \). The inference of the future state given \( D_t \) can be thought of as running the filtering process recursively without new observations being received. This means that the mean and covariance of \( w_{t+k} \) can be computed from those of the state at \( w_{t+k-1} \) as follows:

\[
\hat{w}_{t+k|t} = F_{t+k} \hat{w}_{t+k-1|t},
\]

\[
P_{t+k|t} = F_{t+k} P_{t+k-1|t} F_{t+k}^T + Q_{t+k}. \tag{2.61}
\]

Using Equations (2.59), (2.60) and (2.61), we can readily compute two moments of the forecast distribution \( p(y_{t+k}|D_t) \):

\[
\hat{y}_{t+k|t} = H_{t+k} \hat{w}_{t+k|t},
\]

\[
S_{t+k|t} = H_{t+k} P_{t+k|t} H_{t+k}^T + R_{t+k}. \tag{2.63}
\]

In the case of a constant evolution matrix, i.e., \( F_t = F \), inferring the distribution of the future state \( w_{t+k} \) is significantly simplified without recursive computation. In other words, it can be computed directly from the filtered distribution at \( t \), i.e. \( p(w_t|D_t) \). The first two moments of the distribution are given by

\[
\hat{w}_{t+k|t} = F^k \hat{w}_{t|t}, \tag{2.64}
\]

\[
P_{t+k|t} = F^k P_{t|t} (F^k)^T + \sum_{i=0}^{k-1} F^i Q_{t+k-i} (F^i)^T. \tag{2.65}
\]

Accordingly, the mean and covariance of the forecast distribution of the observation variable are

\[
\hat{y}_{t+k|t} = H_{t+k} F^k \hat{w}_{t|t}, \tag{2.66}
\]

\[
S_{t+k|t} = H_{t+k} P_{t+k|t} H_{t+k}^T + R_{t+k}. \tag{2.67}
\]
2.4 Non-linear Kalman Filters

We have so far dealt with dynamic linear models in which an observation variable is linearly related to a state variable. However, in many real-world problems such linear models are not suitable and therefore we consider the following non-linear model:

\[
y_t = h_t(w_t) + n_t,
\]
\[
w_t = f_t(w_{t-1}) + v_t,
\]

where \(f_t\) and \(h_t\) are known, non-linear, differentiable vector functions, and \(n_t\) and \(v_t\) are noise variables with zero mean vectors and covariances \(R_t\) and \(Q_t\) respectively. In this section we will discuss three sub-optimal recursive filters designed to handle the non-linear dynamic model.

2.4.1 The Extended Kalman Filter

As with the dynamic linear model, we aim to obtain an optimal estimate of the state variable given a set of observations \(D_t\); that is,

\[
\hat{w}_{t|t} = \arg\min_{\hat{w}_{t|t}} E\left[ (w_t - \hat{w}_{t|t})^T (w_t - \hat{w}_{t|t}) \right].
\]

Unfortunately the completeness and mathematical elegance of the theory developed for the linear case cannot be extended to the non-linear model. In order to emulate the advantages of the linear theory, the most straightforward approach is to linearise the non-linear observation and state evolution functions using the first-order Taylor series approximation.

Suppose that we have the prior mean and covariance of \(w_t\), i.e. \(\hat{w}_{t|t-1}\) and \(P_{t|t-1}\) respectively. The non-linear evolution function is linearised according to the first-order Taylor series expansion, and thus the evolution equation can be approximately rewritten as

\[
w_t = f_t(w_{t-1}) + v_t \approx f_t(\hat{w}_{t-1|t-1}) + F_t(w_{t-1} - \hat{w}_{t-1|t-1}) + v_t,
\]

where \(F_t\) is the matrix derivative of the evolution function evaluated at \(\hat{w}_{t-1|t-1}\); that is,

\[
F_t \triangleq \frac{\partial f_t(w_{t-1})}{\partial w_{t-1}} \bigg|_{w_{t-1}=\hat{w}_{t-1|t-1}}.
\]
Similarly, the non-linear observation equation is approximated by

\[ y_t = h_t(w_t) + n_t \approx h_t(\hat{w}_{t|t-1}) + H_t(w_t - \hat{w}_{t|t-1}) + n_t, \]  

where

\[ H_t \triangleq \frac{\partial h_t(w_t)}{\partial w_t} \bigg|_{w_t = \hat{w}_{t|t-1}}. \]  

Using the linearised evolution equation of Equation (2.71), the approximate prior mean and covariance of \( w_t \) given \( D_{t-1} \) can be computed as follows:

\[
\hat{w}_{t|t-1} = \mathbb{E}(w_t|D_{t-1}) \\
\approx \mathbb{E} \left[ f_t(\hat{w}_{t-1|t-1}) + F_t(w_{t-1} - \hat{w}_{t-1|t-1}) + v_t \big| D_{t-1} \right] \\
\approx f_t(\hat{w}_{t-1|t-1}) + F_t(E(w_{t-1}|D_{t-1}) - \hat{w}_{t-1|t-1}) + E(v_t|D_{t-1}) \\
\approx f_t(\hat{w}_{t-1|t-1}),
\]

and

\[
P_{t|t-1} = \text{Cov}(w_t|D_{t-1}) \\
\approx \text{Cov} \left[ f_t(\hat{w}_{t-1|t-1}) + F_t(w_{t-1} - \hat{w}_{t-1|t-1}) + v_t \big| D_{t-1} \right] \\
\approx \text{Cov}(F_t w_{t-1}|D_{t-1}) + \text{Cov}(v_t|D_{t-1}) \\
\approx F_t P_{t-1|t-1} F_t^T + Q_t,
\]

After observing a new observation \( y_t \), the prior mean and covariance are updated according to following equations:

\[
\hat{w}_{t|t} \approx \hat{w}_{t|t-1} + K_t (y_t - H_t \hat{w}_{t|t-1}), \tag{2.77}
\]

\[
P_{t|t} \approx (I - K_t H_t) P_{t|t-1}, \tag{2.78}
\]

where

\[
K_t \approx P_{t|t-1} H_t^T \left[ H_t P_{t|t-1} H_t^T + R_t \right]^{-1}. \tag{2.79}
\]

They are equivalent to those of the linear Kalman filter except that the observation matrix is replaced with the matrix derivative of the observation function. This recursive algorithm is known as the extended Kalman filter (e.g. (Jazwinski, 1970)).
There is a phenomenon associated with the extended Kalman filter, which is referred to as divergence. The divergence occurs when the prior estimate covariance $P_{t|t-1}$ becomes undetectably small in comparison to an actual estimate error; the small $P_{t|t-1}$ leads to a small Kalman gain, thereby a new measurement $y_t$ is given little weight. This implies that the new observation does not have much contribution to the updated estimates, which makes the recursive estimation procedure meaningless. The divergence is a direct consequence of approximation errors introduced by the linear approximation because $P_{t|t-1}$ is just an approximation (Sorenson, 1966). In order to avoid the divergence problem we could think of approximating the non-linear functions more accurately via, for example, a higher-order Taylor series approximation. However, any possible improvements achieved by the idea cost much computation. For this reason any non-linear filters considering beyond the first-order Taylor series approximation are rarely used in practical use.

### 2.4.2 The Unscented Kalman Filter

Julier and Uhlmann (1997, 2004) proposed a filter named the unscented Kalman filter. It is based on the unscented transformation, which is a method for calculating statistics of a random variable that undergoes a non-linear transformation and is founded on the principle that it is easier to approximate a probability distribution than an arbitrary non-linear function. The unscented transformation resembles Monte Carlo sampling methods in that a number of points are selected and the points are propagated through a non-linear function. However, the unscented transformation selects a set of points, referred to as sigma points, not randomly but deterministically, so that they preserve the statistics (e.g. mean and variance) of their underlying distribution.

Suppose we have two random variables $a$ and $c$, and a non-linear function $b(\cdot)$. We would like to infer the probability density function of $c = b(a)$; we denote the means and covariances of $a$ and $c$ by $\hat{a}$, $\hat{c}$, $P_a$ and $P_c$ respectively. We first select $2L + 1$ ($L$ is the
dimension of \( \alpha \) \textit{sigma} vectors \( \mathcal{A}_i \) in the following deterministic way:

\[
\mathcal{A}_0 = \hat{\alpha} \\
\mathcal{A}_i = \hat{\alpha} + \left( \sqrt{(L + \lambda)P_a} \right)_i, \quad i = 1, \ldots, L \\
\mathcal{A}_i = \hat{\alpha} - \left( \sqrt{(L + \lambda)P_a} \right)_i, \quad i = L + 1, \ldots, 2L
\] (2.80)

where \( \lambda = \alpha^2(L + \kappa) - L \). \( \left( \sqrt{(L + \lambda)P_a} \right)_i \) represents the \( i \)th column of the matrix square root. \( \alpha \) and \( \kappa \) are parameters of the method; the former is a scaling parameter that determines the spread of the sigma points around \( \hat{\alpha} \) and is normally set to a small positive value; and the latter is a secondary scaling parameter and is normally set to 0 (Wan and van der Merwe, 2001).

In order to compute the statistics of the transformed vector \( \mathcal{C} \), the sigma vectors \( \{ \mathcal{A}_i \} \) are propagated through the non-linear function and we have another set of sigma vectors:

\[
\mathcal{C}_i = b(\mathcal{A}_i), \quad i = 0, 1, \ldots, 2L. 
\] (2.81)

The unscented transformation approximates the mean and covariance of \( \mathcal{C} \) as weighted sums of the propagated sigma vectors according to

\[
\hat{\mathcal{C}} \approx \sum_{i=0}^{2L} \tau_{i}^{(m)} \mathcal{C}_i, \\
P_{\mathcal{C}} \approx \sum_{i=0}^{2L} \tau_{i}^{(c)} (\mathcal{C}_i - \bar{\mathcal{C}})(\mathcal{C}_i - \bar{\mathcal{C}})^T 
\] (2.82)

Here weights \( \tau_i \) are given by

\[
\tau_0^{(m)} = \frac{\lambda}{L + \lambda}, \\
\tau_0^{(c)} = \frac{\lambda}{L + \lambda} + 1 - \alpha^2 + \beta, \\
\tau_i^{(m)} = \tau_i^{(c)} = \frac{1}{2(L + \lambda)}, \quad i = 1, \ldots, 2L
\] (2.84)

where \( \beta \) is another parameter that incorporates prior knowledge of the distribution of \( \alpha \). For Gaussian distributions, \( \beta = 2 \) is optimal (Wan and van der Merwe, 2001).
In the unscented Kalman filter, the prior and posterior means and covariances of the state variable of a dynamic non-linear model are approximated by the unscented transformation. The procedure for the unscented Kalman filter is as follows:

1. Select a set of sigma vectors for the posterior distribution of $w_{t-1}$ as follows:

$$\mathcal{W}^0_{t-1|t-1} = \hat{w}_{t-1|t-1}$$

$$\mathcal{W}^i_{t-1|t-1} = \hat{w}_{t-1|t-1} + \left(\sqrt{(W + \lambda)P_{t-1|t-1}}\right)_i, \quad i = 1, \ldots, W \tag{2.85}$$

$$\mathcal{W}^{i-1}_t = \hat{w}_{t-1|t-1} - \left(\sqrt{(W + \lambda)P_{t-1|t-1}}\right)_i, \quad i = W + 1, \ldots, 2W$$

where $W$ is the dimension of the state variable.

2. Propagate the sigma vectors through the evolution function $f_t$ given in Equation (2.69) and compute other set of sigma vectors:

$$\mathcal{W}^i_{t|t-1} = f_t(\mathcal{W}^i_{t-1|t-1}). \tag{2.86}$$

3. Compute the prior mean and covariance of $w_t$ with the new sigma vectors:

$$\hat{w}_{t|t-1} \approx \sum_{i=0}^{2W} \tau_i^{(m)} \mathcal{W}^i_{t|t-1}, \tag{2.87}$$

$$P_{t|t-1} \approx \sum_{i=0}^{2W} \tau_i^{(c)} (\mathcal{W}^i_{t|t-1} - \hat{w}_{t|t-1})(\mathcal{W}^i_{t|t-1} - \hat{w}_{t|t-1})^T + Q_t. \tag{2.88}$$

4. Propagate $\mathcal{W}^i_{t|t-1}$ through the observation function $h_t$ given in Equation (2.68) and compute another set of sigma vectors:

$$\mathcal{Y}^i_{t|t-1} = h_t(\mathcal{W}^i_{t|t-1}). \tag{2.89}$$

5. Compute the mean and covariance of the one-step ahead forecast:

$$\hat{y}_{t|t-1} \approx \sum_{i=0}^{2W} \tau_i^{(m)} \mathcal{Y}^i_{t|t-1}. \tag{2.90}$$

$$S_{t|t-1} \approx \sum_{i=0}^{2W} \tau_i^{(c)} (\mathcal{Y}^i_{t|t-1} - \hat{y}_{t|t-1})(\mathcal{Y}^i_{t|t-1} - \hat{y}_{t|t-1})^T + R_t. \tag{2.91}$$
6. Compute the covariance between the state and observation variables:

\[
P_{w,y} \approx \sum_{i=0}^{2W} \tau_i^{(c)} (\mathcal{Y}_{t|t-1} - \hat{w}_{t|t-1})(\mathcal{Y}_{t|t-1} - \hat{y}_{t|t-1})^T.
\]  
(2.92)

7. Compute the Kalman gain and the posterior mean and covariance of \(w_t\)

\[
K_t = P_{w,y} S_{t|t-1}^{-1}
\]
(2.93)

\[
w_{t|t} = w_{t|t-1} + K_t(y_t - y_{t|t-1})
\]
(2.94)

\[
P_{t|t} = P_{t|t-1} - K_t S_{t|t-1} K_t^T
\]
(2.95)

There are two major advantages of the unscented Kalman filter over the extended Kalman filter. Firstly, whereas the linearisation method of the extended Kalman filter approximates the true mean and covariance up to the first order, the mean and covariance obtained by the unscented Kalman filter for Gaussians are accurate up to the third order for all non-linearities; and for non-Gaussians, accuracy up to at least the second order is guaranteed (Julier and Uhlmann, 1996). Secondly, despite the improvement of the approximation accuracy, the computational cost required by the unscented transformation is equal to the linearisation method. A comparison between the linearisation approach, the unscented transformation and a sampling method is excellently illustrated in Wan and van der Merwe (2001).

### 2.4.3 The Chebyshev Kalman Filter

As Deutsch (1965) pointed out, an alternative technique for the linearisation method is to replace a non-linear function by an appropriate approximating function. Since the Chebyshev approximation is known as the best approximation method to a function in a sense of minimax (van Loan, 1997; Mathews and Fink, 2004), we propose a non-linear filter that exploits the advantage of the approximation method, named the Chebyshev Kalman filter.

We start with a brief review of the Chebyshev polynomials and the Chebyshev approximation (e.g. Press et al. (1992)). The Chebyshev polynomial of degree \(n\) is defined by

\[
T_n(x) = \cos(n \arccos x) \quad \text{for} \quad -1 \leq x \leq 1.
\]
(2.96)
First three Chebyshev polynomials are, for instance, \( T_0(x) = 1 \), \( T_1(x) = x \) and \( T_2(x) = 2x^2 - 1 \). The polynomial \( T_n(x) \) has \( n \) zeros at the following points in the interval \([-1, 1]\):

\[
x_k = \cos \left( \frac{\pi (2k - 1)}{2n} \right) \quad \text{for} \quad k = 1, 2, \ldots, n.
\]

(2.97)

These are referred to as Chebyshev nodes. The Chebyshev polynomials are orthogonal with respect to the weight \( \sqrt{1-x^2} \) in the interval \([-1, 1]\); that is,

\[
\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} \, dx = \begin{cases} 
0 & , \ i \neq j \\
\frac{\pi}{2} & , \ i = j \neq 0 \\
\pi & , \ i = j = 0
\end{cases}.
\]

(2.98)

The Chebyshev polynomials also satisfy a discrete orthogonality relation as shown below:

\[
\sum_{k=1}^{n} T_i(x_k)T_j(x_k) = \begin{cases} 
0 & , \ i \neq j \\
n/2 & , \ i = j \neq 0 \\
n & , \ i = j = 0
\end{cases}.
\]

(2.99)

**Theorem 4 (Chebyshev Approximation)** An arbitrary function \( f(x) \) in the interval \([-1, 1]\) is approximated by a sum of Chebyshev polynomials of degree up to \( n \):

\[
f(x) \approx \sum_{j=0}^{n} c_j T_j(x).
\]

(2.100)

The coefficients \( \{c_j\} \) are defined by

\[
c_0 = \frac{1}{n+1} \sum_{k=1}^{n+1} f(x_k)T_0(x_k) = \frac{1}{n+1} \sum_{k=1}^{n+1} f(x_k),
\]

\[
c_j = \frac{2}{n+1} \sum_{k=1}^{n+1} f(x_k)T_j(x_k)
\]

\[
= \frac{2}{n+1} \sum_{k=1}^{n+1} f(x_k) \cos \left( \frac{j\pi (2k - 1)}{2n} \right) \quad \text{for} \quad j = 1, 2, \ldots, n.
\]

(2.101)

Note that when computing the coefficients, we need the Chebyshev nodes of \( T_{n+1}(x) \).

It is necessary to generalise the Chebyshev approximation by expanding the range of approximation from the interval \([-1, 1]\) to two arbitrary limits, say \([a, b]\). If we change the variable \( a \leq z \leq b \) to \(-1 \leq x \leq 1\) according to

\[
x = -1 + \frac{2}{b-a}(z-a),
\]

(2.102)
2.4 Non-linear Kalman Filters

then we can have Chebyshev nodes on $[a, b]$:

$$z_k = a + \frac{b-a}{2}(x_k + 1)$$

for $j = 1, 2, \ldots, n,$ \hspace{1cm} (2.103)

where $x_k$ are the Chebyshev nodes of $T_n(x)$ on $[-1, 1]$ that are defined in Equation (2.97). A function $f(z)$ over $[a, b]$ is approximated by Chebyshev polynomials in $x$ as follows:

$$f(z) \approx \sum_{j=0}^{n} c_j T_j(x),$$

where the coefficients are defined by

$$c_0 = \frac{1}{n+1} \sum_{k=1}^{n+1} f(z_k),$$

$$c_j = \frac{2}{n+1} \sum_{k=1}^{n+1} f(z_k) T_j(x_k) \quad \text{for} \quad j = 1, 2, \ldots, n.$$ \hspace{1cm} (2.105)

With this generalised Chebyshev approximation we propose a non-linear recursive estimation algorithm. Consider a non-linear dynamic model,

$$y_t = h(w_t) + n_t,$$ \hspace{1cm} (2.106)

$$w_t = f(w_{t-1}) + v_t.$$ \hspace{1cm} (2.107)

Note that $h$ and $f$ are scalar non-linear functions and the observation and state variables are not a vector but a scalar. Noise processes, $n_t$ and $v_t$, are assumed to have zero means and variances $r_t$ and $q_t$ respectively. We have seen that in the extended Kalman filter the non-linear functions $h$ and $f$ are approximated by the first-order Taylor approximation. In the new filter we approximate them with the Chebyshev approximation of degree 1; for example,

$$h(w_t) \approx c_0 T_0(u_t) + c_1 T_1(u_t)$$

$$\approx c_0 + c_1 u_t,$$ \hspace{1cm} (2.108)

where $a \leq w_t \leq b$ and $-1 \leq u_t \leq 1$. If we regard $w_t$ and $u_t$ as $z$ and $x$ respectively in the generalised Chebyshev approximation procedure, then the coefficients, $c_0$ and $c_1$, can be readily computed. The problem of determining the approximation range, i.e. $a$ and $b$, will be discussed later in this section.
Suppose we have the posterior mean and variance of \( w_{t-1} \) denoted by \( \hat{w}_{t-1|t-1} \) and \( P_{t-1|t-1} \). The prior mean and variance of \( w_t \) given \( D_{t-1} = \{y_1, \ldots, y_{t-1}\} \) are approximated by the Chebyshev approximation as shown below:

\[
\begin{align*}
\hat{w}_{t|t-1} &= E(w_t|D_{t-1}) \\
&\approx E(c_0^f + c_1^f u_{t-1} + v_t|D_{t-1}) \\
&\approx c_0^f + c_1^f E(u_{t-1}|D_{t-1}), \
\end{align*}
\]

(2.109)

\[
\begin{align*}
P_{t|t-1} &= \text{Var}(w_t|D_{t-1}) \\
&\approx \text{Var}(c_0^f + c_1^f u_{t-1} + v_t|D_{t-1}) \\
&\approx (c_1^f)^2 \text{Var}(u_{t-1}|D_{t-1}) + q_t, \
\end{align*}
\]

(2.110)

where the superscript \( f \) of the coefficients indicates that they are for the evolution function.

Here, since \( u_t \) is related to \( w_t \) according to

\[
u_t = -1 + \frac{2}{b-a}(w_t - a), \quad E[u_{t-1}|D_{t-1}] \text{ and Var}[u_{t-1}|D_{t-1}] \]

in Equations (2.109) and (2.110) can be rewritten as

\[
\begin{align*}
E[u_{t-1}|D_{t-1}] &= E\left[-1 + \frac{2}{b-a}(w_{t-1}-a) \bigg| D_{t-1}\right] \\
&= -1 + \frac{2}{b-a} \left(\hat{w}_{t-1|t-1} - a\right), \
\end{align*}
\]

(2.111)

\[
\begin{align*}
\text{Var}[u_{t-1}|D_{t-1}] &= \text{Var}\left[-1 + \frac{2}{b-a}(w_{t-1}-a) \bigg| D_{t-1}\right] \\
&= \left(\frac{2}{b-a}\right)^2 P_{t-1|t-1}. \
\end{align*}
\]

(2.112)

As a result, the prior mean and variance of \( w_t \) are approximated by

\[
\begin{align*}
\hat{w}_{t|t-1} &\approx c_0^f + c_1^f \left(-1 + \frac{2}{b-a}(\hat{w}_{t-1|t-1} - a)\right) \\
P_{t|t-1} &\approx (c_1^f)^2 \left(\frac{2}{b-a}\right)^2 P_{t-1|t-1} + q_t. \
\end{align*}
\]

(2.113)

(2.114)

In a similar way the mean and variance of the one-step ahead forecast can be approximated as follows:

\[
\begin{align*}
\hat{y}_{t|t-1} &\approx c_0^h + c_1^h \left(-1 + \frac{2}{b-a}(\hat{w}_{t|t-1} - a)\right) \\
s_{t|t-1} &\approx (c_1^h)^2 \left(\frac{2}{b-a}\right)^2 P_{t|t-1} + r_t, \
\end{align*}
\]

(2.115)

(2.116)
where the superscript $h$ of the coefficients indicates that they are for the observation function. After receiving a new observation $y_t$, the prior mean and variance of $w_t$ are updated according to the following equations:

$$\hat{w}_{t|t} = \hat{w}_{t|t-1} + K_t (y_t - \hat{y}_{t|t-1}), \quad (2.117)$$
$$P_{t|t} = P_{t|t-1} - s_{t|t-1} K_t^2, \quad (2.118)$$

where

$$K_t = c_t^h \left( \frac{2}{b - a} \right) \frac{P_{t|t-1}}{s_{t|t-1}}. \quad (2.119)$$

In order to implement this Chebyshev Kalman filter we need to determine the approximation range of $w_t$, i.e. $a$ and $b$. We suggest to use \{mean $\pm$ 3 $\cdot$ (standard deviation)\} as the upper and lower bounds of the interval; for example, in the computation of $\hat{w}_{t|t-1}$ the interval would be $[\hat{w}_{t-1|t-1} - 3 \sqrt{P_{t-1|t-1}}, \hat{w}_{t-1|t-1} + 3 \sqrt{P_{t-1|t-1}}]$; or alternatively, we can use $[\hat{w}_{t-1|t-1} - 1, \hat{w}_{t-1|t-1} + 1]$.

### 2.4.4 Simulation

In this subsection we examine a performance of the Chebyshev Kalman filter when applied to non-linear dynamic models. The performance will be compared with that of the extended Kalman filter and the unscented Kalman filter. We consider three dynamic non-linear models (Table 2.1); they are different in terms of the degree of non-linearity. Observation noise variance $r_t$ and state noise variance $q_t$ are commonly set to 1. We computed the root mean square error of each filter over time span of $t = 1,000$ after 100 runs (Table 2.2). Despite very small differences of the root mean square error, the unscented Kalman filter uniformly outperformed the extended Kalman filter and the Chebyshev Kalman filter. We can also find that the Chebyshev Kalman filter performed slightly better than the extended Kalman filter.

We have proposed a novel non-linear Kalman filter based on the Chebyshev approximation. Despite the filter’s performance as good as other existing non-linear filters, the extended Kalman filter and the unscented Kalman filter, shown by simulation work, it has a major disadvantage: it is not straightforward to generalise the Chebyshev Kalman filter from a scalar
2.4 Non-linear Kalman Filters

<table>
<thead>
<tr>
<th>Observation model</th>
<th>State model</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_t = w_t + n_t )</td>
<td>( w_t = \frac{1}{200} w_{t-1}^2 + 3 + v_t )</td>
</tr>
<tr>
<td>( y_t = w_t + n_t )</td>
<td>( w_t = \frac{1}{2} w_{t-1} + \frac{25w_{t-1}}{1+w_{t-1}^2} + v_t )</td>
</tr>
<tr>
<td>( y_t = \frac{1}{2} w_t + \frac{25w_t}{1+w_t^2} + n_t )</td>
<td>( w_t = \frac{1}{2} w_{t-1} + \frac{25w_{t-1}}{1+w_{t-1}^2} + v_t )</td>
</tr>
</tbody>
</table>

Table 2.1: Three dynamic non-linear models used to test non-linear filters.

<table>
<thead>
<tr>
<th></th>
<th>EKF</th>
<th>UKF</th>
<th>CKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.7057</td>
<td>0.7057</td>
<td>0.7057</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.7087</td>
<td>0.7084</td>
<td>0.7084</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.9522</td>
<td>0.9313</td>
<td>0.9515</td>
</tr>
</tbody>
</table>

Table 2.2: Root mean square error of the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the Chebyshev Kalman filter (CKF) on the three dynamic non-linear models.

variable case to a multivariate one. This fact prevents its use in real-world problems, which requires further research. However, it is worth knowing that other approximation methods besides the Taylor series approximation can be employed in the framework of the recursive optimal estimation. Other filters dealing with dynamic non-linear models employ sequential Monte Carlo sampling methods such as the particle filter (Doucet et al., 2001).
Chapter 3

Dynamic Generalised Linear Models

We have discussed that the Kalman filter provides a recursive, optimal algorithm for dynamic linear models and its modified algorithms are successfully applied to dynamic non-linear systems. Since they are mathematically convenient and computationally straightforward, the filters have been widely used in real-world problems. However, its restriction to Gaussian driven processes prevents its wider use; there are many situations in which observations appear to follow distributions (e.g. Poisson, binomial, exponential, etc.) other than a Gaussian distribution. An exponential family is a class of probability distributions sharing a certain form as shown later in Equation (3.3), which is chosen because of some useful algebraic properties. Also the distributions in the exponential family are natural distributions to consider in a sense that they reflect many real-world phenomena. Generalised linear models are a flexible generalisation of ordinary linear regression models and they are applicable to non-Gaussian distributions of the exponential family. In this chapter we review the generalised linear models and introduce a framework in which they are applied to dynamic systems.

3.1 Generalised Linear Models

In ordinary linear regression models a scalar observation random variable \( y \) is modelled by

\[
y = h^T w + n,
\]

(3.1)

where \( h \) represents a known input variable (referred to as predictor variable) and \( w \) is a model parameter vector; \( n \) is a Gaussian error variable with mean 0 and variance \( \sigma^2 \). The
observation variable is also a Gaussian distribution with the following mean and variance

$$E(y) = \mu = h^T w, \quad \text{Var}(y) = \sigma^2.$$  \hfill (3.2)

Note that the mean is a linear function of the predictor variable, $h$.

In generalised linear models two generalisations are made from the standard linear models above: (1) the probability distribution of an observation variable is not restricted to be a Gaussian but is allowed to have any distribution in the exponential family; (2) it is not the mean of the observation variable but a function of the mean that is linearly related to a predictor variable.

An observation random variable $y$ that has a distribution in the exponential family takes the form,

$$p(y|\theta, \phi) = \exp \left( \frac{d(y)\theta - b(\theta)}{a(\phi)} + c(y, \phi) \right),$$  \hfill (3.3)

where $\theta$ is called canonical parameter and $\phi$ dispersion parameter, and $a, b, c, d$ are all known functions. The mean and variance of the variable are related to the canonical and dispersion parameters through the known functions as follows:

$$E(d(y)|\theta, \phi) = \mu = b'(\theta),$$  \hfill (3.4)

$$\text{Var}(d(y)|\theta, \phi) = b''(\theta)a(\phi).$$  \hfill (3.5)

Members of the exponential family include Gaussian, binomial, Poisson, exponential, gamma, inverse Gaussian distributions, etc.

In the ordinary Gaussian linear model the mean is linearly related to the predictor variable, $h$; that is,

$$\mu = h^T w.$$  \hfill (3.6)

The right-hand side of the equation is denoted by $\eta \triangleq h^T w$, and $\eta$ is referred to as a linear predictor. Note that the linear predictor and the mean of a Gaussian variable can take any real values. However, this may or may not be true for other distributions in the exponential family; for instance, the mean of a Poisson distribution must be only positive. Hence, we
need a way of relating the linear predictor to the mean, which is given by

$$g(\mu) = \eta = h^T w.$$  \hfill (3.7)

Here $g$ is referred to as a *link function*. The mean can be computed simply by inverting the link function:

$$\mu = g^{-1}(\eta) = g^{-1}(h^T w).$$  \hfill (3.8)

From Equation (3.6) we can see that the link function for the standard Gaussian linear model is an identity function. A particular form of the link function is of importance. A link function that makes the linear predictor $\eta$ equal to the canonical parameter $\theta$ is called a *canonical link*:

$$g(\mu) = \eta = \theta.$$  \hfill (3.9)

The canonical link is important due to the existence of minimal sufficient statistics for the model parameter, thereby being used in most cases (McCullagh and Nelder, 1989).

### 3.2 Exponential Family Dynamic Models

Summarising the generalised linear model, there are two key characteristics: (1) the non-Gaussianity of an observation variable and (2) the non-linear relation of the mean to a predictor variable. We can view dynamic generalised linear models in two different ways. Firstly, it is a “generalised” version of the dynamic linear model that allows an observation variable to be non-Gaussian. Secondly, it is a “dynamic” version of the generalised linear model that allows model parameters to vary over time. We here introduce the dynamic generalised linear model for a scalar observation variable, which was first proposed by West et al. (1985). For multivariate cases, refer to Fahrmeir and Tutz (2001).

Suppose that an observation variable $y$ having a distribution in the exponential family is observed over time. The *observation model* of a dynamic generalised linear model is of form

$$p(y_t|\theta_t, \phi_t) = \exp \left( \frac{d(y_t)\theta_t - b(\theta_t)}{a(\phi_t)} + c(y_t, \phi_t) \right),$$  \hfill (3.10)
3.2 Exponential Family Dynamic Models

with the following link equation

\[ g(\mu_t) = \eta_t = H_t^T w_t, \tag{3.11} \]

where \( \mu_t = \mathbb{E}(y_t) \). If \( y_t \) has a Gaussian distribution, then the observation model and link equation can be merged into a single observation equation as follows:

\[ y_t = H_t^T w_t + n_t, \tag{3.12} \]

where \( n_t \sim N(0, r_t) \) and \( \mu_t = H_t^T w_t \). This is equivalent to the observation equation of the dynamic linear model detailed in Section 2.2. The state variable (or model parameter vector) \( w_t \) is assumed not to be static but to evolve over time; the state process is

\[ w_t = F_t w_{t-1} + v_t, \tag{3.13} \]

where \( v_t \sim (0, Q_t) \). Notation \( a \sim (b, c) \) represents that the mean and variance of variable \( a \) are \( b \) and \( c \) respectively without specifying the probability density of the variable.

We review a sequential inference method for the state variable in the Bayesian setting (West and Harrison, 1997). It is an approximate method in a sense that distributions are only specified in terms of their first and second moments without full distributional information. Suppose that, given \( D_{t-1} = \{y_1, \ldots, y_{t-1}\} \), the state variable \( w_{t-1} \) has mean \( \hat{w}_{t-1|t-1} \) and covariance \( P_{t-1|t-1} \); that is,

\[ w_{t-1|D_{t-1}} \sim (\hat{w}_{t-1|t-1}, P_{t-1|t-1}). \tag{3.14} \]

Since the state process in Equation (3.13) is linear, the prior distribution of the state variable at \( t \) can be easily computed and it is of form

\[ w_{t|D_{t-1}} \sim (\hat{w}_{t|t-1}, P_{t|t-1}), \tag{3.15} \]

where

\[ \hat{w}_{t|t-1} = F_t \hat{w}_{t-1|t-1}, \tag{3.16} \]

\[ P_{t|t-1} = F_t P_{t|t-1} F_T + Q_t. \tag{3.17} \]
We note that what is linearly related to the state variable is not the mean of the observation variable but a function of the mean, i.e. \( g(\mu_t) = \eta_t = h^T_t w_t \). Owing to this linearity, we can readily compute the prior distribution of \( \eta_t \); the joint prior distribution of \( \eta_t \) and \( w_t \) given \( D_{t-1} \) is

\[
\begin{bmatrix} \eta_t \\ w_t \end{bmatrix} \mid D_{t-1} \sim \left( \begin{bmatrix} \hat{\eta}_{t|t-1} \\ \hat{w}_{t|t-1} \end{bmatrix}, \begin{bmatrix} S_{t|t-1} & h_t P_{t|t-1} \\ h^T_t P^T_{t|t-1} & h_t P_{t|t-1} \end{bmatrix} \right),
\]

(3.18)

where

\[
\hat{\eta}_{t|t-1} = h^T_t \hat{w}_{t|t-1},
\]

(3.19)

\[
s_{t|t-1} = h^T_t P_{t|t-1} h_t.
\]

(3.20)

The one-step ahead forecast distribution (i.e. the distribution of \( y_t \) given \( D_{t-1} \)) can be obtained by marginalising over the canonical parameter assuming that the dispersion parameter is known:

\[
p(y_t | D_{t-1}) = \int p(y_t | \theta_t) p(\theta_t | D_{t-1}) d\theta_t.
\]

(3.21)

To solve the equation we need to specify the prior distribution of the canonical parameter, \( p(\theta_t | D_{t-1}) \). A conjugate prior is widely used because it allows us to derive the forecast distribution analytically. The conjugate prior distribution is of form

\[
p(\theta_t | D_{t-1}) = \tau(k_t, m_t) \exp(k_t \theta_t - m_t b(\theta_t)),
\]

(3.22)

where \( \tau \) is a known function providing the normalising constant, and \( k_t \) and \( m_t \) are hyperparameters of the conjugate prior. A complete form of the one-step ahead forecast distribution is obtained in the following way:

\[
p(y_t | D_{t-1}) = \int p(\theta_t | D_{t-1}) p(y_t | \theta_t) d\theta_t
\]

\[
= \int \tau(k_t, m_t) \exp(k_t \theta_t - m_t b(\theta_t)) \exp \left( \frac{d(y_t) \theta_t - b(\theta_t)}{a(\phi_t)} + c(y_t, \phi_t) \right) d\theta_t
\]

\[
= \tau(k_t, m_t) \exp(c(y_t, \phi_t)) \int \exp \left( k_t + \frac{d(y_t)}{a(\phi_t)^2} \theta_t - m_t + \frac{1}{a(\phi_t)^2} b(\theta_t) \right) d\theta_t
\]

\[
= \frac{\tau(k_t + \frac{d(y_t)}{a(\phi_t)^2}, m_t + \frac{1}{a(\phi_t)^2})}{\tau(k_t, m_t)} \exp(c(y_t, \phi_t)).
\]

(3.23)
3.2 Exponential Family Dynamic Models

In order to employ the one-step ahead forecast distribution we need to determine the values of the hyper-parameters of the conjugate prior, \( k_t \) and \( m_t \). If a canonical link is used, which ensures \( \theta_t = \eta_t \), then we know that

\[
E(\theta_t|D_{t-1}) = E(\eta_t|D_{t-1}) = \hat{\eta}_{t|t-1},
\]
\[
\text{Var}(\theta_t|D_{t-1}) = \text{Var}(\eta_t|D_{t-1}) = s_{t|t-1}.
\]

(3.24)
(3.25)

From the form of the conjugate prior in Equation (3.22), we know \( E(\theta_t|D_{t-1}) \) and \( \text{Var}(\theta_t|D_{t-1}) \) in terms of the hyper-parameters. In addition, we know the values of \( \hat{\eta}_{t|t-1} \) and \( s_{t|t-1} \) that are given in Equation (3.20). We can, therefore, calculate the values of the hyper-parameters.

According to Bayes’ theorem, the posterior distribution of the canonical parameter \( \theta_t \) can be represented by

\[
p(\theta_t|D_t) \propto p(\theta_t|D_{t-1})p(y_t|\theta_t).
\]

(3.26)

In a similar way to the derivation of \( p(y_t|D_{t-1}) \) given in Equation (3.23) we can compute the full distribution of \( p(\theta_t|D_t) \):

\[
p(\theta_t|D_t) = \tau \left( k_t + \frac{d(y_t)}{a(\phi_t)}, m_t + \frac{1}{a(\phi_t)} \right) \exp \left( \left( k_t + \frac{d(y_t)}{a(\phi_t)} \right) \theta_t - \left( m_t + \frac{1}{a(\phi_t)} \right) b(\theta_t) \right).
\]

(3.27)

Since using the canonical link makes the posterior distribution of \( \eta_t \) equivalent to that of \( \theta_t \), we have the posterior distribution of \( \eta_t \) of from

\[
\eta_t|D_t \sim (\hat{\eta}_{t|t}, s_{t|t}),
\]

(3.28)

where

\[
\hat{\eta}_{t|t} = E(\eta_t|D_t) = E(\theta_t|D_t),
\]

(3.29)

\[
s_{t|t} = \text{Var}(\eta_t|D_t) = \text{Var}(\theta_t|D_t).
\]

(3.30)

Here \( E(\theta_t|D_t) \) and \( \text{Var}(\theta_t|D_t) \) can be computed from Equation (3.27).

By Bayes’ theorem we can rewrite the posterior distribution of the state variable as fol-
3.2 Exponential Family Dynamic Models

\[
p(w_t|D_t) = \int p(\eta_t, w_t|D_t) d\eta_t = \int p(w_t|\eta_t, D_{t-1}) p(\eta_t|D_t) d\eta_t
\]  
(3.31)

A detailed derivation is given in Appendix A.1. The first two moments of \(p(\eta_t|D_t)\) are given by Equations (3.29) and (3.30). The mean and covariance of \(p(w_t|\eta_t, D_{t-1})\) are estimated using linear Bayes’ estimation (pages 122–128 of West and Harrison (1997)), and are given by

\[
\hat{E}(w_t|\eta_t, D_{t-1}) = \hat{w}_{t|t-1} + \frac{P_{t|t-1}^T h_t}{s_{t|t-1}} (\eta_t - \hat{\eta}_{t|t-1}), \\
\hat{\text{Var}}(w_t|\eta_t, D_{t-1}) = P_{t|t-1} - \frac{P_{t|t-1}^T h_t}{s_{t|t-1}} h_T P_{t|t-1}.
\]  
(3.32), (3.33)

From Equation (3.31) we can write the expectation of the posterior distribution of \(w_t\) as follows:

\[
E(w_t|D_t) = \int w_t p(w_t|D_t) dw_t = \int w_t \int p(w_t|\eta_t, D_{t-1}) p(\eta_t|D_t) d\eta_t dw_t = \int \int w_t p(w_t|\eta_t, D_{t-1}) dw_t p(\eta_t|D_t) d\eta_t = \int E(w_t|\eta_t, D_{t-1}) p(\eta_t|D_t) d\eta_t = E[E(w_t|\eta_t, D_{t-1})|D_t]. \\
\]  
(3.34)

Similarly the variance of the posterior distribution of \(w_t\) can be written as

\[
\text{Var}(w_t|D_t) = \text{Var}[E(w_t|\eta_t, D_{t-1})|D_t] + E[\text{Var}(w_t|\eta_t, D_{t-1})|D_t].
\]  
(3.35)

Lastly, if we combine Equations (3.32) and (3.33) with Equations (3.34) and (3.35), then we can obtain the posterior mean and covariance of \(w_t\):

\[
w_t|D_t \sim (\hat{w}_{t|t}, P_{t|t}),
\]  
(3.36)
where

\[
\hat{w}_{t|t} = \mathbb{E}[\mathbb{E}(\mathbf{w}_t | \eta_t, D_{t-1}) | D_t]
= \mathbb{E} \left[ \hat{w}_{t|t-1} + \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} (\hat{\eta}_t - \hat{\eta}_{t|t-1}) | D_t \right]
= \hat{w}_{t|t-1} + \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} (\mathbb{E}(\eta_t | D_t) - \hat{\eta}_{t|t-1})
= \hat{w}_{t|t-1} + \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} (\hat{\eta}_t - \hat{\eta}_{t|t-1}),
\]

and

\[
P_{t|t} = \text{Var}[\mathbb{E}(\mathbf{w}_t | \eta_t, D_{t-1}) | D_t] + \mathbb{E}[\text{Var}(\mathbf{w}_t | \eta_t, D_{t-1}) | D_t]
= \text{Var} \left[ \hat{w}_{t|t-1} + \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} (\hat{\eta}_t - \hat{\eta}_{t|t-1}) | D_t \right] + \mathbb{E} \left[ \mathbf{P}_{t|t-1} - \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} \mathbf{h}_t^T \mathbf{P}_{t|t-1} | D_t \right]
= \frac{\mathbf{P}_{t|t-1} \mathbf{h}_t}{(s_{t|t-1})^2} \mathbf{h}_t^T \mathbf{P}_{t|t-1} \text{Var}(\eta_t | D_t) + \mathbf{P}_{t|t-1} - \frac{\mathbf{P}_t^T \eta_t}{s_{t|t-1}} \mathbf{h}_t^T \mathbf{P}_{t|t-1}
= \mathbf{P}_{t|t-1} - \frac{\mathbf{P}_{t|t-1} \mathbf{h}_t}{s_{t|t-1}} \mathbf{h}_t^T \mathbf{P}_{t|t-1} \left( 1 - \frac{s_{t|t}}{s_{t|t-1}} \right).
\]

Summarising the discussion above, we can represent an exponential family dynamic model in the following two steps:

**Prediction Step**

\[
\begin{align*}
\hat{w}_{t|t-1} &= \mathbf{F}_t \hat{w}_{t-1|t-1} \\
\mathbf{P}_{t|t-1} &= \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t \\
\hat{\eta}_{t|t-1} &= \mathbf{h}_t^T \hat{w}_{t|t-1}, \\
s_{t|t-1} &= \mathbf{h}_t^T \mathbf{P}_{t|t-1} \mathbf{h}_t.
\end{align*}
\]

**Update Step**

\[
\begin{align*}
\hat{\eta}_{t|t} &= \mathbb{E}(\theta_t | D_t) \\
s_{t|t} &= \text{Var}(\theta_t | D_t) \\
\hat{w}_{t|t} &= \hat{w}_{t|t-1} + \frac{\mathbf{P}_{t|t-1} \mathbf{h}_t}{s_{t|t-1}} (\hat{\eta}_{t|t} - \hat{\eta}_{t|t-1}) \\
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \frac{\mathbf{P}_{t|t-1} \mathbf{h}_t}{s_{t|t-1}} \mathbf{h}_t^T \mathbf{P}_{t|t-1} \left( 1 - \frac{s_{t|t}}{s_{t|t-1}} \right).
\end{align*}
\]
We have so far demonstrated a general derivation procedure for distributions in the exponential family. Triantafyllopoulos (2008) applied various distributions including binomial, Poisson, gamma, Weibull and Pareto to forecasting problems in the dynamic model framework. We will look into the dynamic binomial model in Chapter 6 where we deal with the problem of sequential binary classification. For a general treatment of Bayesian generalised linear models, refer to Dey et al. (2000); and for multivariate dynamic generalised linear models, see Fahrmeir and Tutz (2001).
Chapter 4

Making a Sequence of Optimal Decisions

An agent often needs to make decisions and take actions over time. It does not merely aim to make a decision that returns high immediate rewards, but is interested in finding a sequence of decisions that will lead to high long term rewards. In particular, it is sometimes given incomplete knowledge of the environments; that is, the decision and actions need to be made with limited information and observations. The core of techniques typically designed to deal with this type of decision making problem is based on ‘trial-and-error’; the decisions are made after carrying out many trials and learning from the consequent errors. In this chapter, we briefly review Markov decision processes that help to formulate the problem, and reinforcement learning that is a model-free approach to making a sequence of decisions over time. We discuss how the problem of continuous state and action spaces can be dealt with in the reinforcement learning framework. In addition, we modify a traditional reinforcement learning algorithm in such way that it is suitable for a non-stationary environment.

4.1 Markov Decision Processes

Ross (1983) gave an excellent description of sequential decision making problems:

A problem typical of those with which we are concerned involves a process that is observed at the beginning of a discrete time period to be in a particular state. After observation of the state, an action must be chosen; and based only on the state at that time and the action chosen, an expected reward is earned and the
probability distribution for the next state is determined. The problem of interest is to choose a policy that maximises the expected value of the sum of the rewards earned over a given finite time span of length \( n \).

*Markov decision processes* formulate such a sequential decision making problem (e.g. Sutton (1997); Feinberg and Shwart (2002)). For continuous time cases refer to the theory of semi-Markov decision processes (Bradtke and Duff, 1995). A Markov decision process consists of the following four elements:

- **States**: \( w_t \) represents a state at time \( t \).
- **Actions**: \( a_t \) represents an action at time \( t \).
- **Transition probability function**: \( \Pr(w_{t+1}|w_t,a_t) \) is defined as the probability that performing action \( a_t \) in state \( w_t \) transforms the state into another state \( w_{t+1} \). For discrete states and actions we simplify the notation of the transition probability function to \( P_{ij}(a) \triangleq \Pr(w_{t+1} = j|w_t = i, a_t = a) \).
- **Reward function**: \( r(w_t,a_t,w_{t+1}) \) is defined as a real-valued reward that an agent receives when performing action \( a_t \) in state \( w_t \) and ending up in next state \( w_{t+1} \). We use a simplified notation \( r_{t+1} \) for the reward.

An action is a function of a state, which is represented by

\[
a_t = \pi(w_t),
\]

where \( \pi \) maps a state space into an action space; \( \pi \) is referred to as a policy or an action function. Summarising a Markov decision process, a state \( w_t \) is transformed into another state \( w_{t+1} \) through an action \( a_t \), and consequently a reward \( r_{t+1} \) is earned (Figure 4.1).

This Markov decision process formulation can be replaced with that of stochastic control, in which the transition probability function \( \Pr(w_{t+1}|w_t,a_t) \) is represented by a stationary discrete-time dynamic system:

\[
w_{t+1} = f(w_t,a_t,v_t),
\]
4.1 Markov Decision Processes

where $v_t$ is a random disturbance term. Both formulations are essentially equivalent (Bertsekas, 2005, 2007), so we will continue with the Markov decision process formulation throughout this chapter. In the Markov decision process we assume that states are completely observable, which is of our interest in this chapter. However, algorithms for partially observable states have been developed under the name of partially observable Markov decision processes (e.g. Kaelbling et al. (1998)).

We aim to find an optimal policy that maximises the expected value of the sum of the rewards earned over a long time span; that is, the following expected total discounted reward:

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) = E\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right),$$

(4.3)

where $0 < \gamma < 1$ is a discount factor. Note that $\gamma$ decreases exponentially over time, which means that the further away from the present time a reward is, the less significant its importance becomes.

4.1.1 Dynamic Programming

Dynamic programming is a method of computing an optimal policy given a Markov decision process. We consider an infinite-horizon, finite-state, discrete-time Markov decision process. Given an initial state $w_0$ and a policy $\pi$, the sum of expected future rewards of $\pi$ starting at

![Figure 4.1: A diagram of a Markov decision process.](image)
4.1 Markov Decision Processes

$w_0$ is defined by

$$V^\pi(w_0) = \lim_{N \to \infty} E \left[ \sum_{t=0}^{N-1} \gamma^t r(w_t, \pi(w_t), w_{t+1}) \right].$$

(4.4)

Here $V^\pi$ is referred to as a value function of $\pi$, and $V^\pi(w_0)$ as the value of $w_0$. An optimal policy denoted by $\pi^*$ is defined as a policy which maximises the above value function; that is,

$$V^{\pi^*}(w_0) = \max_{\pi} V^\pi(w_0),$$

(4.5)

where $V^{\pi^*}$ is referred to as an optimal value function. We will use a simplified notation $V^*$ for the optimal value function. It satisfies the following recursive relation:

$$V^*(w_t) = \max_a E \left[ r(w_t, a_t, w_{t+1}) + \gamma V^*(w_{t+1}) \right].$$

(4.6)

This is known as Bellman's optimality equation. In the case of discrete states and actions, Bellman's optimality equation can be rewritten as

$$V^*(i) = \max_a \sum_j P_{ij}(a) \left[ r(i, a, j) + \gamma V^*(j) \right].$$

(4.7)

There are two widely-used dynamic programming algorithms, namely value iteration and policy iteration (see Section 1.3 of Bertsekas (2007) for proofs of what follows). These iterative algorithms generate a value function that converges to an optimal value function; an optimal policy is obtained from the optimal value function. We review the two algorithms in the discrete state and action case.

Firstly, the value iteration is a successive approximation procedure of computing an optimal value function. It operates as follows: an arbitrary value is first assigned to $V_0(i)$ for all $i$ and iteratively update the value of $i$ according to

$$V_{k+1}(i) = \max_a \sum_j P_{ij}(a) \left[ r(i, a, j) + \gamma V_k(j) \right] \quad \text{for} \quad k = 0, 1, 2, \ldots,$$

(4.8)

where $k$ denotes the iteration number. We continue to update the values of all states $i$ until the difference between $V_{k+1}(i)$ and $V_k(i)$ becomes small. The value function $V_k$ is considered as an approximation to an optimal value function $V^*$. Once having obtained the optimal value
function, it is straightforward to compute an optimal policy. Since an optimal policy, \( \pi^* \), is defined as a policy that produces the optimal value function \( V^* \), we can compute it using Bellman’s optimality equation as follows:

\[
\pi^*(i) = \arg\max_a \sum_j P_{ij}(a) \left[ r(i, a, j) + \gamma V^*(j) \right] \quad \text{for all } i. \tag{4.9}
\]

Policy iteration is the second dynamic programming algorithm we consider; it consists of two steps, namely \textit{policy evaluation} and \textit{policy improvement}. The policy iteration algorithm starts with a given policy, say \( \pi \). This contrasts with the value iteration method, in which we first compute an optimal value function iteratively and then obtain an optimal policy. For any stationary policy \( \pi \), its value function should satisfy the following equation:

\[
V^\pi(i) = \sum_j P_{ij}(\pi(i)) \left[ r(i, \pi(i), j) + \gamma V^\pi(j) \right] \quad \text{for all } i. \tag{4.10}
\]

In the policy evaluation step \( V^\pi \) is iteratively computed according to

\[
V^\pi_{k+1}(i) = \sum_j P_{ij}(\pi(i)) \left[ r(i, \pi(i), j) + \gamma V^\pi_k(j) \right] \quad \text{for all } i. \tag{4.11}
\]

Here \( V^\pi_k \) converges to the true value function \( V^\pi \) satisfying Equation (4.10). In the policy improvement step the given policy \( \pi \) is iteratively updated as follows:

\[
\pi_{\text{new}}(i) := \arg\max_a \sum_j P_{ij}(a) \left[ r(i, a, j) + \gamma V^\pi_{\text{old}}(j) \right] \quad \text{for all } i. \tag{4.12}
\]

The policy iteration repeats the policy evaluation and policy improvement steps until no changes are made between two successive policies.

The classical dynamic programming algorithms have two characteristics. Firstly, they compute an optimal value function \( V^* \) by storing the values of all states on a state space in a ‘lookup’ table. Secondly, a transition probability function \( P_{ij}(a) \) and a reward function \( r(i, a, j) \) are required to compute a value function and to update a policy; in other words, the dynamic programming algorithms assume that environments are completely known. These characteristics imply limitations to the algorithms. The first limitation is that for a large state space, representing the values of all states in a lookup table is restrictive as computational cost and memory storage capacity necessary increase exponentially; this phenomenon
is known as “curse of dimensionality” (Bellman, 1957). Secondly, in many real-world problems information about the environments is not available; for example, the transition probability function and the reward function are not known. In the next section we discuss the reinforcement learning that aims to tackle these difficulties of the dynamic programming.

4.2 Reinforcement Learning

Reinforcement learning is a class of suboptimal methods that aim to solve incompletely known Markov decision processes (Sutton and Barto, 1998). It is also known as ‘neuro-dynamic programming’ (Bertsekas and Tsitsiklis, 1996). The reinforcement learning approach is based on two ideas: (1) function approximation and (2) the use of a sequence of observations.

Consider a situation in which the number of states is too large or a state space is continuous. It would be inefficient or impossible to store the values of the states in a lookup table. Hence, we approximate a value function $V(w)$ by $\hat{V}(w, \theta)$, where $\theta$ is the parameter vector of the approximation model $\hat{V}$. In this parametric representation it is common to use the method of feature extraction. We map the state space onto the feature space; for example, a feature $f$ transforms $w$ into $f(w)$. The feature vector of size $m$ is defined by $f(w) = (f_1(w), \ldots, f_m(w))$. With this feature vector we can approximate the value function using a function approximation method as follows:

$$V(w) \approx \hat{V}(w, \theta) = g(f(w), \theta),$$

(4.13)

where $g$ is an approximation function; it can be for example a linear regression function, or a non-linear function. Examples of the latter include the use of feedforward neural networks to model $g(\cdot)$ (Tesauro, 1992, 1995).

By using a sequence of observed data we can solve the problem of the lack of environmental knowledge. Suppose that a sampled trajectory is generated under a given policy $\pi$:

$$\{w_0, \pi(w_0), w_1, r_1, \pi(w_1), w_2, r_2, \pi(w_2), w_3, r_3, \ldots, w_{T-1}, \pi(w_{T-1}), w_T, r_T\},$$

(4.14)
where $T$ is a terminal stage. In the supervised learning context, unknown parameters of a model are estimated with a set of training data consisting of input-output pairs. We can construct the training set with the observed trajectory: an output sample of an input $w_t$ can be thought of as $v(w_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$, and thus we have a set of input-output training data, $\{w_t, v(w_t)\}$. As with the supervised learning methods, we can estimate the parameter vector $\theta$ of $V^\pi(w_t, \theta)$ from the training set. This approach is a simple example of Monte Carlo methods for reinforcement learning (refer to Chapter 5 of Sutton and Barto (1998)).

\subsection{Temporal-Difference Learning}

In order to implement the above Monte Carlo method, a batch of trajectories is necessary. However, there are many situations in which no terminal stages exist; it is impossible to have a complete set of training data. As a result, we are interested in on-line methods of solving the problem of incomplete Markov decision processes. We introduce an on-line reinforcement learning algorithm named temporal-difference learning. It originated in Samuel (1959), and Sutton (1988) generalised the algorithm and referred to it as TD($\lambda$) learning.

To assist understanding, we first consider the lookup table representation of the value function. Note that the value function of a given policy $\pi$ satisfies the following equation:

$$V^\pi(w_t) = E[r(w_t, \pi(w_t), w_{t+1}) + \gamma V^\pi(w_{t+1})].$$

(4.15)

We have only an observation sequence of successive states and a reward, i.e. $\{w_t, w_{t+1}, r_{t+1}\}$. Since a transition probability function and a reward function are unknown, the expectation term of Equation (4.15) cannot be evaluated. Instead, the value of $w_t$ is estimated with the observation sequence $\{w_t, w_{t+1}, r_{t+1}\}$ as follows:

$$V^\pi(w_t) \approx r_{t+1} + \gamma V^\pi(w_{t+1}).$$

(4.16)

Ideally the values on the two sides should be equal. The temporal difference learning is based on a difference between these two values: $V^\pi(w_t)$ is the current estimate value of $w_t$; $r_{t+1} + \gamma V^\pi(w_{t+1})$ is a new estimate obtained after receiving a reward. The difference is
referred to as *temporal difference* defined by

\[
\xi_t \triangleq r_{t+1} + \gamma V^\pi(w_{t+1}) - V^\pi(w_t). \tag{4.17}
\]

When the temporal difference is not zero, the current estimate is adjusted so as to maximally reduce the difference as follows:

\[
V^\pi_{(\text{new})}(w_t) := V^\pi_{(\text{old})}(w_t) + \alpha \xi_t, \tag{4.18}
\]

where \(\alpha\) is a learning parameter. This is the temporal-difference update rule.

For the parametric representation, the true value function is approximated by \(\hat{V}^\pi(w_t, \theta)\). Therefore, instead of updating the value of each state, the parameter vector \(\theta\) is adjusted so as to move the current approximation towards an improved one based on a new observation sequence \(\{w_t, w_{t+1}, r_{t+1}\}\). The temporal-difference update rule for the parametric representation is

\[
\theta_{(\text{new})} := \theta_{(\text{old})} + \alpha \xi_t \nabla_\theta \hat{V}^\pi(w_t, \theta_{(\text{old})}), \tag{4.19}
\]

where \(\nabla_\theta \hat{V}^\pi\) is the partial derivative of \(\hat{V}^\pi\) with respect to \(\theta\). Here the temporal difference is \(\xi_t = r_{t+1} + \gamma \hat{V}^\pi(w_{t+1}, \theta_{(\text{old})}) - V^\pi(w_t, \theta_{(\text{old})}).\)

Before discussing the TD(\(\lambda\)) learning, we re-consider the Monte Carlo approach. With a complete trajectory an estimate of the true value of \(w_t\) would be

\[
V^\pi_{(MC)}(w_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T, \tag{4.20}
\]

where \(T\) is a terminal stage. We refer to this as a Monte Carlo estimate. A difference between this estimate and the current estimate would be represented by

\[
V^\pi_{(MC)}(w_t) - V^\pi_{(\text{old})}(w_t) = (r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T) - V^\pi_{(\text{old})}(w_t). \tag{4.21}
\]

Therefore, we could have a temporal-difference-type update rule using the Monte Carlo estimate, which is

\[
V^\pi_{(\text{new})}(w_t) := V^\pi_{(\text{old})}(w_t) + \alpha \left[ (r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T) - V^\pi_{(\text{old})}(w_t) \right]. \tag{4.22}
\]
This update rule can be rewritten in terms of temporal differences as follows:

\[
V_{(new)}^{\pi}(w_t) := V_{(odd)}^{\pi}(w_t) + \alpha \left[ \xi_t + \gamma \xi_{t+1} + \cdots + \gamma^{T-t-1} \xi_{T-1} \right],
\]

(4.23)

because

\[
\begin{align*}
&\left( r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T - V^{\pi}(w_t) \right) \\
= &\left\{ r_{t+1} + \gamma V^{\pi}(w_{t+1}) - V^{\pi}(w_t) \right\} + \gamma \left\{ r_{t+2} + \gamma V^{\pi}(w_{t+2}) - V^{\pi}(w_{t+1}) \right\} + \cdots \\
&+ \gamma^{T-t-2} \left\{ r_{T-1} + \gamma V^{\pi}(w_{T-1}) - V^{\pi}(w_{T-2}) \right\} + \gamma^{T-t-1} \left\{ r_T - V^{\pi}(w_{T-1}) \right\} \\
= &\xi_t + \gamma \xi_{t+1} + \cdots + \gamma^{T-t-1} \xi_{T-1}.
\end{align*}
\]

(4.24)

In this Monte Carlo approach, the expected value of the expression given Equation (4.20) is \( V^{\pi}(w_t) \), therefore it is an unbiased estimate of \( V^{\pi}(w_t) \). However, it has a large variance as it sums the variances of the stochastic rewards, which is an inherent drawback of Monte Carlo methods. Meanwhile, for the temporal-difference update rule in Equation (4.18), the new estimate of \( V^{\pi}(w_t) \) is a biased one as it is based on an estimated value of the next state, i.e. \( V^{\pi}(w_{t+1}) \), as shown in Equation (4.16). However, it has a small variance. Sutton (1984, 1988) suggested the TD(\( \lambda \)) algorithm that circumvents this dilemma of ‘small bias versus small variance’. The TD(\( \lambda \)) update rule is

\[
V_{(new)}^{\pi}(w_t) := V_{(odd)}^{\pi}(w_t) + \alpha \left[ \xi_t + (\gamma \lambda) \xi_{t+1} + \cdots + (\gamma \lambda)^{T-t-1} \xi_{T-1} \right].
\]

Here the parameter \( \lambda \) plays a role of assigning an exponential discount on future temporal differences. Note that the update rules given in Equations (4.18) and (4.23) are equivalent to TD(0) and TD(1) respectively.

The TD(\( \lambda \)) update rule of Equation (4.25) cannot be implemented in an on-line manner because unobservable future rewards are necessary. By introducing an additional memory variable, \( e \), associated with each state, the update can be sequentially carried out as follows:

\[
V_{(new)}^{\pi}(w_t) := V_{(odd)}^{\pi}(w_t) + \alpha \left[ r_{t+1} + \gamma V_{(odd)}^{\pi}(w_{t+1}) - V_{(odd)}^{\pi}(w_t) \right] e_{(new)}(w_t),
\]

(4.26)

where the variable \( e_{(new)} \) is updated according to

\[
e_{(new)}(w_t) := \gamma \lambda e_{(odd)}(w_t) + 1.
\]

(4.27)
4.2 Reinforcement Learning

It is referred to as the *eligibility trace* of the state. Note that the parameter $\lambda$ assigns an exponential discount on past states, which contrasts with the previous interpretation that it assigns an exponential discount on future temporal differences. These two different perspectives on TD$(\lambda)$ are essentially the same. For proof refer to Chapter 7 of Sutton and Barto (1998).

For the parametric representation the on-line TD$(\lambda)$ update rule is

$$
\theta_{t+1} := \theta_t + \alpha \left[ r_{t+1} + \gamma \hat{V}(w_{t+1}, \theta_t) - \hat{V}(w_t, \theta_t) \right] \sum_{k=1}^{t} \lambda^{t-k} \nabla_{\theta} \hat{V}(w_k, \theta_k).
$$

(4.28)

This update rule can be simplified computationally by calculating incrementally the summation term via the eligibility trace variable:

$$
e_t := \nabla_{\theta} \hat{V}(w_t, \theta_t) + (\gamma \lambda) e_{t-1}.
$$

(4.29)

The on-line TD$(\lambda)$ update rule using the eligibility trace is of form

$$
\theta_{t+1} := \theta_t + \alpha \left[ r_{t+1} + \gamma \hat{V}(w_{t+1}, \theta_t) - \hat{V}(w_t, \theta_t) \right] e_t.
$$

(4.30)

4.2.2 Control Tasks

The temporal-difference learning is a method of recursively estimating a value function. However our ultimate goal is to find a (suboptimal) policy. In the reinforcement learning framework we need to carry out simultaneously learning to predict (value function) and learning to act (policy). A general approach is that we compute an approximate value function and then select an action from the approximate value function via a one-step look-ahead search. In the dynamic programming context, when we have an optimal value function $V^*$, the following action-selection scheme guarantees its optimality (Section 2.2 of Ross (1983)):

$$
\pi^*(w_t) = \arg\max_{a_t} \sum_{w_{t+1}} \Pr(w_{t+1}|w_t, a_t) \left[ r(w_t, a_t, w_{t+1}) + \gamma V^*(w_{t+1}) \right].
$$

(4.31)

Since we only have an approximate value function, observed states and rewards, the optimality of an action cannot be guaranteed. Instead, we select an action “greedily” on the basis of the current approximate value function $\hat{V}$ and the observed reward $r_{t+1}$ as follows:

$$
\pi(w_t) = \arg\max_{a_t} \sum_{w_{t+1}} \Pr(w_{t+1}|w_t, a_t) \left[ r_{t+1} + \gamma \hat{V}(w_{t+1}) \right].
$$

(4.32)
4.2 Reinforcement Learning

The greedily selected action is unlikely to be a best action in the sense of maximising the ‘true’ value of the right-hand side of Equation (4.31). As a result, to ensure that we do not only ‘exploit’ the greedy action that is not best, we need to ‘explore’ other actions. This phenomenon is known as exploration/exploitation tradeoff. The $\epsilon$-policy is a widely-used simple method, in which we select a random action with probability $\epsilon$ and a greedy action at all other times. Other example of the action selection scheme is softmax action selection (Section 2.3 of Sutton and Barto (1998)).

4.2.3 Action-Value Function

As mentioned above, when selecting an action through the one-step look-ahead search, we need to know the transition probability function. We can estimate it with sequences of observed states and actions (model-based approach). It is also possible to choose an action via the one-step look-ahead search without building an approximate model of the environments by introducing the action-value function (model-free approach). The action-value function is defined by

$$Q(w_t, a_t) \triangleq \mathbb{E}_{w_{t+1}} [r(w_t, a_t, w_{t+1}) + \gamma V(w_{t+1})].$$  \hspace{1cm} (4.33)

It is often referred to as Q-function (Watkins, 1989). The action-value function can be learned by temporal-difference-learning-type algorithms (Chapter 6 of Sutton and Barto (1998)). If we use the action-value function, then the action selection problem becomes simple because a transition probability function is not required. In other words, we can choose a greedy action according to

$$\pi^*(w_t) = \arg\max_{a_t} Q^*(w_t, a_t).$$  \hspace{1cm} (4.34)
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

For a continuous state space we consider using a parametric function of features to approximate the value function as follows:

\[ V(w) \approx \hat{V}(w, \theta) = g(f(w), \theta), \]  

(4.35)

where \( f \) is a set of features, \( \{f_1, \ldots, f_K\} \) and \( \theta = \{\theta_1, \ldots, \theta_K\} \). In particular, we are interested in a linear approximation function \( g \); that is,

\[ \hat{V}(w, \theta) = \sum_{k=1}^{K} \theta_k f_k(w). \]  

(4.36)

In this section we discuss using a radial basis function model as a function approximator, in which each feature \( f \) is a Gaussian basis function.

4.3.1 Radial Basis Functions

Consider a linear basis function model of the form

\[ y = g(w, \theta) + n = \sum_{k=1}^{K} \theta_k \phi_k(w) + n, \]  

(4.37)

where \( w \) is a multi-dimensional input vector, \( y \) an output variable, and \( n \) is a Gaussian noise term. \( \phi_k(w) \) is a non-linear function of the input variable, which is referred to as a basis function. The most common form of the radial basis function is the Gaussian basis function, which has the form

\[ \phi_k(w; \mu_k, \Sigma_k) \propto \exp \left( -\frac{1}{2} (w - \mu_k)^T \Sigma_k^{-1} (w - \mu_k) \right), \]  

(4.38)

where \( \mu_k \) is the centre of the \( k \)th radial basis function; \( \Sigma_k \) is a covariance matrix that controls the smoothness and orientation of the basis function. This Gaussian basis function network has been proved to be capable of universal approximation (Hartman et al., 1990). Throughout this chapter, the terms ‘Gaussian basis function’ and ‘radial basis function’ will be interchangeably used.
Radial basis function models were initially developed for the purpose of exact interpolation (Chapter 5 of Bishop (1995)). However, it can be used as a smooth approximation method, in which the number of basis functions is set to be much smaller than the number of data points. We consider the following radial basis function model:

\[ y = \sum_{k=1}^{K} \theta_k \phi_k(w; \mu_k, \Sigma_k) + n \]

Note that training the radial basis function model involves two separate training procedures: (1) determining the number of basis functions, \( K \), and their parameters, \( \mu_k \) and \( \Sigma_k \); (2) determining the model parameter vector, \( \theta \). Unsupervised learning methods such as clustering algorithms (e.g. Chapter 9 of Bishop (2006)) provide solutions for the former problem; and the model parameters can be inferred using, for example, least squares estimation.

With the radial basis function model we can approximate a value function as follows:

\[ V(w_t) \approx \hat{V}(w_t, \theta) = \sum_{k=1}^{K} \theta_k \phi_k(w_t), \]

where each \( \phi_k \) is a Gaussian basis function. The approximate value function can be sequentially updated via the temporal-difference learning detailed in the previous section. Since we are interested in the model-free approach in which we do not need to build a transition probability function, we discuss a novel method for estimating the action-value function using the radial basis function model to deal with continuous state and/or action spaces.

### 4.3.2 Continuous State and Discrete Action Spaces

We consider a system with a multi-dimensional continuous state space and a single-dimensional discrete action space of small size. Discretisation is a widely used approach to the continuous state space problem. Tile coding is a discretisation method that partitions the continuous space into a number of tilings; each tiling consists of non-overlapping elements called tiles (Chapter 8 of Sutton and Barto (1998)). We discuss how we can use a radial basis function
model to estimate the action-value function $Q(w_t, a_t)$. Since the action-value function is a function of the state and the action, it is not straightforward to estimate it with a single radial basis function model solely of the state, in comparison with the case of estimating a value function shown in Equation (4.40). As a solution, we suggest a method in which we consider that each discrete action has its own value function and then the value function is approximated by a radial basis function model. For example, if there are two one-dimensional discrete actions, say $a_t = 0$ and $a_t = 1$, then we can write the action-value function $Q(w_t, a_t)$ as

$$Q(w_t, a_t) = \delta_{a_t, 0} \cdot A_0(w_t) + \delta_{a_t, 1} \cdot A_1(w_t),$$

(4.41)

where $\delta$ is an identity-indicator function defined by

$$\delta_{a_t, i} = \begin{cases} 0, & a_t \neq i \\ 1, & a_t = i. \end{cases}$$

(4.42)

Here $A_i$ is the value function of action $i$ defined by $A_i(w_t) = Q(w_t, i)$.

Instead of directly estimating $Q(w_t, a_t)$, we estimate the two value functions ($A_0$ and $A_1$) separately as follows:

$$\hat{Q}(w_t, a_t) = \delta_{a_t, 0} \cdot \hat{A}_0(w_t; \theta_0) + \delta_{a_t, 1} \cdot \hat{A}_1(w_t; \theta_1)$$

$$= \delta_{a_t, 0} \cdot \sum_{k=1}^{K} \theta_0^k \phi_k(w_t; \mu_k, \Sigma) + \delta_{a_t, 1} \cdot \sum_{k=1}^{K} \theta_1^k \phi_k(w_t; \mu_k, \Sigma),$$

(4.43)

where $\theta^0$ and $\theta^1$ denote the parameter vectors of $\hat{A}_0$ and $\hat{A}_1$ respectively. We update each of the parameter vectors according to the following TD($\lambda$) update rule:

$$\theta_i^{t+1} := \theta_i^t + \alpha \left[ r_{t+1} + \gamma \theta_i^T \phi(w_{t+1}) - \theta_i^T \phi(w_t) \right] e_t,$$

(4.44)

where $i = 0, 1$ and $\phi = [\phi_1, \ldots, \phi_K]^T$. The eligibility trace is recursively computed by

$$e_t = \phi(w_t) + (\gamma \lambda) e_{t-1}.$$

(4.45)

Once we have the approximate action-value function, we can readily select a greedy action by computing $\arg\max_{a_t} \hat{Q}(w_t, a_t)$. 

4.3 Decision Making on Continuous Spaces Using Radial Basis Functions
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

4.3.3 Continuous State and Action Spaces

When state and action spaces are both continuous, it is not possible to implement the above approach because there are an innumerable number of actions. We propose estimating the action-value function $Q(w_t, a_t)$ with a single radial basis function model after combining the state variable and the action variable into a new ‘augmented state’ variable. In other words, we form $s = [w^T, a^T]^T$. Hence, a value function of the new state variable $s_t$ is of form

$$Q(s_t) = Q\left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}\right).$$  \hspace{1cm} (4.46)

The notation ‘Q’ may not be appropriate, as $Q(s_t)$ is a value function of the state, but we retain it to emphasise that the augmented state variable involves the action. We can estimate this value function with the following radial basis function model:

$$\hat{Q}(s_t) = \hat{Q}\left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}, \theta\right) = \sum_{k=1}^{K} \theta_k \phi_k \left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}; \mu_k, \Sigma\right).$$  \hspace{1cm} (4.47)

The parameter vector is updated via the TD($\lambda$) update rule:

$$\theta_{t+1} := \theta_t + \alpha \left[r_{t+1} + \gamma \hat{Q}^T \phi(s_{t+1}) - \hat{Q}^T \phi(s_t)\right] e_t.$$  \hspace{1cm} (4.48)

The eligibility trace is computed by

$$e_t = \phi(s_t) + (\gamma \lambda) e_{t-1}.$$  \hspace{1cm} (4.49)

The small-sized discrete action problem can also be solved in this framework. For example, if $a_t = \{0, 1\}$, then $Q(s_t) = Q\left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}\right)$ is approximated by $\hat{Q}\left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}, \theta\right) = \sum_{k=1}^{K} \theta_k \phi_k \left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}; \mu_k, \Sigma\right)$. With the approximate value function, a greedy action is determined by selecting the larger of $\hat{Q}\left(\begin{bmatrix}w_t \\ 0\end{bmatrix}, \theta\right)$ and $\hat{Q}\left(\begin{bmatrix}w_t \\ 1\end{bmatrix}, \theta\right)$. For a continuous action space, however, the action selection process is not trivial, because we need to find a continuous action that maximises $\hat{Q}\left(\begin{bmatrix}w_t \\ a_t\end{bmatrix}, \theta\right)$. The most intuitive method for solving the difficulty is to discretise the continuous action space: dividing it into a number of discrete
actions, calculating the value of the approximate value function for each action, and choosing that which has the largest value. We employ this approach in an experiment presented in the next section. Alternatively, global optimisation techniques could be used to find a maximising continuous action. A novel global optimisation method using Gaussian processes (Osborne et al., 2008b) may be applicable, but this is not investigated here.

For a particular continuous system in which state and observation processes are linear and a reward function has a quadratic form, the optimal continuous action can be determined by solving the Riccati equation (e.g. Bertsekas (2005, 2007)). In the case of unknown system models, Bradtke (1993); Bradtke et al. (1994) showed that continuous actions could be selected recursively using the action-value function.

### 4.3.4 Results

As a problem having continuous state and discrete action spaces, we consider the mountain-car task in which we aim to drive an underpowered car up a steep mountain road (Figure 4.2). The car’s engine is not strong enough so that even at full power it cannot accelerate up the steep slope. The only solution is to use inertia; we drive backwards away from the goal and then exploit inertia along with the engine’s full power to propel the car up the slope. The car receives a reward of $-1$ on all time steps until it reaches the goal position. There are three discrete actions: full power forward (+1), full power reverse (−1), and zero power (0). The dynamics of the car’s movement is

\[
\begin{align*}
w_{t+1} &= \text{bound}[w_t + \dot{w}_t] \\
\dot{w}_{t+1} &= \text{bound}[\dot{w}_t + 0.001a_t - 0.0025\cos(3w_t)],
\end{align*}
\]

where $w_t$ and $\dot{w}_t$ denote the car’s position and velocity respectively, and $a_t$ represents the action. The scopes of the position and velocity are constrained to $-1.2 \leq w_{t+1} \leq 0.5$ and $-0.07 \leq \dot{w}_{t+1} \leq 0.07$, which is denoted by $\text{bound}[:]$ in Equation (4.50). When $w_{t+1}$ reaches the left bound, $\dot{w}_{t+1}$ is reset to zero. When it reaches the right bound, the goal is achieved.

We apply the two approximation methods using the radial basis function model: (1) estimating the action-value function with separate value functions for each action (Section
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

Figure 4.2: The mountain-car task.

4.3.2); and (2) estimating the action-value function with the augmented state variable (Section 4.3.3). For the sake of brevity we refer to them as ‘RBF-S’ and ‘RBF-A’ respectively. Their performances will be compared with that of the tile coding method, which is referred to as ‘Tile Coding’. For RBF-S and RBF-A we use five Gaussian basis functions on each of the position and velocity spaces. Their centres are chosen to be equally spaced over the spaces, and their widths are set to 0.1 (position) and 0.01 (velocity). These values are chosen not to generalise but also not to localise excessively (Figure 4.3). For RBF-A we use three Gaussian basis functions on the action space, centred at \{-1, 0, 1\} and having a common width of 0.00001. The very small value of the width is selected because it is unnecessary to generalise because we only consider three discrete actions that are valued −1, 0 or 1. For Tile Coding we use 4 (5 × 5) tilings. The hyper-parameters of the TD(λ) learning are set to \(\alpha = 0.1\), \(\gamma = 0.99\), and \(\lambda = 0.3\) because they produced the best performance and thus provide a more useful comparison for our novel approaches.

In the first experiment we set the initial values of the position and velocity to \(w_0 = [0, 0]^T\) for all episodes. The term ‘episode’ is interchangeably used with ‘trial’ in the literature. The MATLAB code for the Tile Coding algorithm employed in this section can be obtained at www.waxworksmath.com. We ran the three algorithms for 500 episodes; at the end of each episode we counted the number of steps taken for the cart to reach the goal position (Table 4.1). We can clearly see that RBF-S and RBF-A outperformed the discretisation-
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

Based method, Tile Coding; the policies found by RBF-S and RBF-A took 78 steps for the
car to arrive at the goal, whereas 80 steps for Tile Coding. Also, the radial-basis-function-
based algorithms converged to their own best policies much faster than Tile Coding; RBF-S
and RBF-A converged to their policies (78 steps) after 164 and 152 episodes respectively
and were close to optimal after only 10 episodes; Tile Coding converged to its policy (80
steps) after 450 episodes.

<table>
<thead>
<tr>
<th></th>
<th>Episode 1</th>
<th>Episode 10</th>
<th>Episode 50</th>
<th>Episode 100</th>
<th>Episode 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF-S</td>
<td>442</td>
<td>92</td>
<td>81</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>RBF-A</td>
<td>360</td>
<td>88</td>
<td>80</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>Tile Coding</td>
<td>2,831</td>
<td>477</td>
<td>208</td>
<td>84</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 4.1: The number of steps taken for the car to reach the goal position when the car’s
starting position and velocity are commonly fixed to 0 in every episode.

From the properties of the mountain-car task we know that the value of being near the
goal position and having positive velocity should be larger than that of being away from the
goal and having negative velocity. In addition, the worst state to be is the middle area of
the mountain with near zero velocity, because we need to drive the car backward up to the
mountain to gain speed generated by the inertia. The estimated value functions of RBF-S
and RBF-A appear to show these characteristics as the number of episodes increases (first
and second columns of Figure 4.4); the bottom-right corners of the figures have high values coloured dark red, whereas the middle regions have low values coloured dark blue. However, we cannot find such plausible characteristics in the estimated value function of Tile Coding (third column of Figure 4.4); the bottom left and top right corners, which are supposed to have low values, are coloured red. As a result it is expected that the policy derived from the estimated value function of Tile Coding is not as good as that of RBF-S and RBF-A.

The second experiment deals with the case in which in the beginning of each episode the car’s position and velocity are not fixed to zero but are randomly selected. It allows the car to explore the state space more extensively. We carried out 500 episodes, each having randomly selected starting position and velocity. Compared with RBF-S and RBF-A, Tile Coding took more steps to reach the goal, occasionally near 1,000 steps (top row of Figure 4.5). The estimated value function of Tile Coding after 500 trials is less optimal as it has high values in the bottom left corner where the car’s position is far away from the goal (middle row of Figure 4.5). Meanwhile, the estimated value functions of RBF-S and RBF-A reflect well the characteristics of the mountain-car task. The bottom row of Figure 4.5 illustrates policy functions obtained from the estimated value functions. The policy functions of RBF-S and RBF-A appear to be correctly estimated: when the car is located on the bottom of the valley and has non-positive velocity, a correct action would be to take full power reverse (blue); and in other states to take full power forward (red). However, the policy function of Tile Coding shows no consistent action plan: the three actions (full power forward, full power reverse, zero power (yellow)) are spread over the state space with no clear patterns.

We have so far considered the case of three discrete actions, i.e. \{-1, 0, 1\}. Suppose that we have a continuous action space ranging from -1 to 1. An advantage of the method of augmenting the state variable is that we can estimate an action-value function of continuous states and actions with a radial basis function model. We used five radial basis functions with equally spaced centres and the common width of 0.3 over the continuous action space.

What remains difficult is the action selection. Given the values of car’s position and velocity, we need to find a greedy action that maximises \( \hat{Q}(\left[ \begin{array}{c} w_t \\ a_t \\ \theta \end{array} \right], \theta) \). To find a greedy
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

Figure 4.4: Value functions ($\max_{a_t} Q(w_t, a_t)$) estimated after 1 (first row), 10 (second row), 100 (third row) and 1,000 (fourth row) episodes for RBF-S (left column), RBF-A (middle column) and Tile Coding (right column). The estimated value functions of RBF-S and RBF-A rightly reflect the characteristics of the mountain-car task: they show high values (dark red) for high position and high velocity, and low values (dark blue) for middle position and low velocity. Meanwhile, the estimated value function of Tile Coding does not have these characteristics.
4.3 Decision Making on Continuous Spaces Using Radial Basis Functions

Figure 4.5: (Top row) The number of steps taken to the goal over 500 episodes. (Middle row) Value functions estimated after 500 episodes. (Bottom row) Policy functions: full power forward (red), full power reverse (blue), zero power (yellow). The policy function of Tile Coding does not reflect the characteristics of the mountain-car problem.
action for a given state we divided the continuous action space into 21 equally space discrete actions, and then computed a maximising action. This is a time-consuming task if the space is densely divided. We refer to this approach as ‘cRBF-A’ to distinguish it from the discrete action method, RBF-A. For the case of fixed starting state, i.e. \([w_0, \dot{w}_0] = [0, 0]\), cRBF-A found a better policy after 500 episodes, in which the car reached the goal position in 73 steps (Table 4.2). For the case of a random starting state, the value functions and policy functions estimated by RBF-A and cRBF-A appear to be slightly different. The policy function of RBF-A has a wider area for taking full power reverse (blue) than that of cRBF-A (Figure 4.6). The average step values of RBF-A and cRBF-A over the 500 episodes were 68.17 and 67.78 respectively. This indicates that the policy found when the action space is assumed to be continuous is slightly closer to optimal than when it is assumed to be discrete.

<table>
<thead>
<tr>
<th></th>
<th>Episode 1</th>
<th>Episode 10</th>
<th>Episode 50</th>
<th>Episode 100</th>
<th>Episode 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF-A</td>
<td>360</td>
<td>88</td>
<td>80</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>cRBF-A</td>
<td>197</td>
<td>86</td>
<td>80</td>
<td>79</td>
<td>73</td>
</tr>
</tbody>
</table>

Table 4.2: The number of steps taken for the car to reach the goal position when the car’s starting position and velocity are commonly fixed to 0 in every episode. When the action space is assumed to be continuous (cRBF-A), we found a better policy that took the car to the goal in 73 steps.

In this section we suggested a simple method for handling continuous spaces by using Gaussian basis networks and combining state and action spaces. Our experiment results have shown that this approach is more suitable than discretisation-based-approaches, such as Tile Coding that has been widely used in the reinforcement learning literature.

4.4 Finding a Path in Non-stationary Environments

Reinforcement learning is a trial-and-error based method. In other words, it learns a model of an environment by exploring the environment and receiving feedback. This approach is suitable for a static environment because it is possible to learn it with many trials. In this section we discuss the case of a non-stationary environment and suggest a method of dealing
Figure 4.6: Value functions (top row) and policy functions (bottom row) obtained after 500 episodes for RBF-A (left column) and cRBF-A (right column) when the car’s starting position and velocity are randomly selected in every episode.
4.4 Finding a Path in Non-stationary Environments

with it in the reinforcement learning framework.

Consider a situation in which we try to find the shortest path from a starting position to a goal. There are three hidden posts that we must discover before reaching the goal (Figure 4.7). The state space is represented by a $10 \times 10$ grid and there are four discrete actions (up, down, left, right). At each time step we are punished by receiving a reward of $-1$ until arriving at the goal, which is the end of an episode. The three posts coloured grey return a reward of $+10$ when they are discovered for the first time, and have no values thereafter. In addition, we receive a reward of $+10$ at the goal.

![Figure 4.7: A shortest path problem. We are interested in finding a route from 'S' to 'G' after discovering three hidden areas coloured grey.](image)

A standard approach to solving this problem would be to store the values of state-action pairs in a lookup table of form $Q(n_t, a_t)$, where the variable $n_t = \{1, 2, \ldots, 100\}$ denotes the state and $a_t = \{1, 2, 3, 4\}$ the action. If we have an observation sequence of successive states, action and reward at time $t$, i.e. $\{n_t, a_t, n_{t+1}, r_{t+1}\}$, then the value of an action-value pair is updated via the TD($\lambda$) update rule:

$$Q(n_t, a_t) := Q(n_t, a_t) + \alpha [r_{t+1} + \gamma Q(n_{t+1}, a_{t+1}) - Q(n_t, a_t)] e(n_t, a_t), \quad (4.51)$$
for all \( n_t \) and \( a_t \). The eligibility trace, \( e(n_t, a_t) \), is computed by

\[
e(n_t, a_t) := \begin{cases} 1 + \gamma \lambda e(n_t, a_t) & \text{for observed } n_t, a_t \\ \gamma \lambda e(n_t, a_t) & \text{otherwise} \end{cases} \tag{4.52}
\]

An action is \( \epsilon \)-greedily determined via \( \arg\max_{a_t} Q(n_t, a_t) \) with probability \((1 - \epsilon)\): with probability \( \epsilon \), an action is randomly selected to ensure exploration. Parameters are set to \( \alpha = 0.1, \gamma = 0.99, \lambda = 0.1 \) and \( \epsilon = (0.9)^{0.9999t} \). Note that \( \epsilon \) decreases over time in order to exploit what has been learned after a large number of trials. We carried out this standard learning algorithm for 2,000 times. A path found by the standard algorithm was disastrously far away from an optimal one (left panel of Figure 4.8). The path took 2,731 steps from the start to the goal and the total reward was \(-2,690\). This failure results from the fact that the environment of the problem is not static. For example, once a hidden area has been discovered, being next to it is not as beneficial as when it was undiscovered, because re-visiting the area returns no rewards. We can conclude that the ordinary action-value function consisting only of states and actions is not suitable for capturing the non-stationary characteristic of the problem.

We discuss a method of tackling this challenge. The value of being in a particular state and taking an action changes according to whether or not any of the three unknown posts have been discovered. Since there are three posts to find, we need to consider eight different settings of the environment: \( 2 \times 2 \times 2 = 8 \). We introduce an extra variable \( s_t = \{1, 2, \ldots, 8\} \) that represents status of the posts. Instead of the ordinary action-value function \( Q(n_t, a_t) \), we use a modified action-value function of form \( Q(n_t, a_t, s_t) \). With an observation sequences \( \{n_t, a_t, n_{t+1}, s_{t+1}, r_{t+1}\} \), we update the new action-value function via the following TD(\( \lambda \)) update rule:

\[
Q(n_t, a_t, s_t) := Q(n_t, a_t, s_t) + \alpha (r_{t+1} + \gamma Q(n_{t+1}, a_{t+1}, s_{t+1}) - Q(n_t, a_t, s_t))e(n_t, a_t, s_t),
\]

for all \( n_t, a_t \) and \( s_t \). Eligibility trace is computed by

\[
e(n_t, a_t, s_t) := \begin{cases} 1 + \gamma \lambda e(n_t, a_t, s_t) & \text{for observed } n_t, a_t, s_t \\ \gamma \lambda e(n_t, a_t, s_t) & \text{otherwise} \end{cases} \tag{4.54}
\]

Parameters are set to the same values as the above experiment.
This novel approach successfully found the shortest path taking 25 steps to the goal and having the total reward of 16 (right panel of Figure 4.8). The algorithm converged to the policy after 1,000 episodes. This result tells us that it is possible to deal with non-stationary environments in the reinforcement learning framework by considering an extra variable for the non-stationarity. If we combine this approach with the method of using a radial basis function model detailed in Section 4.3, we may be able to deal with the problem of non-stationary continuous spaces. This interesting extension requires further research but significantly increases the computational burden.

Figure 4.8: Paths found after 2,000 episodes by the naive approach (left) and the novel approach (right). Since the novel approach consider the change of the environment, it successfully found the shortest path.

In this chapter we have discussed reinforcement learning that aims to find an optimal policy in a Markov decision processes. Traditional reinforcement learning methods are designed to deal with discrete and stationary problems. We have proposed techniques that allow us to handle continuous or non-stationary environments, which can, in principle, be extended to continuous non-stationary problems. Experimental results have shown that these novel approaches are more suitable for such environments than the traditional algorithms. All
the methods considered in this chapter, however, make the tacit assumption that information from a sensor, for example, is fully observed. In the next chapter we will examine the dynamic forecasting problem in which information comes from multiple sources and the information is partially unobservable.
In the previous chapter we presented an on-line learning problem of finding an optimal policy in a Markov decision process. We assumed that the information available is a series of states, actions and rewards, all of which are completely known and/or observed. A stationary environment returns this information repeatedly and we may find approaches which converge to the optimal policy. However, in many environments, we are concerned with dynamically forecasting future values of multiple sensors, actions and states, in particular when the observations are partially or entirely missing. In this chapter we consider multivariate time series analysis in such incomplete environments. We adopt the dynamic linear model framework detailed in Chapter 2 to model the time series processes. In comparison to traditional statistical time series analysis (e.g. Box et al. (1970); Chatfield (2004)), an important feature of the dynamic linear model is that it can cope easily with incomplete and/or missing observations (Harvey, 1990; Durbin and Koopman, 2001). We discuss three multivariate time series forecasting models: two of them are grounded in the dynamic linear model framework, namely a dynamic multivariate autoregressive model and a multivariate local trend model; the third is a Gaussian process model. To make comparison of the model performances, we analyse two weather condition data sets.
5.1 Learning Dynamic Linear Models

It is generally assumed that components of a dynamic linear models (i.e. observation and state evolution matrices and noise covariances) are known to a data-modeller. However, it is common that part of, or the whole of, the components is not available. This makes it necessary to estimate simultaneously the state variable and the unknown components (or parameters) of the dynamic linear model. In this section we briefly introduce the expectation-maximisation algorithm (Dempster et al., 1977), which is a method of estimating latent variables. In addition, we discuss how the expectation-maximisation algorithm is applied to learning a dynamic linear model, which was first proposed by Shumway and Stoffer (1982).

5.1.1 The Expectation-Maximisation Algorithm

The expectation-maximisation algorithm is an iterative method of finding maximum likelihood estimates of latent variables or unknown parameters when data available is incomplete. We first consider a standard maximum likelihood estimation method. Suppose that we have a data set \( Y = \{ y_1, \ldots, y_T \} \) drawn from a probability density function \( p(y; \Theta) \), which is governed by a set of parameters, \( \Theta \). Assuming that each observation \( y_i \) is independent and identically distributed, the joint probability density is represented by

\[
p(Y; \Theta) = \prod_{i=1}^{T} p(y_i; \Theta).
\]  

(5.1)

This can be viewed as a function of the parameter set given the data set, and it is referred to as a likelihood function denoted by \( \mathcal{L}(\Theta|Y) \). The maximum likelihood estimate of \( \Theta \) is one that maximises the likelihood function; that is,

\[
\hat{\Theta} = \arg\max_{\Theta} \mathcal{L}(\Theta|Y).
\]  

(5.2)

If \( p(y; \Theta) \) is a multivariate Gaussian with \( \Theta = \{ \mu, \Sigma \} \), then we can readily compute the maximum likelihood estimate of \( \Theta \) by setting to zero the derivative of the logarithm of the likelihood function and solving the resultant equation with respect to \( \mu \) and \( \Sigma \).
We now suppose to have a latent variable \( w \) in addition to the observable variable \( y \). Hence, a complete data set is \( Z = \{ Y, W \} \). We wish to find an estimate of \( \Theta \) that maximises the following likelihood function

\[
L(\Theta|Z) = p(Z; \Theta) = p(Y, W; \Theta).
\] (5.3)

However, as \( w \) is a latent variable, it is not so straightforward to find the maximum likelihood estimate of \( \Theta \). In other words, we are not given the complete data set \( Z \), but only the observable part \( Y \). The basic concept of the expectation-maximisation algorithm is elegantly simple. Since we cannot directly evaluate \( p(Y, Z; \Theta) \), we compute its expectation over the unobserved variable \( w \) (expectation step), and then we obtain the maximum likelihood estimate of \( \Theta \) by maximising the expectation (maximisation step).

The probability density of the complete set can be rewritten as

\[
p(Z; \Theta) = p(Y, W; \Theta) = p(W|Y; \Theta)p(Y; \Theta).
\] (5.4)

In other words, we know the values of the latent variables in \( W \) through the conditional density of \( W \) given \( Y \), \( p(W|Y; \Theta) \). In the expectation step, assuming that the observation set \( \{y_1, \ldots, y_T\} \) and an estimate of the parameter set \( \hat{\Theta} \) are given, we take expectations over \( p(Y, W; \Theta) \) with regard to the latent variables \( W \) as follows:

\[
E_W[p(Y, W; \Theta) \mid Y, \hat{\Theta}] = \int p(Y, W; \Theta)p(W|Y, \hat{\Theta})dW.
\] (5.5)

In the maximisation step we maximise the above expectation term with respect to \( \Theta \). Hence, the given estimate \( \hat{\Theta} \) is updated according to

\[
\hat{\Theta}_{(new)} := \arg\max_{\Theta} E_W[p(Y, W; \Theta) \mid Y, \hat{\Theta}_{(old)}].
\] (5.6)

The expectation and maximisation steps alternate until the value of the log-likelihood converges. It is known that the log-likelihood is guaranteed to increase at each iteration and thus the algorithm is guaranteed to reach a local maximum of the likelihood function (Dempster et al., 1977).
5.1 Learning Dynamic Linear Models

5.1.2 Estimating Parameters of Dynamic Linear Models

Shumway and Stoffer (1982) suggested a method of learning unknown components of a dynamic linear model using the expectation-maximisation algorithm. This subsection summarises their approach. Suppose that we have the following dynamic linear model:

\[
y_t = H_t w_t + n_t, \quad (5.7)
\]
\[
w_t = F w_{t-1} + v_t, \quad (5.8)
\]

where \( n_t \sim N(0, R) \) and \( v_t \sim N(0, Q) \). Note that time index \( t \) is not present on \( F, R \) and \( Q \) because they are assumed to be static, not because of brevity of notation. In the dynamic linear model we assume that the observation matrix, \( H_t \), is known, but the state evolution matrix, \( F \), and the state and observation noise covariances, \( Q \) and \( R \), are not given. In addition, we assume that the initial state \( w_0 \) is Gaussian distributed with mean \( \mu_0 \) and covariance \( \Sigma_0 \), which are also unknown components of the model. If we regard the unknown components of the model as a set of parameters, i.e. \( \Theta = \{F, R, Q, \mu_0, \Sigma_0\} \), and the state variable \( w \) as a latent variable, the we can use the expectation-maximisation algorithm to compute the maximum likelihood estimate of \( \Theta \) with a set of observations, \( D_T = \{y_1, \ldots, y_T\} \), given.

Since the state and observation variables are all Gaussian distributed, the likelihood function of the complete data is of form

\[
L(\Theta | w_0, \{y_i\}_{i=1}^T, \{w_i\}_{i=1}^T) =
\]
\[
\frac{1}{(2\pi)^{\alpha/2} |\Sigma_0|^{1/2}} \text{exp} \left( -\frac{1}{2} (w_0 - \mu_0)^T \Sigma_0^{-1} (w_0 - \mu_0) \right) \times \frac{1}{(2\pi)^{\alpha T/2} |Q|^{1/2}} \text{exp} \left( -\frac{1}{2} \sum_{t=1}^T (w_t - F w_{t-1})^T Q^{-1} (w_t - F w_{t-1}) \right) \times \frac{1}{(2\pi)^{\beta T/2} |R|^{1/2}} \text{exp} \left( -\frac{1}{2} \sum_{t=1}^T (y_t - H_t w_t)^T R^{-1} (y_t - H_t w_t) \right). \quad (5.9)
\]

Here \( \alpha \) and \( \beta \) denote the dimension of \( w \) and \( y \) respectively. Ignoring constants, hence, the
5.1 Learning Dynamic Linear Models

The log likelihood function of the complete data is

\[
\log L = - \frac{1}{2} \log |\Sigma_0| - \frac{1}{2} (w_0 - \mu_0)^T \Sigma^{-1} (w_0 - \mu_0)
\]

\[
- \frac{T}{2} \log |Q| - \frac{1}{2} \sum_{t=1}^{T} (w_t - Fw_{t-1})^T Q^{-1} (w_t - Fw_{t-1})
\]

\[
- \frac{T}{2} \log |R| - \frac{1}{2} \sum_{t=1}^{T} (y_t - H_t w_t)^T R^{-1} (y_t - H_t w_t)
\]  \hspace{1cm} (5.10)

In the expectation step we compute the conditional expectation of the log likelihood with regard to the latent states, given \(D_T\) and the \((j-1)\)th iterate estimate of the parameter set denoted by \(\hat{\Theta}^{(j-1)}\). We can compute this conditional expectation (Shumway and Stoffer, 1982):

\[
E[\log L | D_T, \hat{\Theta}^{(j-1)}] =
\]

\[
- \frac{1}{2} \log |\Sigma_0^{(j-1)}| - \frac{1}{2} \text{tr} \left\{ \left( \Sigma_0^{(j-1)} \right)^{-1} \left( P_{0|T} + (\hat{w}_0|T - \mu_0^{(j-1)})(\hat{w}_0|T - \mu_0^{(j-1)})^T \right) \right\}
\]

\[
- \frac{T}{2} \log |Q| - \frac{1}{2} \text{tr} \left\{ Q^{-1} (C - BF^T - FB^T + FA^T) \right\}
\]

\[
- \frac{T}{2} \log |R| - \frac{1}{2} \text{tr} \left\{ R^{-1} \sum_{t=1}^{T} \left( (y_t - H_t \hat{w}_t|T)(y_t - H_t \hat{w}_t|T)^T + H_t P_{t|T} H_t^T \right) \right\}.
\]  \hspace{1cm} (5.11)

Here \(\text{tr}\) denotes the trace of a matrix and

\[
A = \sum_{t=1}^{T} \left( P_{t-1|T} + \hat{w}_{t-1|T} \hat{w}_{t-1|T}^T \right), \hspace{1cm} (5.12)
\]

\[
B = \sum_{t=1}^{T} \left( P_{t,t-1|T} + \hat{w}_{t|T} \hat{w}_{t-1|T}^T \right), \hspace{1cm} (5.13)
\]

\[
C = \sum_{t=1}^{T} \left( P_{t|T} + \hat{w}_{t|T} \hat{w}_{t|T}^T \right). \hspace{1cm} (5.14)
\]

The conditional mean and covariance terms (i.e. \(\hat{w}_{t|T}, P_{t|T}\) and \(P_{t,t-1|T}\)) can be computed by the Kalman-Rauch smoother detailed in Section 2.3.1. In the maximisation step the parameter set is re-estimated by setting the corresponding partial derivative of the expected log likelihood to zero and solving it with respect to each element of the parameter set. The \((j)\)th
iterate estimates for the parameters are given by

\[
\hat{F}(j) = BA^{-1}, \quad (5.15)
\]

\[
\hat{Q}(j) = \frac{1}{T}(C - BA^{-1}B^T), \quad (5.16)
\]

\[
\hat{R}(j) = \frac{1}{T} \sum_{t=1}^{T} [(y_t - H_t\hat{w}_t|T)(y_t - H_t\hat{w}_t|T)^T + H_tP_t|T H_t^T], \quad (5.17)
\]

\[
\hat{\mu}(j) = \hat{w}_0|T, \quad (5.18)
\]

\[
\hat{\Sigma}(j) = P_0|T - \hat{w}_0|T\hat{w}_0^T|T. \quad (5.19)
\]

We iterate the estimation and maximisation steps until the value of the log likelihood function converges.

Ghahramani and Hinton (1996) proposed a generalised algorithm in which the observation matrix \( H \) is also regarded as a parameter. In addition, for a non-linear dynamic model Roweis and Ghahramani (2001) suggested a method that employs extended Kalman smoothing to estimate parameters of the model. For time-varying parameters, it is possible to provide estimate using observations over a time window.

### 5.2 Multivariate Time Series Models

A need for dealing with multivariate time series data is common in many problem domains. For example, we can consider a network of sensors measuring weather conditions, or multiple financial indices reflecting global and domestic economic states. Since such data are essentially non-stationary, correlated and often have periodic patterns, it is a hard challenge to analyse and forecast a multivariate time series. Moreover, such a time series can be occasionally incomplete: some sensor readings or financial instruments may not be available. In this section we discuss forecasting models that deal with an incomplete multivariate time series.
5.2 Multivariate Time Series Models

5.2.1 Dynamic Multivariate Autoregressive Model

An autoregressive (AR) model is one of the most widely used time series models. It is a type of multiple linear regression model in which a dependent variable is regressed not on independent variables but on past values of the dependent variable. For a scalar observation variable \( y \) an autoregressive model of order \( p \) is defined by

\[
y_t = w_0 + w_1 y_{t-1} + \cdots + w_p y_{t-p} + n_t, \tag{5.20}
\]

where \( \{w_0, w_1, \ldots, w_p\} \) are parameters of the model and \( n_t \) is a Gaussian noise term with zero mean and finite variance. This model is abbreviated to AR(\( p \)). Here the order \( p \) determines the number of past time steps required to predict the current value.

If there are multiple observation variables and they interrelate, we can consider a multivariate autoregressive (MAR) model. Suppose that there are \( M \) different sensors, each of which has a time series. These sensors are correlated in a way that the value of a sensor at \( t \) is related to the \( p \) previous values of all other sensors including its own past values. We can represent each sensor’s value at \( t \) as follows:

\[
y_t^{(1)} = w_0^{(1)} + w_{11}^{(1)} y_{t-1} + \cdots + w_{D1}^{(1)} y_{t-1} + \cdots + w_{1p}^{(1)} y_{t-p} + \cdots + w_{Dp}^{(1)} y_{t-p} + n_t^{(1)},
\]

\[
\vdots
\]

\[
y_t^{(M)} = w_0^{(M)} + w_{11}^{(M)} y_{t-1} + \cdots + w_{D1}^{(M)} y_{t-1} + \cdots + w_{1p}^{(M)} y_{t-p} + \cdots + w_{Dp}^{(M)} y_{t-p} + n_t^{(M)}. \tag{5.21}
\]

These equations can be rewritten as a single equation: a MAR(\( p \)) model is defined by

\[
y_t = w_0 + W_1 y_{t-1} + \cdots + W_p y_{t-p} + n_t, \tag{5.22}
\]

where \( y_t = [y_t^{(1)} \ldots y_t^{(M)}]^T \), \( w_0 = [w_0^{(1)} \ldots w_0^{(M)}]^T \), \( W_i = \begin{bmatrix} w_{1i}^{(1)} & \cdots & w_{Di}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{1i}^{(M)} & \cdots & w_{Di}^{(M)} \end{bmatrix} \) and \( n_t = [n_t^{(1)} \ldots n_t^{(M)}]^T \).

For the MAR(\( p \)) model to deal with a non-stationary property of a time series we consider a dynamic version of the model by allowing the parameters (i.e. \( w_0 \) and \( W_i \)) to vary over time. Re-arranging Equation (5.22) we have

\[
y_t = H_t w + n_t, \tag{5.23}
\]
where \( w = [w^{(1)}_0 \ldots w^{(1)}_{D_1} \ldots w^{(1)}_1 \ldots w^{(M)}_0 \ldots w^{(M)}_{D_1} \ldots w^{(M)}_1 \ldots w^{(M)}_{D_p}]^T \). The observation matrix \( H_t \) consists of \( p \) previous values of the observation variable \( y_t \):

\[
H_t = [1, y_{t-1}^T, \ldots, y_{t-p}^T] \otimes I_M.
\]

Here \( I_M \) is the identity matrix of dimension \( M \), and \( \otimes \) denotes the Kronecker product. We assume that the state variable evolves according to a random walk. Therefore, a dynamic multivariate autoregressive (DMAR) model of order \( p \) is defined by:

\[
y_t = H_t w_t + n_t,
\]

\[
w_t = w_{t-1} + v_t,
\]

where \( n_t \sim N(0, R_t = r_t I) \) and \( v_t \sim N(0, Q_t = q_t I) \).

We estimate the unknown noise variances, \( r_t \) and \( q_t \), adaptively by covariance-matching techniques (Jazwinski, 1970; Mehra, 1972). The key idea underlying the technique is to use a residual defined by:

\[
e_{t|t-1} = y_t - \hat{y}_{t|t-1}.
\]

The covariance matrix of the residual error is:

\[
S_{t-1} \triangleq E[(y_t - \hat{y}_{t|t-1})(y_t - \hat{y}_{t|t-1})^T] = r_t I + H_t P_{t|t-1} H_t^T + q_t H_t H_t^T.
\]

This covariance can be approximated as \( S_{t-1} \approx e_{t|t-1} e_{t|t-1}^T \), which we denote by \( C_{t-1} \). If \( q_t \) is assumed to be zero, \( S_{t-1} \) reduces to \( r_t I + H_t P_{t|t-1} H_t^T \), which is denoted by \( S_{t-1}^{q=0} \). We can, hence, argue that if \( C_{t-1} \leq S_{t-1}^{q=0} \), then \( \hat{q}_t = 0 \). Otherwise,

\[
\hat{q}_t = \frac{C_{t-1} - S_{t-1}^{q=0}}{H_t H_t^T}.
\]

We smooth the estimate, \( \hat{q}_t \), with its one-step previous estimate, assuming that \( q_t \) change smoothly:

\[
\hat{q}_t := (1 - \alpha)\hat{q}_{t-1} + \alpha \hat{q}_t,
\]
where \(0 < \alpha < 1\) is a smoothing parameter.

From Equation (5.28) we can see that if \(r_t\) is assumed to be zero, then \(S_{t-1}\) is simply \(H_t P_{t|t-1} H_t^T\). As with the estimation for \(q_t\), this value is compared with \(C_{t-1}^t\); if \(C_{t-1}^t \leq H_t P_{t|t-1} H_t^T\), then \(\hat{r}_t = 0\). Otherwise,

\[
\hat{r}_t = C_{t-1}^t - H_t P_{t|t-1} H_t^T. 
\] (5.31)

The smoothing scheme is also applied to this case:

\[
\hat{r}_t := (1 - \beta) \hat{r}_{t-1} + \beta \hat{r}_t, 
\] (5.32)

where \(0 < \beta < 1\) is another smoothing parameter. Note that we smooth the estimates of \(q\) and \(r\) because they are based on only one residual, which implies that the estimates have little statistical significance. Hence, a small value for \(\alpha\) or \(\beta\) is undesirable in order to ensure a smooth change of the estimates. We used \(\alpha = 0.9\) and \(\beta = 0.7\) in experiments conducted in the next section.

We can consider an advanced model in which the state variable varies according to unknown dynamics, not according to a random walk; that is, it is of form

\[
y_t = H_t w_t + n_t, 
\] (5.33)

\[
w_t = F w_{t-1} + v_t, 
\] (5.34)

where the unknown matrix \(F\) governs the movement of the state variable. In addition to the noise covariances we could estimate this matrix via the expectation-maximisation algorithm detailed in Section 5.1.2. However, it should be noted that the dimension of \(w_t\) is \((1 + pM)M \times 1\), and accordingly, that of \(F\) is \((1+pM)M \times (1+pM)M\). If the AR order, \(p\), and/or the dimension of \(y_t\), \(M\), are large, the number of parameters to estimate is innumerably many. For this reason it would be unrealistic to use this approach in practice.
5.2 Multivariate Time Series Models

5.2.2 Multivariate Local Trend Model

As a multivariate time series model we consider the following simple dynamic linear model

\[ y_t = w_t + n_t, \quad (5.35) \]
\[ w_t = F_t w_{t-1} + v_t, \quad (5.36) \]

where \( n_t \sim N(0, R_t) \) and \( v_t \sim N(0, Q_t) \). A major difference from the DMAR model in Equations (5.25) and (5.26) depends on how we treat the state variable \( w_t \). In the DMAR model \( w_t \) represents a time-varying parameter vector of a multivariate autoregressive model. Meanwhile, in the above model \( w_t \) is treated as unknown true values of observations. We assume that the true values of \( M \) sensors at \( t \) are related to the true values at \( t - 1 \) as follows:

\[
\begin{bmatrix}
  w_{t}^{(1)} \\
  \vdots \\
  w_{t}^{(M)}
\end{bmatrix} =
\begin{bmatrix}
  f_{11} & \cdots & f_{1M} \\
  \vdots & \ddots & \vdots \\
  f_{M1} & \cdots & f_{MM}
\end{bmatrix}
\begin{bmatrix}
  w_{t-1}^{(1)} \\
  \vdots \\
  w_{t-1}^{(M)}
\end{bmatrix}
+ 
\begin{bmatrix}
  v_{t}^{(1)} \\
  \vdots \\
  v_{t}^{(M)}
\end{bmatrix},
\]

\[ = F_t w_{t-1} + v_t. \quad (5.37) \]

In this model we think of \( y_t \) as a noisy observation vector of the sensors’ true values.

The diagonal elements of the state evolution matrix, \( F_t \), depicts unknown local trends of the sensors; the off-diagonal elements represent the relationships between the sensors. We refer to this model as a multivariate local trend (MLT) model. We can estimate the unknown components of the model, i.e. \( F_t \), \( Q_t \) and \( R_t \), using the expectation-maximisation algorithm. Note that this MLT model is similar to the DMAR(\( p = 1 \)) model in that they are both based on a first-order Markov model. However, they differ in how the first-order Markov model is treated. For the the DMAR(\( p = 1 \)) model it is represented by a state equation, \( y_t = H_t w_t + n_t \), whereas for the MLT model by an observation equation, \( w_t = F_t w_{t-1} + v_t \). In other words, the DMAR model’s forecast is a linear function of a past observation vector (i.e. \( H_t = [1, y_{t-1}^T] \otimes I_M \)). However, that of the MLT model is a linear function of a latent variable (i.e. \( w_{t-1} \)), which implies that it is noise-filtered and smoother.
5.2 Multivariate Time Series Models

5.2.3 Gaussian Process Model

Gaussian processes (Rasmussen and Williams, 2006) represent a method to perform Bayesian inference about functions. A Gaussian process (GP) is defined as being a probability distribution over a (possibly infinite) number of variables, such that the distribution over any finite subset of them is a multivariate Gaussian.

We consider a function \( y(t) \). We are principally concerned with prediction and hence functions in time. For a single sensor case we use \( y = \{ y_1, y_2, \ldots, y_n \} \) to refer to a possible vector of function outputs and \( t = \{ t_1, t_2, \ldots, t_n \} \), function inputs. We define a GP as the following multivariate Gaussian distribution

\[
p(y|t) = N(\mu, C). \tag{5.38}
\]

Here \( C \) is a square matrix indicating the strength of correlations amongst the entries of \( y \). It is generated by a covariance function, \( c(t, t') \):

\[
C = \begin{bmatrix}
c(t_1, t_1) & c(t_1, t_2) & \cdots & c(t_1, t_n) \\
c(t_2, t_1) & c(t_2, t_2) & \cdots & c(t_2, t_n) \\
\vdots & \vdots & \ddots & \vdots \\
c(t_n, t_1) & c(t_n, t_2) & \cdots & c(t_n, t_n)
\end{bmatrix}. \tag{5.39}
\]

Almost all functions of interest, we expect, possess some degree of smoothness. That is, the value of a function at \( t \) is strongly correlated with the values close to \( t \), these correlations becoming weaker further away. A prototypical choice is the squared exponential

\[
c(t, t') = h^2 \exp \left( -\frac{1}{2} \frac{|t - t'|^2}{m} \right). \tag{5.40}
\]

Here \( h > 0 \) and \( m > 0 \) specify the expected length scales of the function in output and input spaces respectively.

Consider knowing the predictor data \( y \) and \( t \) and being interested in the value of the predictant \( y_s \) at known \( t_s \). Assuming \( \mu = 0 \) we can write a conditional distribution of \( y_s \) as follows:

\[
p(y_s|t_s, y, t) = N(C_s C_s^{-1} y, c(t_s, t_s) - C_s C_s^{-1} C_s^T), \tag{5.41}
\]
5.3 Results

where \( C_s = [c(t_s, t_1) \quad c(t_s, t_2) \quad \cdots \quad c(t_s, t_n)] \). Our best estimate for \( y_s \) is the mean of this distribution and the uncertainty in our estimate is captured in its variance. Osborne and Roberts (2007) presented an on-line formulation of a multi-dimensional GP that allows us to model the correlations between sensor readings, and to update this model on-line as new observations are sequentially available. Their idea is based on iteratively computing the inverse of the matrix \( C \) in Equation (5.41): when we receive new data \( t_s \), \( C \) is changed only in the addition of a new row and a new column, and thus a matrix inversion computed at the last iteration can with care be recycled.

5.3 Results

In this section we apply the multivariate time series models detailed in the previous section to real-world data. We are particularly interested in the case in which sensor readings are partially or entirely missing. We examine two data sets: (1) annual surface temperature series (single sensor problem) and (2) air temperature series measured at three locations (multiple sensor problem). To obtain the GP results given in this section we used a set of MATLAB code written by Osborne and Roberts (2007).

5.3.1 Single Sensor Case

We consider a global temperature series record from 1880 to 2007 (Figure 5.1). The data is a combination of air-sea temperature anomalies; the temperature anomalies are computed relative to the base period 1951 ~ 1980. For further information about the data refer to http://data.giss.nasa.gov/gistemp/. It appears that there exists an overall upward trend in the series with a level-off for 1940 ~ 1980 followed by a rapid upward trend. It is arguable whether the overall trend is natural or artificial, or whether there even exists any upward trend. In this experiment we are primarily interested in examining how our forecasting models discover a hidden trend in the time series and forecast future values. Hence, we will focus on their performances not on historical data but on future data.

In the dynamic linear model framework the forecasting problem can be treated as the
problem of missing observations. In other words, if we are interested in forecasting a series of future observations from \( t + 2 \) to \( t + T \), then we can compute them via the Kalman filter merely by treating \( \{y_{t+1}, \ldots, y_{t+T-1}\} \) as missing values. A GP model does not need such replacement because it forecasts future values as a function of time.

We first apply to this data dynamic multivariate autoregressive (DMAR) models of various AR orders – as this time series is one-dimensional the term ‘dynamic autoregressive’ would be more suitable, but for consistency we shall stick to the term ‘DMAR’. To demonstrate how these models perform, refer to Figure 5.2. We can see that when an AR order is small (i.e. (a) \( p = 1 \) or (b) \( p = 5 \)), the DMAR models failed to forecast any temperature trends because with such a small AR order the models are able to detect no more than noise. However, if we consider more past observations (i.e. (c) \( p = 10 \) or (d) \( p = 20 \)), the models predicted temperature forecasts showing an upward trend, with a steeper increase rate for DMAR(\( p = 20 \)). This improvement results from the fact that looking further backwards allows the models to find better any directional movement of temperature.
Figure 5.2: Predicted values for annual surface temperature anomaly by a DMAR model of an AR order: (a) $p = 1$, (b) $p = 5$, (c) $p = 10$ and (d) $p = 20$. Observed values are represented by a black line, and predictions by a red line. Shaded areas depict a 95% confidence region of the predictions. DMAR models with a higher AR order predicted temperature forecasts showing an upward trend.
5.3 Results

We now consider a multivariate local trend (MLT) model in which a hidden local trend is estimated within a window of recent observations (again, it is more appropriate to drop the term ‘multivariate’, but we retain it for consistency). We used two different sizes of a data window, namely \( n = 50 \) and \( n = 80 \); we sequentially build a MLT model with 50 or 80 past data points from 1880 to year 2007, and forecast values for the next 33 years using the trend found at 2007. Figure 5.3 illustrates forecasting results of the models and corresponding local trend estimates. When 50 past data points were used, an estimated local trend (i.e. \( \hat{F}_t \) in Equation (5.36)) at 2007 was 1.0598, which gave rise to a steep upward trend (Figure 5.3 (a)). Meanwhile, when more past data points were considered, the estimated local trend at 2007 was 1.0375 and this model produced less steeply increasing forecasts (Figure 5.3 (c)). Comparing between the two series of local trend estimates by MLT(\( n = 50 \)) and the MLT(\( n = 80 \)), we can find that the former is more erratic, ranging from 0.2628 to 1.0818, whereas the latter is more stable, varying between 0.9139 and 1.0461, which seems more realistic (Figure 5.3 (b) and (d)). Note that the predicted values of the MLT models are smooth compared with that of a DMAR model displayed Figure 5.2, in which future temperature forecasts wobble despite no such features being noticed from past observations.

As the third forecaster we consider a GP model, in which a covariance function determines the relationship between values at different times. We used a squared exponential covariance function given in Equation (5.40) because it is a prototypical choice when no prior information about data is available. When a GP model used one squared exponential term that represents local disturbances in temperature, the model could not forecast an upward trend (Figure 5.4 (a)). To allow a GP model to deal with a trend, we considered a covariance function that is the sum of two squared exponential terms: one for modelling short term irregularities and the other for modelling a long-term trend, i.e. \( c(t, t') = \tau_1 c_1(t, t') + \tau_2 c_2(t, t') \). Here \( c_1 \) and \( c_2 \) are a squared exponential covariance function, and \( \tau_1 \) and \( \tau_2 \) are weights. A squared exponential covariance function is chosen because we assume no prior knowledge about the data. A GP model using this covariance function predicted a mildly increasing trend (Figure 5.4 (b)). It should be noted that the predicted values by the GP model are
5.3 Results

Figure 5.3: Predicted values for annual surface temperature anomaly by a MLT model with a data window of size (a) $n = 50$ and (c) $n = 80$. (b) and (d) depict time plots of estimated local trends for MLT($n = 50$) and MLT($n = 80$) respectively; that is, each represents an estimate annual change rate of temperature. A MLT model considering a larger number of past data (corresponding (c) and (d)) appears to estimate more realistic annual temperature changes.
smooth as it models underlying true temperature, not noisy observations. This is a nice property of a GP model in comparison to a dynamic autoregressive model.

Figure 5.4: Predicted values for annual surface temperature anomaly by a GP model with a covariance function having (a) one squared exponential term (b) two squared exponential terms. A GP model with two squared exponential terms (one for local disturbances and the other for a trend) found a mildly increasing trend.

In this section we applied three forecasting models to annual surface temperature data. Each model provided us with a different forecasting result on a future trend of the temperature. As discussed earlier, no one could argue with certainty that an upward trend of the global surface temperature exists, or how rapidly it grows if any. The question of how to make use of a model and how to interpret it is a challenge faced by a data modeller.

5.3.2 Multiple Sensor Case

We consider air temperature data collected by a network of weather sensors located off the coast near Southampton (Chan, 2000). Three sensors located at Chichester Bar, Southampton Dockhead and Bramble Pile read the weather condition every second, and every five minutes the readings are averaged and uploaded to the internet with free-to-the-public access (Figure 5.5). Figure 5.6 (a) illustrates an example of such data of size $t = 1,493$ (i.e. about 5.2 days). We can see that there are many missing observations; such occurrences of missing
5.3 Results

Data are due to a failure of transmitting collected data to the internet. Moreover, the sensor readings display a strong degree of correlation; as the three sensor locations are a few tens of kilometers apart from each other, the air temperature readings are expected to be reasonably similar. These conditions are common in real-world applications and what we are most interested in studying. An important feature of this data is a periodic cycle that describes daytime ups and nightly downs in air temperature. For analysis of this data we use three time series models detailed in Section 5.2. Missing values are recovered after a certain amount of time has passed and thus true air temperature readings are available (Figure 5.6 (b)). As a result, we can evaluate forecasting performances of the models. Five grey regions in the figures are of our particular interest as these are periods when much data was missing.

![Figure 5.5](image)

Figure 5.5: The Bramble Bank weather station and associated web site (see [www.bramblemet.co.uk](http://www.bramblemet.co.uk)). This figure is provided by Osborne et al. (2008c).

We consider two DMAR models of AR order, \( p = 1 \) and \( p = 15 \). To demonstrate how these models perform in sparse and correlated data, refer to Figure 5.7. First, the DMAR(\( p = 1 \)) model performed very poorly: except for the first grey area it failed to predict any upward or downward trends when sensor readings are partially or completely missing. On the contrary, the DMAR(\( p = 15 \)) model predicted reasonably well an upward trend of the sensors in the second interest region. However, the model could not capture the daily cyclical pattern of the data and predicted poor forecasts in the third and fourth grey areas.
Figure 5.6: (a) Air temperature data collected in real time by a network of weather sensors located at Chichester Bar (red), Southampton Dockhead (green) and Bramble Pile (blue). (b) True air temperature readings recovered afterwards. Five grey regions are of our particular interest as these are periods when missing values regularly occurred.
where all sensors were missing.

Figure 5.7: Predicted values for three sensors by a DMAR model of AR order (a) \( p = 1 \) and (b) \( p = 15 \). Pink areas represent a 95% confidence region of the predictions. A DMAR model with a higher AR order performed better particularly in the second interest region. However, both models failed to predict a daily cyclical pattern of air temperature as shown in the third and fourth grey areas.

We now consider a MLT model in which a state evolution matrix is sequentially estimated from a set of observations up to date via the expectation-maximisation algorithm. Figure 5.8 illustrates the predicted values of the three sensors computed by the MLT model. In the second region of interest, long sequences of missing values of the red and blue sensors were
well predicted when the green sensor readings were available. However, when all sensors were missing, the model could not predict a daily cyclical pattern of the data. We can see this failure in the third and fourth grey areas as well. This is an expected failure as the MLT model is a first-order Markov model and it only considers local trends of the sensors. We will shortly discuss a method that allows DMAR and MLT models to deal with a periodic cycle as well as a local trend.

![Multivariate Local Trend Model](image)

Figure 5.8: Predicted values for three sensors by a MLT model. Despite partial missing values being well-predicted, the MLT model, which is based on a first-order Markov model, failed to capture a daily cyclical pattern of air temperature.

We also consider a GP model, in which a covariance function is constructed so as to account for two main characteristics of the air temperature data: correlated sensor readings and a daily cycle of temperature. We used a covariance function that consists of three terms: (1) a periodic term whose period is found to be almost exactly 1 day, (2) a longer-term drift term having a time scale of 3 days and (3) a higher frequency disturbance term possessing a time scale of 4 hours (Osborne and Roberts, 2007). To illustrate how the GP model performed, refer to Figure 5.9. We can see that partial missing values were well-predicted on the basis of observed information. Moreover, the daily periodic feature was captured by the model, and accordingly it predicted well the missing values when all sensors were unavailable.
outstanding result is not surprising as we explicitly incorporated prior knowledge about the data into the model in the form of a covariance function.

![Gaussian Process Model](image)

Figure 5.9: Predicted values for three sensors by a GP model. Since the GP model incorporated prior knowledge about the data such as a daily cyclical pattern into a covariance function, it successfully predicted missing values even when all sensors were missing.

**Cycle-Augmented Model**

Dynamic-linear-model-based models (i.e. DMAR and MLT) were found to be unsuitable for data possessing a periodic cycle. There are various ways of allowing for a periodic term in traditional statistical time series analysis (e.g. Chatfield (2004)). We here discuss an idea of incorporating a periodic cycle term into a dynamic linear model.

We know that in our air temperature data, sensor readings are observed every 5 minutes, thereby giving rise to a daily cycle having a period of 288 time steps. To allow a dynamic linear model to deal with the cycle we introduce a variable for such a cyclic pattern and treat its values as if they were observations. In other words, we create a time series, denoted by $s_t$, having a sine wave with a period of a day (Figure 5.10), and use the following ‘augmented’ observation vector:

$$\tilde{y}_t = \begin{bmatrix} y_t \\ s_t \end{bmatrix}. \tag{5.42}$$
Results

The cycle variable $s_t$ can be thought of as a ‘time stamp’, which provides information about which time of a day to forecast. In other words, even when all sensors are missing, the information is still available to a forecasting model, which allows it to predict missing values possessing a cyclical period. In the analysis of our air temperature data it is sufficient to consider a single periodic term because all sensors measure the same object. However, we may need to consider more than one periodic terms; for example, when each sensor has a cycle of different period. This case could be solved by adding several “artificial” observation variables for the cycles to an original observation variable.

To illustrate how DMAR and MLT models perform when using the above new observations, refer to Figure 5.11: (a) and (b) depict the results of DMAR models of AR order $p = 1$ and $p = 15$ respectively; (c) shows that of a MLT model. In comparison to their corresponding results when using original observations, which are displayed in Figures 5.7 and 5.8, we can easily see improvements made by augmenting the cycle variable. In particular, the DMAR($p = 15$) model correctly captures the daily cyclical pattern and predicted well missing values even when all sensors were unavailable. It is interesting to note that the first-order Markov models (i.e. DMAR($p = 1$) and MLT) underperformed compared to the higher-order model, DMAR($p = 15$). This result indicates that a high-order Markov model is more suitable for discovering an underlying trend as it looks further backward.
Figure 5.11: Predicted values for three sensors by (a) the DMAR($p = 1$), (b) the DMAR($p = 15$) and (c) the MLT when cycle-augmented observations are used. By allowing the models to be informed of a daily cyclical pattern, we obtained much improved performances, particularly for the DMAR($p = 15$) model. It predicted very well the periodic pattern of the air temperature even when all sensors were missing.
5.3 Results

Comparison

To evaluate performances of the three multivariate forecasting models studied so far, we computed the root mean square error of each model for the three sensors (Table 5.1). For the DMAR and MLT models we used cycle-augmented observations, and the AR order of the DMAR was $p = 15$. We can see that the high-order Markov model, DMAR($p = 15$), and the GP model outperformed the first-order Markov model, MLT; the MLT model stood out in Area 2 but it performed poorly in other areas, which can be visually checked in Figure 5.11 (c). From the bottom part of Table 5.1 we can see that the GP model was better for the blue and red sensors, and the DMAR model for the green sensor. To check if these performance differences between the two models are statistically significant, we performed a two-sided t-test; the null hypothesis is that the differences between the two models’ prediction errors are a random sample from a Gaussian distribution with mean 0 and unknown variance. For the blue sensor the null hypothesis is not rejected with the $p$-value being 0.7677. This indicates that DMAR and GP did not perform differently for the blue sensor. For the green and red sensors the $p$-value is 0.0005 and 0.003 respectively, thereby rejecting the null hypothesis. These results confirm that DMAR performed better for the green sensor than GP, and for the red sensor GP better than DMAR.

In this section we analysed real-world air temperature data, in which multiple (correlated) sensors measured noisy and periodic air temperature, and frequent (partially or completely) missing observations occurred. These features of the data provided a nice testbed for our multivariate forecasting models. By using cycle-augmented observations we could improve DMAR and MLT models in a way that they deal with a daily periodic cycle in air temperature. We summarise characteristics of these forecasting models (Table 5.2). All the models are suitable for predicting a trend. However, the GP model accounts for periodic or seasonal terms in a more elegant manner as a covariance function encapsulates these effects. This feature gives rise to more flexibility to the GP model in comparison to the DMAR and MLT models. In return for the flexibility the GP model has more hyper-parameters to adjust. This is not a trivial problem, but Osborne et al. (2008a) proposed a Bayesian method that allows
### Table 5.1: Root mean square error of predictions by DMAR, MLT and GP models.

<table>
<thead>
<tr>
<th>Area</th>
<th>Sensor 1 (Blue)</th>
<th>Sensor 2 (Green)</th>
<th>Sensor 3 (Red)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DMAR</td>
<td>MLT</td>
<td>GP</td>
</tr>
<tr>
<td>Area 1</td>
<td>1.3302</td>
<td>1.1178</td>
<td><strong>0.8458</strong></td>
</tr>
<tr>
<td></td>
<td>2.3824</td>
<td><strong>1.3243</strong></td>
<td>3.7949</td>
</tr>
<tr>
<td></td>
<td>0.9999</td>
<td><strong>0.6791</strong></td>
<td>1.4998</td>
</tr>
<tr>
<td>Area 2</td>
<td>1.1664</td>
<td><strong>0.5920</strong></td>
<td>1.0728</td>
</tr>
<tr>
<td></td>
<td>0.7465</td>
<td><strong>0.6200</strong></td>
<td>0.7783</td>
</tr>
<tr>
<td></td>
<td>1.5141</td>
<td><strong>0.5647</strong></td>
<td>0.9065</td>
</tr>
<tr>
<td>Area 3</td>
<td>0.5928</td>
<td>2.7788</td>
<td><strong>0.5451</strong></td>
</tr>
<tr>
<td></td>
<td><strong>1.9748</strong></td>
<td>4.2150</td>
<td>2.0685</td>
</tr>
<tr>
<td></td>
<td><strong>0.4213</strong></td>
<td>2.7883</td>
<td>0.5575</td>
</tr>
<tr>
<td>Area 4</td>
<td><strong>0.2009</strong></td>
<td>2.3582</td>
<td>0.7000</td>
</tr>
<tr>
<td></td>
<td><strong>1.6594</strong></td>
<td>3.9596</td>
<td>1.7338</td>
</tr>
<tr>
<td></td>
<td><strong>0.3262</strong></td>
<td>2.4805</td>
<td>0.9191</td>
</tr>
<tr>
<td>Area 5</td>
<td><strong>0.4141</strong></td>
<td>0.4253</td>
<td>0.6731</td>
</tr>
<tr>
<td></td>
<td>0.8549</td>
<td>0.9313</td>
<td><strong>0.8332</strong></td>
</tr>
<tr>
<td></td>
<td>0.4667</td>
<td>0.2312</td>
<td><strong>0.1970</strong></td>
</tr>
<tr>
<td>Total</td>
<td><strong>0.7966</strong></td>
<td>1.0141</td>
<td><strong>0.7267</strong></td>
</tr>
<tr>
<td></td>
<td><strong>1.0294</strong></td>
<td>1.5556</td>
<td>1.0439</td>
</tr>
<tr>
<td></td>
<td>0.9437</td>
<td>1.0237</td>
<td><strong>0.7005</strong></td>
</tr>
</tbody>
</table>
the GP model to infer the values of the hyper-parameters in an automated sequential manner. These useful properties of the GP model come at the cost of very high computation. Meanwhile, the DMAR model is computationally simple and rapid in its estimations.

For the air temperature data analysed here, the dynamic multivariate autoregressive model using cycle-augmented observations performed as well as the Gaussian process model. Moreover, the former is computationally considerably faster than the latter. We cannot conclude which model is universally superior. Generally speaking, there exists no such ‘blind black-box’ model in which we put in a data point and receive a best-possible prediction. It should be a data-modeller’s wise decision which model to use in what circumstances.

<table>
<thead>
<tr>
<th></th>
<th>DMAR</th>
<th>MLT</th>
<th>GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trend</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cycle</td>
<td>Partial yes</td>
<td>Partial yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Complexity</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Computation</td>
<td>Very cheap</td>
<td>Expensive</td>
<td>Very expensive</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison between three multivariate forecasting models.

In the next chapter we discuss the on-line problem of decision making in which a binary classification is made for a given input. Often, information about classification labels and/or inputs is not available. In particular, the case of missing inputs is closely related to the problem of multivariate time series forecasting: we will consider using one of the forecasting methods presented in this chapter to predict the missing information.
Many data analysis problems require on-line learning and we have so far studied two on-line problems: (1) finding an optimal policy in a Markov decision process detailed in Chapter 4; (2) forecasting future values of a multivariate time series discussed in Chapter 5. Another interesting on-line problem is that of adaptive classification, which is the subject of this chapter. As with other on-line problems we are faced with noisy, non-stationary, incomplete environments, which make a classification in an on-line manner a significant challenge. To tackle this challenge, we adopt the dynamic model framework. For a binary classification problem, we generally train a logistic regression model to produce posterior class probabilities. A dynamic logistic regression model proposed by Penny and Roberts (1999) uses the extended Kalman filter (Section 2.4.1) to solve the dynamic model. We suggest using the unscented Kalman filter (Section 2.4.2) as a variant of the dynamic logistic regression. In addition, we discuss how the binary classification problem can be solved by a dynamic generalised linear models detailed in Chapter 3. We apply these adaptive classification models to synthetic and real data; carry out experiments to see how these models handle the cases in which parts of information are missing. Results from numerical examples are presented.
6.1 Dynamic Logistic Regression

In this section we deal with methods of binary classification. For a static problem an ordinary logistic regression model is widely used. To enable sequential classification we can consider a dynamic version of the simple logistic regression. We discuss two dynamic logistic regression models: using the extended Kalman filter and the unscented Kalman filter.

6.1.1 Logistic Regression

For a binary decision problem, target labels are represented by $y = 0$ and $y = 1$. We are interested in inferring a class probability defined by

$$\pi(h) \triangleq \Pr(y = 1|h),$$

where $h$ is a given input vector. By definition, $\Pr(y = 0|h) = 1 - \pi(h)$. Logistic regression models the class probability as follows:

$$\pi(h) = \ell(h^Tw),$$

where $w$ is a vector of weights and $\ell(\cdot)$ is a logistic function defined by

$$\ell(h^Tw) = \frac{\exp(h^Tw)}{1 + \exp(h^Tw)}.$$  \hspace{1cm} (6.3)

Instead of using the input vector \textit{per se}, we can consider its extended form:

$$\varphi(h) = \begin{cases} h \\ \phi(h) \\ 1 \end{cases},$$

where $\phi$ represents a set of non-linear kernel functions, such as Gaussian radial basis functions. Throughout this chapter we consider, unless otherwise stated, a logistic function with activation $a = \varphi(h)^Tw$, not $a = h^Tw$. With this kernel input vector it is possible to find a non-linear boundary between two classes. The benefit of adding the unity component (i.e. 1) and the linear term (i.e. $h$) into the kernel vector $\varphi(h)$ is described in Lowne et al. (2008).
Bayesian approach

Suppose that the weight vector \( \mathbf{w} \) is a random variable having a probability distribution \( p(\mathbf{w}) \). The class probability given in Equation (6.2) ignores any uncertainty associated with the weight vector. A Bayesian approach takes into account its distribution, and accordingly, moderates the class probability. If it is assumed that the distribution of \( \mathbf{w} \) is a multivariate Gaussian with mean \( \hat{\mathbf{w}} \) and covariance \( \mathbf{P}_w \), then the distribution of activation \( a \) is also a Gaussian with mean \( \hat{a} \) and variance \( c^2 \) defined by

\[
\hat{a} = \varphi(h)^T \hat{\mathbf{w}}, \quad (6.5)
\]
\[
c^2 = \varphi^T(h) \mathbf{P}_w \varphi(h). \quad (6.6)
\]

By integrating \( a \) out, the posterior class probability \( \pi \) is moderated; that is,

\[
\pi = \int \ell(a)p(a)da. \quad (6.7)
\]

Since it is impossible to compute this probability analytically, it is approximated by (MacKay, 1992):

\[
\pi \approx \ell(\kappa(c^2)\bar{a}), \quad (6.8)
\]

where

\[
\kappa(c^2) = \left(1 + \frac{\pi c^2}{8}\right)^{-1/2}. \quad (6.9)
\]

Here \( \pi \) represents the constant approximately equal to 3.14159. The effect of this moderation is illustrated in Figure 6.1. If the variance of activation is close to zero, the extent of moderation is negligible. However, as the variance becomes very large, the posterior class probability is moderated to lie closer to the prior at 0.5.

6.1.2 Non-stationary Logistic Regression

In the stationary logistic regression model given in Equation (6.2), the weight vector \( \mathbf{w} \) is assumed to be static. Penny and Roberts (1999) proposed the dynamic logistic regression in
6.1 Dynamic Logistic Regression

Figure 6.1: Moderated posterior class probability for different sizes of the variance of activation.

which the weight vector evolves according to the random walk. It has the following form:

\[ \pi_t = \ell(\varphi(h_t)^T w_t), \]
\[ w_t = w_{t-1} + v_t, \]  

(6.10)  
(6.11)

where \( v_t \) is a state noise variable assumed to be a Gaussian with mean 0 and covariance \( q_t I \). Note that Equation (6.10) does not include a noise term. In the dynamic linear model, as an observation variable \( y \) is assumed to be Gaussian, it is natural to relate the observation variable with a Gaussian noise. However, in the dynamic logistic model an observation variable \( \pi \) is the probability of a class variable \( y \); that is, \( y \) follows a Bernoulli distribution with parameter \( \pi \). Therefore, it would not be sensible to relate \( \pi \) to a noise term. This issue will be covered when we deal with a dynamic classification problem in the framework of the dynamic generalised linear model in Section 6.2. From now on we simplify the notation \( \varphi(h_t) \) to \( \varphi_t \), and the term ‘state variable’ is preferred to ‘weight vector’.

In the dynamic logistic regression the non-stationarity is captured by a time-evolving state variable \( w_t \), and a binary decision is made according to the logistic function. In order
6.1 Dynamic Logistic Regression

to conduct adaptive classification processes, we have to estimate the state variable \( w_t \) in an on-line manner. This dynamic logistic regression can be thought of as a non-linear dynamic model detailed in Section 2.4, in which an observation process is the non-linear logistic function and a state process is a linear first-order Markov process. Hence, instead of the standard Kalman filter, a non-linear Kalman filter is more suitable for solving the adaptive classification problem.

**Dynamic Classification using the Extended Kalman Filter**

The extended Kalman filter is the most commonly used non-linear filter. We discuss how the extended Kalman filter can be used to solve the dynamic logistic regression. Suppose that we have the posterior mean and covariance of state variable \( w_{t-1} \) denoted by \( \hat{w}_{t-1|t-1} \) and \( P_{t-1|t-1} \) respectively. Since the state process is linear, we can easily compute the prior mean and covariance of \( w_T \) given \( D_{t-1} = \{y_1, y_2, \ldots, y_{t-1}\} \) as follows:

\[
\hat{w}_{t|t-1} = \hat{w}_{t-1|t-1},
\]

\[
P_{t|t-1} = P_{t-1|t-1} + q_t I. \tag{6.12}
\]

Owing to the non-linear observation function, it is impossible to compute exactly the mean and variance of the one-step ahead forecast distribution. Thus the non-linear function is approximated by

\[
\pi_t = \ell(\varphi^T_t w_t) \\
\approx \ell(\varphi^T_t \hat{w}_{t|t-1}) + A_t^T (w_t - \hat{w}_{t|t-1}), \tag{6.13}
\]

where

\[
A_t \triangleq \frac{\partial \ell(w_t)}{\partial w_t} \bigg|_{w_{t|t-1}} \\
= \ell(\varphi^T_t \hat{w}_{t|t-1}) \left(1 - \ell(\varphi^T_t \hat{w}_{t|t-1})\right) \varphi_t. \tag{6.14}
\]

The one-step ahead prediction for the class probability is computed according to

\[
\hat{\pi}_{t|t-1} = \ell(\varphi^T_t \hat{w}_{t|t-1}), \tag{6.15}
\]
and its variance is
\[ \text{Var}(\pi_t|D_{t-1}) = A_t^T P_{t|t-1} A_t + r_t, \] (6.16)

where \( r_t \) is a prior observation noise variance for the observed label \( y_t \). From the fact that a binary label variable has a Bernoulli distribution, we can estimate the observation noise variance according to
\[ r_t = \hat{\pi}_{t|t-1} (1 - \hat{\pi}_{t|t-1}). \] (6.17)

Alternatively, we can set \( r_t \) to a fixed value. So long as \( r_t > 0 \), this added observation noise variance prevents singularities in the inverse observation variance.

After observing a new label \( y_t \), we update the prior distribution of the state variable. The posterior mean and covariance of the state variable are computed according to
\[ \hat{w}_{t|t} = \hat{w}_{t|t-1} + K_t (y_t - \hat{\pi}_{t|t-1}), \] (6.18)
\[ P_{t|t} = P_{t|t-1} - K_t (A_t^T P_{t|t-1} A_t + r_t) K_t^T. \] (6.19)

The Kalman gain \( K_t \) is computed as follows:
\[ K_t = P_{t|t-1} A_t (A_t^T P_{t|t-1} A_t + r_t)^{-1}. \] (6.20)

In the prediction context, Jazwinski (1970) proposed an idea for adaptively estimating noise variances by maximising the ‘evidence’ of observations. Penny and Roberts (1999); Lowne et al. (2008) modified the idea to make it suitable for the classification context by maximising the ‘evidence’ of model predictions instead. For example, Lowne et al. (2008) suggested updating state noise variance \( q_t \) according to
\[ q_t = \max \{ u_{t|t} - u_{t|t-1}, 0 \}, \] (6.21)
where
\[ u_{t|t} = \hat{\pi}_{t|t} (1 - \hat{\pi}_{t|t}) \] (6.22)
is the uncertainty (variance) in the predictive Bernoulli distribution.
6.1 Dynamic Logistic Regression

6.1.3 Dynamic Classification using the Unscented Kalman filter

In Section 2.4.2 we have dealt with the unscented Kalman filter and discussed its advantages over the extended Kalman filter. Hence, the application of the nonlinear Kalman filter to the dynamic classification problem may be useful. In this section we present a novel dynamic classification algorithm employing the UKF. As with the case of using the extended Kalman filter, the prior mean and covariance of the state variable \( w \) are given by

\[
\hat{w}_t | t - 1 = \hat{w}_{t-1} | t - 1, \quad (6.23)
\]

\[
P_t | t - 1 = P_{t-1} | t - 1 + q_t I. \quad (6.24)
\]

In order to compute the mean and variance of the one-step ahead prediction distribution, the unscented transformation method is employed. We first draw sigma vectors for the distribution of \( w \) from its estimated mean and covariance, \( \hat{w}_t | t - 1 \) and \( P_t | t - 1 \):

\[
\mathcal{W}_t | t - 1 = \left[ \mathcal{W}_t | t - 1, \ldots, \mathcal{W}_t | t - 1 \right], \quad (6.25)
\]

where \( W \) is the dimension of \( w \), \( \lambda = \alpha^2 (W + \kappa) - L \), and \( \left( \sqrt{(W + \lambda)P_t | t - 1} \right)_i \) is the \( i \)th column of the matrix square root. The hyperparameters (i.e. \( \alpha, \kappa, \lambda \)) are detailed in Section 2.4.2.

The sigma vectors are propagated through the logistic function:

\[
\mathcal{Y}_t | t - 1 = \ell \left( \varphi_i^T \mathcal{W}_t | t - 1 \right), \quad i = 0, 1, \ldots, 2W \quad (6.26)
\]

Using these propagated sigma vectors we can compute a one-step ahead predicted class probability and its variance, which are given by

\[
\hat{\pi}_t | t - 1 = \sum_{i=0}^{2W} C_i^{(m)} \mathcal{Y}_t | t - 1 \quad (6.27)
\]

\[
P_{yy} = \sum_{i=0}^{2W} C_i^{(c)} (\mathcal{Y}_t | t - 1 - \hat{\pi}_{t-1})^2 + r_t. \quad (6.28)
\]
The prior observation noise variance $r_t$ can be either fixed to a static small value or estimated according to

$$r_t = \hat{\pi}_{t|t-1}(1 - \hat{\pi}_{t|t-1}).$$

In addition, we compute the covariance between the state variable and the class probability according to

$$P_{wy} = \sum_{i=0}^{2W} C_i^{(c)}(W_{t|t-1} - \hat{w}_{t|t-1})(W_{t|t-1} - \hat{\pi}_{t|t-1})^T.$$  \hspace{1cm} (6.29)

The weights $C_i^{(m)}$ and $C_i^{(c)}$ are given by

$$
C_0^{(m)} = \frac{\lambda}{W + \lambda},
C_0^{(c)} = \frac{\lambda}{W + \lambda} + 1 - \alpha^2 + \beta,
C_i^{(m)} = C_i^{(c)} = \frac{1}{2(W + \lambda)}, \quad i = 1, \ldots, 2W,
$$

where $\beta$ is another hyper-parameter detailed in Section 2.4.2.

After observing a new label $y_t$, the state variable distribution is updated. The posterior mean and covariance of $w_t$ are computed according to

$$
\hat{w}_{t|t} = \hat{w}_{t|t-1} + K_t(y_t - \hat{\pi}_{t|t-1})
$$
\hspace{1cm} (6.31)

$$
P_{t|t} = P_{t|t-1} - K_tP_{yy}K_t^T,
$$
\hspace{1cm} (6.32)

where the Kalman filter is

$$K_t = P_{wy}P_{yy}^{-1}.$$  \hspace{1cm} (6.33)

In Section 6.1.1 we introduced the Bayesian approach to the logistic regression model in order to take into account the uncertainty associated with the state variable $w$. Note that the unscented Kalman filter concerns the distribution of $w$ by selecting sigma points from the distribution.

## 6.2 Dynamic Classification using the Dynamic Binomial Model

The sequential classification problem can be viewed from a different perspective. Dynamic generalised linear models detailed in Chapter 3 are a powerful method for analysing any distributions in the exponential family. Since a binary label follows a Bernoulli distribution,
which is a special case of a binomial distribution, the dynamic classification problem can be solved in the framework of the dynamic generalised linear model. It should be noted that in the dynamic logistic model given in Equations (6.10) and (6.11) the observation variable \( \pi \) represents a probability of a class variable \( y \); that is, \( y \) follows a Bernoulli distribution with \( \pi \). For this reason, it would be more natural to treat the dynamic classification problem under the framework of the dynamic generalised linear model. In this section we discuss how a dynamic binomial model can provide a solution to the dynamic classification problem.

The total number of ‘successes’ \( y \) in \( n \) independent experiments follows a binomial distribution with probability of success \( \pi \). The binomial distribution is a member of the exponential family and it is defined by

\[
p(y \mid n, \pi) = \binom{n}{y} \pi^y (1 - \pi)^{n-y},
\]

where \( 0 \leq \pi \leq 1, n = \{0, 1, 2, \ldots\} \) and \( y = \{0, 1, 2, \ldots, n\} \). We can re-arrange the above density function in the following way:

\[
p(y \mid n, \pi) = \exp \left( n \left[ \frac{y}{n} \log \left( \frac{\pi}{1 - \pi} \right) - \log \left( \frac{1}{1 - \pi} \right) \right] + \log \left( \frac{n}{y} \right) \right).
\]

Comparing this with the general form of distributions in the exponential family,

\[
p(y \mid \theta, \phi) = \exp \left( \frac{d(y) \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right),
\]

we can find that \( \theta = \log \left( \frac{\pi}{1 - \pi} \right), a(\phi) = \frac{1}{n}, b(\theta) = \log(1 + \exp(\theta)), c(y, \phi) = \log \left( \frac{n}{y} \right) \) and \( d(y) = \frac{y}{n} \). The mean and variance of the binomial variable are

\[
E \left( \frac{Y}{n} \mid n, \pi \right) = \mu = \pi, \quad \text{Var} \left( \frac{Y}{n} \mid n, \pi \right) = \frac{1}{n} \pi (1 - \pi).
\]

For the binomial distribution, the canonical link that makes the linear predictor \( \eta \) equal to the canonical parameter \( \theta \) is the \textit{logit function} (the inverse of the logistic function) defined by

\[
\logit (p) = \log \left( \frac{p}{1 - p} \right).
\]
To solve the dynamic binary classification problem we consider a dynamic binomial model, in which time index $t$ is added to $y$ and $\pi$ in Equation (6.34). The link equation is defined by

$$g(\mu_t) = \eta_t = \varphi(h_t)^T w_t. \quad (6.40)$$

In addition, we assume that the state variable $w_t$ evolves according to

$$w_t = w_{t-1} + v_t, \quad (6.41)$$

where $v_t \sim (0, q_t I)$. The following derivation of the dynamic binomial model may be better understood with reference to Section 3.2.

Suppose that we have the posterior mean and covariance of $w_{t-1}$, given a set of observations $D_{t-1} = \{y_1, \ldots, y_{t-1}\}$. They are denoted by $\hat{w}_{t-1|t-1}$ and $P_{t-1|t-1}$ respectively. Owing to the linear state process, the prior distribution of the state variable is

$$w_t | D_{t-1} \sim (\hat{w}_{t|t-1}, P_{t|t-1}), \quad (6.42)$$

where

$$\hat{w}_{t|t-1} = \hat{w}_{t-1|t-1}, \quad (6.43)$$

$$P_{t|t-1} = P_{t|t-1} + q_t I. \quad (6.44)$$

The mean and variance of the prior distribution of the linear predictor can be easily computed by

$$\hat{\eta}_{t|t-1} = \varphi_t^T \hat{w}_{t|t-1}, \quad (6.45)$$

$$r_{t|t-1} = \varphi_t^T P_{t|t-1} \varphi_t. \quad (6.46)$$

To derive the one-step ahead forecast distribution given by

$$p(y_t | D_{t-1}) = \int p(y_t | \theta_t) p(\theta_t | D_{t-1}) d\theta_t, \quad (6.47)$$

we need to specify the conjugate prior, $p(\theta_t | D_{t-1})$. According to the derivation given in Appendix A.2, we can find that the prior conjugate distribution is

$$p(\theta_t | D_{t-1}) = \frac{\Gamma(m_t)}{\Gamma(k_t) \Gamma(m_t - k_t)} \frac{\exp(k_t \theta_t)}{(1 + \exp(\theta_t))^{m_t}}, \quad (6.48)$$
and its first two moments are approximately
\[
\begin{align*}
E(\theta_t|D_{t-1}) & \approx \log \left( \frac{k_t}{m_t - k_t} \right), \\
\text{Var}(\theta_t|D_{t-1}) & \approx \frac{1}{k_t} + \frac{1}{m_t - k_t}.
\end{align*}
\] (6.49) (6.50)

We know that the mean and variance of the canonical parameter are equivalent to that of the linear predictor because a canonical link is used. Hence, we can calculate the values of the hyper-parameters as follows:
\[
\begin{align*}
k_t &= \frac{1}{r_{tt-1}} (1 + \exp(\eta_{t|t-1})), \\
m_t &= \frac{1}{r_{tt-1}} (1 + \exp(-\eta_{t|t-1}))(1 + \exp(\eta_{t|t-1})).
\end{align*}
\] (6.51) (6.52)

With these hyper-parameter values, the one-step ahead forecast distribution can be analytically obtained as
\[
p(y_t|D_{t-1}) = \int_{-\infty}^{\infty} p(y_t|\theta_t)p(\theta_t|D_{t-1})d\theta_t,
\] (6.53)

A full derivation is given in Appendix A.3. This is a beta-binomial distribution with parameters \(k_t\) and \(m_t - k_t\), and accordingly the mean and variance of the forecast distribution are
\[
\begin{align*}
E(y_t|D_{t-1}) &= \frac{nk_t}{m_t}, \\
\text{Var}(y_t|D_{t-1}) &= \frac{nk_t(m_t - k_t)(m_t + n)}{m_t^2(1 + m_t)}. \\
\end{align*}
\] (6.54) (6.55)

After observing a new datum \(y_t\), we update the distributions of \(\theta_t\) and \(\eta_t\). The posterior distribution of \(\theta_t\) is of form
\[
p(\theta_t|D_t) \propto p(\theta_t|D_{t-1})p(y_t|D_t)
\] (6.56)
The approximate values of the mean and variance of the posterior distribution are

\[
E(\theta_t|D_t) \approx \log \left( \frac{k_t + y_t}{m_t - k_t + n - y_t} \right), \quad (6.57)
\]

\[
\text{Var}(\theta_t|D_t) \approx \frac{1}{k_t + y_t} + \frac{1}{m_t - k_t + n - y_t}. \quad (6.58)
\]

They are equivalent to the mean and variance of the posterior distribution of \( \eta_t \), which are represented by \( \hat{\eta}_{t|t} \) and \( r_{t|t} \) respectively. Finally, the posterior mean and covariance of \( w_t \) are

\[
\hat{w}_{t|t} = \hat{w}_{t|t-1} + \frac{P_{t|t-1}^T \varphi_t}{r_{t|t-1}} (\hat{r}_{t|t} - \hat{r}_{t|t-1}), \quad (6.59)
\]

\[
P_{t|t} = P_{t|t-1} - \frac{P_{t|t-1}^T \varphi_t}{r_{t|t-1}} \varphi_t^T P_{t|t-1} \left( 1 - \frac{r_{t|t}}{r_{t|t-1}} \right). \quad (6.60)
\]

The dynamic classification problem can be solved by a special case of the dynamic binomial model in which the number of trials, \( n \), is set to 1 and the number of success, \( y_t \), is either 0 or 1. Hence, the expected value of the observation, \( E(y_t) \), is equal to the success probability, \( \pi_t \). In the approach of using the non-linear Kalman filters discussed in Sections 6.1.2 and 6.1.3, the class (or success) probability \( \pi_t \) is directly predicted from inputs via the logistic function. However, in the dynamic binomial model approach it is predicted in a full Bayesian manner, allowing for the explicit distribution form of the class variable and its conjugate prior distribution. We will compare performances of these methods in a real-world dynamic classification problem later in this chapter.

### 6.3 Making use of Past Information

The dynamic classification models that have so far been studied consider a one-to-one mapping between the current input vector and the current class observation. We suggest using past information (i.e. past inputs and past labels) to improve the classification performance of the dynamic models. Firstly, in the prediction context it is not uncommon to use past inputs in predicting the current output. This can be thought of as a “wide-sense” Markov process. A new input vector consisting of the current input and \( p \) previously observed inputs is of form

\[
i_{t,p} = [h_{t}^T, \ldots, h_{t-p}^T]^T. \quad (6.61)
\]
6.4 Results

We refer to $p$ as a *look-back* parameter. Hence, the observation process of the dynamic logistic regression has the form

$$\pi_t = \ell \left( \varphi(i_{t,p})^T w_t \right),$$

(6.62)

and for the dynamic binomial model the link function is

$$\eta_t = \varphi(i_{t,p})^T w_t.$$  

(6.63)

The mapping of multiple quantities of input information onto the current output may reveal hidden patterns between past inputs and the current label and consequently produce better classifications.

Secondly, we can think of incorporating previously observed labels as well as past inputs into the current input vector: a new input vector is of form

$$d_{t,p} = [h_t^T, \ldots, h_{t-p}^T, y_{t-1}, \ldots, y_{t-p}]^T.$$  

(6.64)

By allowing for past labels any unknown patterns between the past labels and the current label could be detected via the new input variable. The dynamic classification models can be readily employed with the new inputs, $\varphi(i_{t,p})$ and $\varphi(d_{t,p})$.

6.4 Results

In this section we experiment the three dynamic classification models detailed in Sections 6.1 and 6.2: (1) the dynamic logistic regression using the extended Kalman filter, (2) the dynamic logistic regression using the unscented Kalman filter, and (3) the dynamic binomial model. We refer to them as ‘DLR-E’, ‘DLR-U’ and ‘DBM’ respectively. We are interested in making classifications sequentially not only when input and label information is completely given, but also when part of the information is missing or incomplete. In addition, we discuss a method that allows an adaptive classification model to actively request labels.

6.4.1 Synthetic Data

We consider two overlapping Gaussian distributions rotating in a circular fashion around a central point at $[0, 0]$, with the two distributions out of phase by $\pi$ radians. Target labels are
interleaved, i.e. \{0, 1, 0, 1, \ldots\} (Figure 6.2). Three data sets are constructed with different Bayes error (Figure 6.3). This data is as presented in Lowne et al. (2008). Owing to the non-stationary nature of the data any static classifier would not be suitable, whereas our dynamic classifiers are expected to adapt the change of the environment and produce a dynamic decision boundary. In the experiments the kernel input vector is set to
\[
\varphi(h_t) = \begin{cases} 
1 \\
h_t 
\end{cases},
\]
(6.65)
i.e. with no basis functions, because we know that the underlying boundary is linear. The vector \(h_t\) represents an input at time \(t\). The state evolution noise variance \(q_t\) is set to 0.1, and for DLR-E and DLR-U the observation noise variance \(r_t\) is adaptively estimated.

When presented with a full set of inputs and labels, we computed one-step ahead predictions of the class probability (i.e. \(\hat{\pi}_{t|t-1}\)) with the adaptive classifiers. The predicted label \(\hat{y}_t\) is determined with a decision threshold set to 0.5: if \(\hat{\pi}_{t|t-1} > 0.5\), then \(\hat{y}_t\) is 1, or 0 otherwise. Comparing the predicted labels with the true labels, the classifiers achieved performances of near the Bayes errors (top section of Table 6.1).

This data was further analysed by augmenting the input vector at time \(t\) with past inputs and past labels, as described in Section 6.3. Instead of the current input vector \(h_t\), we used \(i_{t,p} = [h_t^T, \ldots, h_{t-p}^T]^T\) or \(d_{t,p} = [h_t^T, \ldots, h_{t-p}^T, y_{t-1}, \ldots, y_{t-p}]^T\), where \(y_{t-k}\) represents an

![Figure 6.2: Two Gaussian distributions rotating in a circular fashion over time.](image)
observed label at time $t - k$. As the value of the look-back parameter $p$ we chose that which is the smallest among values that performed best. Comparing the classification performances we found that for the data with 4% Bayes error, classifiers using $i_{t,p}$ achieved the perfect classification, and for the data with 22% Bayes error, an accuracy improvement of $13.4\% \sim 15.9\%$. In addition, when the classifiers used $d_{t,p}$ as inputs, classification error was uniformly zero (middle and bottom sections of Table 6.1). These performance improvements are not surprising if we consider how the data is created: the input values rotate slowly over time and the true labels alternate between 0 and 1 every time step. These patterns within the inputs or the labels are implicitly captured by the augmented input vectors.

**Missing Labels**

In practice, it is common that we receive label information sparsely, i.e. labels are missing. We are required nevertheless to make a decision. An important feature of a dynamic model is that it can cope easily with missing observations. When class labels are not known, we may infer the missing label via the predictive distribution of the dynamic model. In this case we have

$$\tilde{y}_t = \Pr(y_t = 1|\mathbf{h}_t) = \pi_{t|t-1}$$  \hspace{1cm} (6.66)
### 6.4 Results

<table>
<thead>
<tr>
<th>Input vector</th>
<th>Classifier</th>
<th>0% Bayes error</th>
<th>4% Bayes error</th>
<th>22% Bayes error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_t$</td>
<td>DLR-E</td>
<td>0.9980</td>
<td>0.9750</td>
<td>0.7740</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>0.9990</td>
<td>0.9680</td>
<td>0.7930</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>0.9970</td>
<td>0.9650</td>
<td>0.7720</td>
</tr>
<tr>
<td>$i_{t,{4/6/3}}$</td>
<td>DLR-E</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9198</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9268</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9308</td>
</tr>
<tr>
<td>$d_{t,{2/4/8}}$</td>
<td>DLR-E</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 6.1: Proportion of correct classification. (Top) The current input vector $h_t$ was used by three classifiers, namely DLR-E, DLR-U and DBM. (Middle) The classifiers considered augmented inputs, $i_{t,p} = [h_t^T, h_{t-1}^T, \ldots, h_{t-p}^T]^T$. Here $i_{t,\{4/6/3\}}$ denotes that the value of $p$ for the data sets with 0%, 4% and 22% Bayes errors is 4, 6 and 3 respectively. (Bottom) Another augmented input vector, $d_{t,p} = [h_t^T, h_{t-1}^T, \ldots, h_{t-p}^T, y_{t-1}, \ldots, y_{t-p}]^T$, were used. The adaptive classifiers performed almost perfectly when augmented inputs were considered.

as a “quasi-target” in the place of a true target $y_t$. This approach allows us not to discard information contained within the input vector $h_t$ such as slow drift. However, as Lowne et al. (2008) pointed out, treating the quasi-targets as if they were true causes a risk of a classifier becoming excessively confident in predictions. To compensate for this illusory knowledge when lacking feedback, they proposed folding the uncertainty in the class probability predicted from the input alone into the state noise. We therefore update $q_t$ using

$$q_t = q + \hat{\pi}_{t|t-1}(1 - \hat{\pi}_{t|t-1}), \quad (6.67)$$

for the dynamic logistic regression based algorithms, and

$$q_t = q + \text{Var}(y_t|D_{t-1}), \quad (6.68)$$

for the dynamic binomial model. The quantity $q$ is a pre-set value for the stationary noise variance and $\text{Var}(y_t|D_{t-1})$ is the variance of the one-step ahead forecasting distribution given in Equation (6.55). We thereby ensure that the classifier does not become overly confident too quickly based on fictitious feedback.
We carried out experiments in which label information is successively randomly removed from 0% to 100% within each data stream. The performances of the classifiers were evaluated over 100 runs (Figure 6.4). We can see that the classification performances of the models did not worsen in proportion to the number of unobserved labels. The performances with 50% labelling were degraded by 0.13% ∼ 0.31% for the data with zero Bayes error, by 0.99% ∼ 1.17% for the data with 4% Bayes error, and by 0.6% ∼ 3.8% for the data with 22% Bayes error; with only 20% class labels available the classifiers’ performances were degraded by only 1.5% ∼ 5.2% in comparison with 100% labelling. This result indicates that an adaptive classifier can deal with sparsely observed labels and maintain its classification performance. This leads the way to enabling the adaptive classifier to decide for itself whether or not to request a label. This problem will be dealt with in the next section when we consider active label requesting.

![Figure 6.4](image)

**Figure 6.4:** Missing label experiments with variation in fractions of observed labels from 0% to 100% for the data sets with (a) 0%, (b) 4% and (c) 22% Bayes errors. The classifiers maintained their classification performances with up to 70% of labels missing.

### 6.4.2 Mountain Fire Scenario

We consider a fire in a wooded mountain with several villages in the nearby region (Figure 6.5). In order to allocate limited resources effectively we must classify if each of the villages
is in potential danger according to local weather conditions measured by weather sensors. For example, the stronger the wind speed is and the higher the air temperature is, the more likely it is for a village on the bottom left corner (denoted by a light blue ring) to be in danger. In addition, unless the wind blows in a northerly direction, the village is more vulnerable to the spread of fire. With weather data collected by a network of weather sensors (Chan, 2000), part of which was presented in Section 5.3.2, we created a data set consisting of three input variables (wind speed, wind direction and air temperature) and a binary class variable (‘danger’ or ‘no danger’).

![Figure 6.5: Mountain fire scenario: a fire starts in a wooded mountain with five villages (denoted by red dots), which are in potential danger according to local weather conditions.](image-url)

The manner in which inputs and target labels of the data are distributed means that no static classifier is able to separate the two groups (the left panel of Figure 6.6). The true labels alternate between ‘danger’ and ‘no danger’ at irregular intervals as the weather variables change (the right panel of Figure 6.6). For the sake of comparison, we ran a standard logistic regression on the data set; 100 input-output pairs were randomly taken from the full data set to use as a test set. The performance of this static model was averaged over 100 runs. When the original input vector, \( \mathbf{h}_t = [(\text{wind speed})_t; (\text{wind direction})_t; (\text{air temperature})_t] \), is used, our three adaptive classifiers significantly outperformed the static one, achieving an improvement of between 19.2% and 22.5% (Table 6.2). When the latest input and label were
incorporated into an augmented input vector, i.e. \(d_{t,1} = [h_t^T, h_{t-1}^T, y_{t-1}]^T\) (the look-back parameter \(p = 1\) was chosen because it achieved the lowest classification error), an improvement of classification accuracy was not significant for DLR-E and DBM. However, for DLR-U it achieved an improvement of 3.9%. As found in the experiment with the synthetic data in the previous section, when label information is successively randomly removed from 0% to 100% within the data stream, the classification performances of the models were not downgraded in proportion to the number of unobserved labels (Figure 6.7).

![Figure 6.6: (Left panel) A 3-D plot of the three input variables and the class variable. (Right panel) A time plot of the class variable. The true labels alternate between 'danger' and 'no danger' at irregular intervals as the weather variables change. Note that we deal with a single village represented by a light blue ring in Figure 6.5.](image)

**Active Label Requesting**

We here discuss the process of *active label requesting*. In practice observing labels may be expensive. Therefore, we can think of actively requesting a label; in other words, an adaptive classifier decides for itself when to request a label without severely deteriorating its classification performance. This problem is closely related to the problem of active data selection (Osborne et al., 2008b), as both methods use uncertainty as a guide to request information. At each time step a classifier makes a one-step ahead label prediction and if uncertainty as-
### 6.4 Results

<table>
<thead>
<tr>
<th>Input vector</th>
<th>Classifier</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_t$</td>
<td>Logistic Regression</td>
<td>0.7390</td>
</tr>
<tr>
<td></td>
<td>DLR-E</td>
<td>0.9636</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>0.9313</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>0.9623</td>
</tr>
<tr>
<td>$d_{t,1}$</td>
<td>DLR-E</td>
<td>0.9649</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>0.9703</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>0.9784</td>
</tr>
</tbody>
</table>

Table 6.2: Proportion of correct classification on the mountain fire data. (Top) The current input vector $h_t$ was used by four classifiers. Three adaptive models outperformed a static logistic regression. (Bottom) Adaptive classifiers used an augmented input vector, $d_{t,1} = [h_t^T, h_{t-1}^T, y_{t-1}]^T$.

Figure 6.7: Missing label experiment on the mountain fire data with variation in fractions of observed labels from 0% to 100%. The classification performances of the models did not worsen in proportion to the number of unobserved labels.
6.4 Results

Associated with the prediction is higher than a threshold, then it requests a label, otherwise, it proceeds without labelling. For dynamic logistic classifiers, the prediction uncertainty is $\hat{\pi}_t|_{t-1}(1 - \hat{\pi}_t|_{t-1})$ because a predicted label is assumed to have a Bernoulli distribution, and for the dynamic binomial model it is the variance of a beta-binomial predictive distribution, $\text{Var}(y_t|D_{t-1})$, given in Equation (6.55).

We carried out a set of experiments in order to obtain classification performances for different proportions of labelling from 100% to 30% (Table 6.3). With 50% labelling classification accuracy was degraded by 1.2% ~ 4.5% (Figure 6.8). In the figure the grey regions represent time steps when a classifier requested a label. We can see that when the models were confident enough about their predictions they did not request a label. Note that despite low classification error, the dynamic binomial model failed to detect the change of label in two occasions, and the dynamic logistic regression using the extended Kalman filter in one occasion.

<table>
<thead>
<tr>
<th>Labelling</th>
<th>DLR-E</th>
<th>DLR-U</th>
<th>DBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>0.9636</td>
<td>0.9313</td>
<td>0.9623</td>
</tr>
<tr>
<td>90%</td>
<td>0.9528</td>
<td>0.9299</td>
<td>0.9407</td>
</tr>
<tr>
<td>70%</td>
<td>0.9528</td>
<td>0.9259</td>
<td>0.9407</td>
</tr>
<tr>
<td><strong>50%</strong></td>
<td><strong>0.9394</strong></td>
<td><strong>0.9191</strong></td>
<td><strong>0.9178</strong></td>
</tr>
<tr>
<td>30%</td>
<td>0.9272</td>
<td>0.8935</td>
<td>0.7278</td>
</tr>
</tbody>
</table>

Table 6.3: Proportion of correct classification on the mountain fire data when a classifier actively requests labels. Despite requesting only half of labels, the classifiers maintained high classification accuracy.

Missing Inputs

It is not only a label but also an input that can be missing. For example, as a result of failure of weather sensors or failure of transmitting, weather condition data can be occasionally unavailable. Our decision models are defined by

\[
\pi_t = \ell(\varphi(h_t)^T w_t),
\]

(6.69)
Figure 6.8: Active label requesting on the mountain fire data when 50% of total labels is requested by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots. The grey regions represent time steps when the classifiers requested a label.
for the DLR models, or
\[ y_t \sim \text{Binomial with } \eta_t = \varphi(h_t)^\top w_t, \]  
(6.70)
for the DBM. Here \( \ell(\cdot) \) is the logistic function given in Equation (6.3) and \( \eta_t \) is the link equation given in Equation (3.11). When the input, \( h_t \), is not available, we can predict it as \( \hat{h}_t \). With a set of observed inputs up to time \( t - 1 \) (i.e. \( \{h_1, \ldots, h_{t-1}\} \)), we can predict the missing input using one of the multivariate forecasting models detailed in Chapter 5. The data streams of the three inputs in the mountain fire scenario (i.e. wind speed, wind direction, air temperature) are illustrated in Figure 6.9 (a). We removed at random 70% of the input data (Figure 6.9 (b)) and predicted the missing input values with a dynamic multivariate autoregressive model of AR order 2 (Figure 6.9 (c)). We can see that the predicted inputs are very similar to the true ones. To compensate for using these predicted inputs, we fold the uncertainty in the predicted values of missing inputs (i.e. the diagonal sum of the covariance matrix of \( \hat{h}_t \), which we denote by \( u_t \)) into the state noise. We therefore update \( q_t \) using
\[ q_t = q + u_t. \]  
(6.71)
The quantity \( q \) is a pre-set value for the stationary noise variance. To evaluate how adaptive classifiers perform when missing inputs, we randomly removed input information successively from 0% to 100% within the data. The performances of the classifiers were evaluated over 100 runs (Figure 6.10). The result of DLR-E stood out: with only 40% of inputs available it classified labels with a more than 90% accuracy.

Combined with the active label requesting, it is possible for a classifier to decide for itself which label to request even when input information is not available. We carried out an experiment in which 50% of inputs is missing and a classifier requests 50% of labelling. We computed the average proportion of correct classification and the corresponding standard deviation for the three adaptive classifiers over 10 runs: 0.8846/0.0229 (DLR-E), 0.7344/0.0221 (DLR-U) and 0.8274/0.0840 (DBM). In comparison to corresponding results when given complete input information provided in Table 6.3, we can see that for DLR-E classification accuracy was degraded by only 5%.
6.4 Results

Figure 6.9: A time plot of (a) true inputs, (b) given inputs (70% missing) and (c) predicted inputs. A dynamic multivariate autoregressive model of AR order 2 was used.

Figure 6.10: Missing input experiment on the mountain fire data with variation in fractions of observed inputs from 0% to 100%. The DLR-E model, in particular, performed very well with only 40% of inputs available.
6.4 Results

6.4.3 Experimental Data

We applied the adaptive classifiers detailed here to an on-line brain-computer interface experiment whose goal was to classify electroencephalogram (EEG) activity into a movement / non-movement label. Data used in this experiment consisted of two channels of EEG, recorded at 256Hz placed over the central portion of the head and one channel of muscle electrical activity (EMG), recorded at 1024Hz over the muscles of the right fore-arm. The EMG was then down-sampled to 256Hz and muscle contraction strength for movement and non-movement detection was evaluated via a simple windowed peak and trough detection; this then formed a movement / non-movement label. The second reflection coefficients of a second-order autoregressive model (Pardey et al., 1996) were calculated over each EEG signal once every 78ms using a sliding one-second-long window, forming a set of feature vectors $h_t$. This data description is extracted from Lowne et al. (2008).

We ran a static logistic classifier to compare with the adaptive ones. It significantly underperformed compared to the dynamic classifiers; DLR-E produced the best performance (top section of Table 6.4). The one-step ahead predictions of the class probability and the label predictions are illustrated in Figure 6.11. They are coloured blue and green respectively. The observed labels (coloured red) are also drawn for reference. We can find that when previously observed inputs and labels were incorporated into the current input variable (i.e. the look-back parameter $p = 1$), all the classifiers produced almost perfect performances (bottom section of Table 6.4). In addition, when we successively removed label information, the performances of the dynamic classifiers were not degraded in proportion to the number of unobserved labels (Figure 6.12). In the figure we can see that the variation in performances is robust to large numbers of unobserved labels: with only 50% class labels, DLR-E achieved a 96.6% accuracy, DLR-U 92.4%, and DBM 91.9%.

We consider how a classification behaves when requesting a label actively. We varied the value of the threshold in order to compute classification error for different proportions of labelling from 100% to 30% (Table 6.5). With 50% labelling our classifiers’ performances were degraded by only 0.7% $\sim 1.6%$. A more detailed description of this case is illustrated
6.4 Results

Figure 6.11: Classification predictions on the EEG data by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots.
## 6.4 Results

<table>
<thead>
<tr>
<th>Input vector</th>
<th>Classifier</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_t )</td>
<td>Logistic Regression</td>
<td>0.7550</td>
</tr>
<tr>
<td></td>
<td>DLR-E</td>
<td>\textbf{0.9742}</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>0.9532</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>0.9348</td>
</tr>
<tr>
<td>( d_{t,1} )</td>
<td>DLR-E</td>
<td>\textbf{0.9922}</td>
</tr>
<tr>
<td></td>
<td>DLR-U</td>
<td>0.9916</td>
</tr>
<tr>
<td></td>
<td>DBM</td>
<td>0.9920</td>
</tr>
</tbody>
</table>

Table 6.4: Proportion of correct classification on the EEG data. (Top) The current input vector \( h_t \) was used by one static and three dynamic classifiers. The adaptive models outperformed the static model. (Bottom) Adaptive classifiers used an augmented input vector, \( d_{t,1} = [h_T^T, h_{t-1}^T, y_{t-1}^T]^T \). The dynamic classifiers with this augmented input classified labels almost perfectly.

![EEG Data](image.png)

**Figure 6.12:** Missing label experiment on the EEG data with variation in fractions of observed labels from 0% to 100%. The classification performances of the models remained high even with a small fraction of labelled data points.
in Figure 6.13. Note that although the models decided for themselves when to request a label, they correctly detected irregular changes of labels and achieved high classification accuracies.

<table>
<thead>
<tr>
<th>Labelling</th>
<th>DLR-E</th>
<th>DLR-U</th>
<th>DBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>0.9742</td>
<td>0.9532</td>
<td>0.9348</td>
</tr>
<tr>
<td>90%</td>
<td>0.9685</td>
<td>0.9526</td>
<td>0.9306</td>
</tr>
<tr>
<td>70%</td>
<td>0.9639</td>
<td>0.9486</td>
<td>0.9276</td>
</tr>
<tr>
<td><strong>50%</strong></td>
<td><strong>0.9585</strong></td>
<td><strong>0.9455</strong></td>
<td><strong>0.9279</strong></td>
</tr>
<tr>
<td>30%</td>
<td>0.9518</td>
<td>0.9367</td>
<td>0.8438</td>
</tr>
</tbody>
</table>

Table 6.5: Proportion of correct classification on the EEG data when a classifier actively requests labels. Despite requesting only half of labels, the classifiers maintained high classification accuracy.

We carried out an experiment for missing input information. As presented in the previous section, input information is successively randomly removed from 0% to 100% (Figure 6.14). We can see that the performance of the dynamic logistic classifiers reached close to optimal even with a small fraction of given inputs; In particular, DLR-E achieved more than a 90% accuracy with 80% absence of input information. However, the dynamic binomial model did not perform consistently. In addition, we experimented a case in which a classifier actively requests 50% of labelling with 50% of inputs available. To evaluate how adaptive classifiers perform in this environment, we computed the average proportion of correct classification and the corresponding standard deviation over 10 runs: 0.9570/0.0030 (DLR-E), 0.9340/0.0079 (DLR-U) and 0.9104/0.0083 (DBM). Comparing these values with results given on the fourth row of Table 6.5, we can find that classification accuracy was degraded by only 0.1% ~ 1.7%.

In this chapter we have dealt with the problem of on-line binary classification using the dynamic logistic regression and the dynamic binomial model. In particular, we have investigated the case where input and/or label information is partially unobservable. The experimental results have shown that the models we considered performed well in such an incomplete environment. In addition, this success has led the way to active label requesting
Figure 6.13: Active label requesting on the EEG data when 50% of total labels is requested by (top) DLR-E, (middle) DLR-U and (bottom) DBM. The one-step ahead predictions for the class probability are represented by blue dots, the label predictions by green dots, and the observed labels by red dots. The grey regions represent time steps when the classifiers requested a label.
6.4 Results

Figure 6.14: Missing input experiment on the EEG data with variation in fractions of observed inputs from 0% to 100%. The DLR-E model outperformed the DLR-U and DBM models: its performance reaches close to optimal even with a small fraction of input data points.

in which an adaptive classifier decides for itself which label to observe.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

In many real-world data analysis problems, observations arrive sequentially in time and it is required to perform inference on-line. Sequential learning provides us with techniques to fuse information, learn policies, analyse risks, forecast outcomes and make decisions in such a way that a current model is updated as new information becomes available. This framework of sequential learning is particularly important in two aspects. Firstly, the environment we are faced with changes over time. This dynamic characteristic of data requires an adaptive model. Secondly, the observations are often partially or entirely missing and information about the environment is incompletely available. It is therefore necessary to develop methods that perform consistently on such occasions. An additional advantage of sequential learning methods is computational simplicity in the sense that they do not require a storage of all the data. This property allows us to deal with numerous practical problems in real time.

This thesis has focused on the problem of sequential forecasting, decision and action making in dynamic and incomplete environments. To solve this problem we have adopted a dynamic model framework. We have discussed five sub-classes of the dynamic model: (1) dynamic linear models, (2) dynamic non-linear models, (3) dynamic generalised linear models, (4) Markov decision processes and (5) delayed reward, reinforcement learning, models.

If data is modelled by a dynamic linear model, the Kalman filter provides a recursive optimal estimation method. Chapter 2 reviewed the theoretical background of the Kalman filter.
However, in many dynamic systems the state process and the observation process are non-linear. The extended Kalman filter is one of the most widely used methods for implementing an approximate non-linear Kalman filter. The underlying idea is to approximate a non-linear function by linearising it via the Taylor approximation. Recently, the unscented Kalman filter has been used in various applications from a navigation problem (van der Merwe and Wan, 2004) to a geophysical problem (Gove and Hollinger, 2006). This filter is different from other non-linear filters in that it does not try to approximate a non-linear function, but rather tries to approximate a probability distribution. We have proposed a novel non-linear filter named the Chebyshev Kalman filter. It is based on the fact that the Chebyshev approximation is known as the best approximation method to a function in a sense of minimax. The novel filter is similar to the extended Kalman filter in that it approximates a non-linear function.

The dynamic linear model generally assumes that the state and observation variables are Gaussian. However, there are numerous problems in which a Gaussian is not suitable. Examples include analysing count data (Poisson distribution), modelling the variance of a population (Gamma distribution), or describing the standardised price returns on individual stocks (Pareto distribution). Generalised linear models are a flexible generalisation of ordinary linear regression models and they are applicable to non-Gaussian distributions of the exponential family. Chapter 3 discussed the framework of dynamic generalised linear models, which allows us to sequentially analyse non-Gaussian data in a non-stationary environment.

We are often faced with a situation in which an agent and an environment interact continually: an action made by the agent affects the environment; the environment responds by giving a reward; and we are interested in an optimal policy that gives rise to the maximal total rewards over time. This is formulated as a Markov decision process. Chapter 4 dealt with Reinforcement Learning, which is a collection of sub-optimal methods for an incompletely known Markov decision process. We have discussed how the problem of continuous state and action spaces can be solved in the reinforcement learning framework. We have suggested
7.1 Conclusions

augmenting states with actions and estimating the value of the augmented states via a radial basis function model. It has been found that this novel approach significantly outperformed the widely-used discretisation-based method. In addition we have modified the traditional reinforcement learning algorithm in such a way that it is suitable for a non-stationary environment. Our proposed method has successfully solved a shortest-path-finding problem in a dynamic setting.

As the second task of sequential learning we have considered the problem of forecasting observations in incomplete multivariate time series. It is a challenging task because multiple sensors are correlated and their observations are occasionally unavailable, partially or completely. This is a common problem we are faced with in financial markets or weather forecasts. In Chapter 5 we discussed three multivariate forecasting models, namely a dynamic multivariate autoregressive model, a multivariate local trend model and a Gaussian process model. To compare the performances of the models we have applied them to air temperature data collected by a network of sensors. The data possess a daily cyclical pattern and sensor readings are occasionally missing. We have found that the Gaussian process model is a flexible forecasting tool as prior knowledge about the data can be incorporated in the form of a covariance function. Meanwhile, the dynamic linear model based methods have been found not to be suitable for predicting the periodic pattern of the data. To solve this problem we have suggested augmenting an observation vector with an “artificial” variable for the cycle. This simple idea allowed the dynamic multivariate autoregressive model to cope with the cyclical behaviour of air temperature. We have found that this approach produced as good a performance as the Gaussian process model while enjoying a significant computational advantage.

The problem of adaptive classification is another interesting sequential learning task. The binary classification problem is normally solved by a logistic regression model. To deal with non-stationarity of data a dynamic logistic regression using the extended Kalman filter has been studied in real-world domains such as brain-computer interface research (Yoon et al., 2008). In Chapter 6 we proposed a novel dynamic classification model that employs
the unscented Kalman filter. In addition, we have showed that a dynamic generalised linear model can be used as an adaptive classifier because binary classes follow a binomial distribution. We have applied these dynamic classification models to various incomplete problems. When class labels are missing we have found that the performance of the dynamic classifiers reached close to optimal even with a small fraction of label information. This result led the way to active label requesting in which an adaptive classifier decides for itself which label to observe. In addition we have investigated the case of missing occurrences of input information. The dynamic multivariate autoregressive model detailed in Chapter 5 has been combined with an adaptive classifier; we have obtained a successful result in which the classifier’s performance with a small fraction of input information was close to that with complete information.

Overall, sequential learning has three main characteristics that can be enjoyed in numerous real applications: on-line updating of information, capability to handle incomplete information and computational simplicity. This allows us to be equipped with a powerful tool when trying to discover hidden patterns in data for purposes such as risk management, forecasting, or decision making.

7.2 Future Extensions

An interesting application domain of sequential learning is quantitative finance. Financial instruments data possess the characteristics we have focused on: noisy, uncertain, correlated, dynamic and incomplete. Many systematic trading models are designed to predict how prices of the financial instruments move by past price movements. Discovering these trends as well as any patterns hidden within the prices is a key to the success of such models. Our dynamic multivariate forecasting models detailed in Chapter 5 can be used to predict price forecasts reflecting changes and movements in the markets. In particular, for the computational reason the dynamic multivariate autoregressive model may allow us to analyse financial data in real time.

Dynamic generalised linear models can also provide practically useful approaches to the
analysis of financial data; as many financial phenomena are known to follow non-Gaussian distributions in the exponential family. For example, suppose that we are interested in estimating the conditional variance of the log-return of the price of an asset (the log-return is defined by \( y_t = \log p_t - \log p_{t-1} \), where \( p_t \) is the price of the asset at time \( t \)); this problem of modelling the volatility of an asset is important in risk management and investment strategies. The generalised autoregressive heteroskedastic (GARCH) is a classical model for this problem in econometrics (Bollerslev, 1986); and it can be solved within the framework of the dynamic generalised linear model (Triantafyllopoulos, 2008). In addition to this, we can consider other problems associated with quantitative finance: the standardised price returns on individual stocks is known to follow a Pareto distribution (Reed, 2001); the quantification of risk in stocks has been investigated via the inverse Gaussian distribution (Bolviken and Benth, 2000). We are interested in learning these models in a sequential manner to capture non-stationarity of such data.

We have discussed the problem of adaptive binary classification in Chapter 6. We can consider extending the argument to the case of multi-class labels. Lowne et al. (2008) suggested an algorithm in which, given a set of \( K \) classes, we evaluate \((K - 1)\) two-class models which successively evaluate the probability that the true label is less than label index \( k \). However, in the context of the dynamic generalised linear model we can solve this problem straightforwardly with a multinomial distribution: the use of a Dirichlet distribution as a conjugate prior allows us to have a dynamic multi-class classifier. In addition we are not only interested in making a one-step ahead classification but also in making a decision at more than one-step into the future. This can be dealt with by forecasting unknown future inputs and then predicting future classifications. This problem can be useful in various disaster management scenarios; for instance, in the mountain fire case demonstrated in Section 6.4.2, it is important to forecast a future classification to allocate resources appropriately in the present and reduce potential future danger.

The adaptive classification problem and the reinforcement learning problem both belong to that of decision making. However, they are different in how a decision interacts
with an environment: in the former case the classification is made solely for a given input from the environment; whereas for the latter problem the decision affects the environment, the environment responds by returning a reward, and the decision is made to maximise the sum of rewards over time. This Markov decision process has long been studied in the optimal control literature. However, the traditional optimal control methods require the use of off-line backward recursion. Szita and Lorincz (2004) showed that a modification of the linear-quadratic-Gaussian Kalman filter model gives rise to an on-line estimation method for optimal control in the reinforcement learning framework. We are interested in extending this work to more general settings: non-linear, non-stationary and non-Gaussian. It is certainly a very challenging target to achieve.

Conclusively, our goal of mathematical data modelling is beautifully encapsulated by Devlin (1998): “Mathematics serves us by making the invisible visible.” We believe that there always exist hidden patterns waiting to be discovered. With better understanding of the environments surrounding us we may be able to explain a large part of currently inexplicable phenomenon. Development of more sophisticated mathematical techniques with increasing computational power will probably lead us one step closer to discovering the hidden patterns; and our desire for “making the invisible visible” will never diminish.
Appendix A

Derivations

A.1 Derivation of Equation (3.31)

The posterior distribution of the state variable can be represented as follows:

\[ p(w_t|D_t) = \int p(\eta_t, w_t|D_t) d\eta_t = \int \frac{p(\eta_t, w_t|D_{t-1})p(y_t|\eta_t)}{p(y_t|D_{t-1})} d\eta_t \]

\[ = \int \frac{p(w_t|\eta_t, D_{t-1})p(\eta_t|D_{t-1})p(y_t|\eta_t)}{p(y_t|D_{t-1})} d\eta_t \]

\[ = \int \frac{p(w_t|\eta_t, D_{t-1})p(\eta_t|D_{t-1}, y_t)p(y_t|D_{t-1})}{p(y_t|D_{t-1})} d\eta_t \]

\[ = \int p(w_t|\eta_t, D_{t-1})p(\eta_t|D_t) d\eta_t \quad \text{(A.1.1)} \]

A.2 Derivation of Equation (6.48): Conjugate prior of a binomial

The conjugate prior for a binomial distribution is of form

\[ p(\theta_t|D_{t-1}) = \omega(k_t, m_t) \exp(k_t\theta_t - m_t \log(1 + \exp(\theta_t))). \quad \text{(A.2.2)} \]
A.2 Derivation of Equation (6.48): Conjugate prior of a binomial

Using \( \theta_t = \log \left( \frac{\pi_t}{1 - \pi_t} \right) \), the density function of \( \pi_t \) is of form

\[
p(\pi_t|D_{t-1}) = \frac{\mathrm{d}}{\mathrm{d}\pi_t} \log \left( \frac{\pi_t}{1 - \pi_t} \right) = \omega(k_t, m_t) \pi_t^{k_t - 1} (1 - \pi_t)^{m_t - k_t - 1}.
\]

(A.2.3)

This is a beta distribution with parameters \( k_t \) and \( m_t - k_t \), thereby we know that \( \omega(k_t, m_t) = \frac{\Gamma(m_t)}{\Gamma(k_t) \Gamma(m_t - k_t)} \).

The mean and variance of the prior distribution are computed in the following way: the moment generating function of \( \theta_t \) is

\[
M_{\theta_t}(z) = \mathbb{E}(\exp(z\theta_t)) = \int_{-\infty}^{\infty} \exp(z\theta_t)p(\theta_t|D_{t-1})\mathrm{d}\theta_t = \frac{\Gamma(k_t + z) \Gamma(m_t - k_t - z)}{\Gamma(k_t) \Gamma(m_t - k_t)}, \tag{A.2.4}
\]

and thus the cumulant generating function is

\[
C_{\theta_t}(z) = \log(M_{\theta_t}(z)) = \log \Gamma(k_t + z) - \log \Gamma(m_t - k_t - z) - \Gamma(k_t) - \Gamma(m_t - k_t) \tag{A.2.5}
\]

From this cumulant generating function two moments of the conjugate prior can be computed:

\[
\mathbb{E}(\theta_t|D_{t-1}) = \left. \frac{\mathrm{d}}{\mathrm{d}z} C_{\theta_t}(z) \right|_{z=0} = \psi(k_t) - \psi(m_t - k_t) \approx \log \left( \frac{k_t}{m_t - k_t} \right), \tag{A.2.6}
\]

\[
\operatorname{Var}(\theta_t|D_{t-1}) = \left. \frac{\mathrm{d}^2}{\mathrm{d}z^2} C_{\theta_t}(z) \right|_{z=0} = \psi'(k_t) + \psi'(m_t - k_t) \approx \frac{1}{k_t} + \frac{1}{m_t - k_t}, \tag{A.2.7}
\]

where \( \psi(\cdot) \) is the digamma function defined by

\[
\psi(x) = \frac{\mathrm{d}}{\mathrm{d}x} \log \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}.
\]

The digamma function and its derivative are approximated by \( \log(x) \) and \( 1/x \) respectively (Abramovitz and Stegun, 1965).
A.3 Derivation of Equation (6.53): One-step forecast distribution of a binomial

When an observation variable $y_t$ follows a binomial distribution, the one-step ahead forecast distribution is computed as follows:

$$p(y_t|D_{t-1}) = \int_{-\infty}^{\infty} p(y_t|\theta_t)p(\theta_t|D_{t-1})d\theta_t,$$

$$= \int_{-\infty}^{\infty} \exp \left( n \left[ \frac{y_t}{n} - \log(1 + \exp(\theta_t)) \right] + \log \left( \frac{n}{y_t} \right) \right) \frac{\Gamma(m_t)}{\Gamma(k_t)\Gamma(m_t - k_t)} \frac{\exp(k_t\theta_t)}{(1 + \exp(\theta_t))^{m_t}}d\theta_t$$

$$= \frac{\Gamma(m_t)}{\Gamma(k_t)\Gamma(m_t - k_t)} \left( \frac{n}{y_t} \right) \int_{-\infty}^{\infty} \exp((k_t + y_t)\theta_t) \frac{\Gamma(k_t + y_t)}{\Gamma(m_t + n)} \left( \frac{n}{y_t} \right) \frac{\Gamma(m_t - k_t + n - y_t)}{\Gamma(m_t + n)}.$$  (A.3.8)
Bibliography


Kawata, Y., Tsuji, Y., Sugimoto, Y., Hayashi, H., Matsutomi, H., Okamura, Y., Hayashi, I.,
Kayane, H., Tanioka, Y., Fujima, K., Imamura, F., Matsuyama, M., Takahashi, T., Maki,
coastal zones by the 2004 Indian Ocean tsunami disaster. Disaster Prevention Research

Pub. Co.

Levinson, N. (1947). The Wiener RMS (Root Mean Square) Error Criterion in Filter Design

fication. *Machine Learning, Submitted*.


tions on*, 17(5):693–698.

preparation, University of Oxford.


