Model-based understanding of facial expressions

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Medical and Human Sciences

2013

Patrick Sauer
School of Medicine
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>2</td>
</tr>
<tr>
<td>Abstract</td>
<td>6</td>
</tr>
<tr>
<td>Declaration</td>
<td>7</td>
</tr>
<tr>
<td>Copyright</td>
<td>8</td>
</tr>
<tr>
<td>List of Tables</td>
<td>10</td>
</tr>
<tr>
<td>List of Figures</td>
<td>11</td>
</tr>
<tr>
<td>Glossary</td>
<td>12</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>14</td>
</tr>
<tr>
<td>1.1 Outline</td>
<td>16</td>
</tr>
<tr>
<td>2 Active Appearance Models</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Procrustes Analysis</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Texture Warping</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Piecewise Affine Warps</td>
<td>19</td>
</tr>
<tr>
<td>2.2.2 Barycentric Coordinates</td>
<td>21</td>
</tr>
<tr>
<td>2.3 Dimensionality Reduction</td>
<td>23</td>
</tr>
<tr>
<td>2.3.1 Principal Component Analysis</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Linear Shape Model</td>
<td>25</td>
</tr>
<tr>
<td>2.5 Linear Texture Model</td>
<td>25</td>
</tr>
<tr>
<td>2.6 Linear Appearance Model</td>
<td>26</td>
</tr>
<tr>
<td>2.7 Fitting Generative AAMs</td>
<td>27</td>
</tr>
<tr>
<td>2.7.1 Objective Function</td>
<td>27</td>
</tr>
</tbody>
</table>
## CONTENTS

2.8 Combined AAM ................................................. 29  
2.9 Simultaneous Inverse Compositional AAM ....................... 30  
2.10 Project-Out Inverse Compositional AAM ........................ 32  
2.11 Conclusions ............................................... 33  

3 Discriminative AAM 34  
3.1 Features .................................................. 35  
  3.1.1 Haar Features ........................................... 35  
  3.1.2 Steerable Pyramid Features ............................. 37  
3.2 Regression Models ........................................... 39  
  3.2.1 Boosting ................................................. 40  
  3.2.2 Random Forest ............................................ 41  
  3.2.3 Gaussian Processes ...................................... 43  
  3.2.4 Sparse Gaussian Process Regression ....................... 46  
3.3 Sequential Discriminative Models ................................. 49  
  3.3.1 Coupled Sequential AAM .................................. 50  
  3.3.2 Independent Sequential AAM .............................. 50  
3.4 Experiments ................................................. 51  
  3.4.1 Error Measure ............................................. 51  
  3.4.2 Train/Test Dataset Generation ............................ 52  
  3.4.3 Generative AAMs ......................................... 53  
  3.4.4 Discriminative AAMs ..................................... 53  
  3.4.5 Implementation .......................................... 55  
  3.4.6 Discussion .............................................. 59  
3.5 Conclusions ................................................. 62  

4 Non-Linear Probabilistic Models for Face Modelling 64  
4.1 Probabilistic Unsupervised Dimensionality Reduction ............ 64  
4.2 Gaussian Process Latent Variable Model .......................... 66  
4.3 Generative AAM with GPLVM .................................. 67  
  4.3.1 GPLVM Texture Model ...................................... 69  
  4.3.2 Texture generation ....................................... 72  
  4.3.3 Latent space projection ................................... 73  
  4.3.4 Non-Linear AAM .......................................... 73  
  4.3.5 Experiments .............................................. 74
## 5 Supervised dimensionality reduction using Gaussian processes

### 5.1 Supervised Variational GPLVM

### 5.2 Variational inference

### 5.3 Classification

### 5.4 Complexity

### 5.5 Emotion Recognition with Supervised Variational GPLVMs
- **5.5.1 Cohn-Kanade Dataset**
- **5.5.2 Features**
- **5.5.3 Models**
- **5.5.4 Results**
- **5.5.5 Comparison**

### 5.6 Implementation

### 5.7 Conclusions

## 6 Conclusions and Future Work

### 6.1 Discriminative AAMs

### 6.2 Non-Linear AAM

### 6.3 Supervised GPLVM

## A Similarity Transforms

### A.1 Calculating transform parameters
- **A.1.1 Translation**
- **A.1.2 Scale and Rotation**

### A.2 Composition

### A.3 Inverse

## B Matrix Identities and Linear Algebra

### B.1 Matrix product

### B.2 Hadamard product

### B.3 Trace

### B.4 Cholesky Factorisation

### B.5 Spectral Decomposition

### B.6 Partitioned Inverse

### B.7 Matrix Inversion Lemma

### B.8 Matrix Determinant Lemma

### B.9 Matrix Derivatives
Abstract

Model-based understanding of facial expressions

Patrick Sauer
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2013

In this thesis we present novel methods for constructing and fitting 2d models of shape and appearance which are used for analysing human faces. The first contribution builds on previous work on discriminative fitting strategies for active appearance models (AAMs) in which regression models are trained to predict the location of shapes based on texture samples. In particular, we investigate non-parametric regression methods including random forests and Gaussian processes which are used together with gradient-like features for shape model fitting. We then develop two training algorithms which combine such models into sequences, and systematically compare their performance to existing linear generative AAM algorithms. Inspired by the performance of the Gaussian process-based regression methods, we investigate a group of non-linear latent variable models known as Gaussian process latent variable models (GPLVM). We discuss how such models may be used to develop a generative active appearance model algorithm whose texture model component is non-linear, and show how this leads to lower-dimensional models which are capable of generating more natural-looking images of faces when compared to equivalent linear models. We conclude by describing a novel supervised non-linear latent variable model based on Gaussian processes which we apply to the problem of recognising emotions from facial expressions.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

I would like to thank my supervisor Chris Taylor for his encouragement, patience and generous support throughout the last four years. He has been a consistent source of advice and enthusiasm. During this time I have been co-supervised by Tim Cootes, who has given me much insight into the technicalities of active appearance models. I am deeply grateful to him for granting me access to his vast collection of C++ software which includes contributions by Phil Tresadern, Martin Roberts, Angela Caunce and many others. My thanks also go to my advisor, Neil Thacker, for sharing his views in many interesting conversations.

Furthermore, I am very grateful for the support I have received from Michalis Titsias, Andreas Damianou and Neil Lawrence, who kindly allowed me to use their Matlab and C++ software, and who introduced me to the concept of Gaussian process latent variable models. I am particularly indebted to Michalis Titsias, without whose patience, expert advice and support I could not have written the latter chapters of this thesis.

I would like to thank my parents for their unstinting emotional and financial support throughout my education. Their care and understanding have helped me face the many ups and downs I have encountered along the way.

Finally, I would like to thank “Es Ratolinet” for all her kindness and patience during the preparation of this thesis. I promise, it’s finished now!
List of Tables

3.1 Settings for training AAM sequences. ........................................ 53
3.2 Model-fitting experiments using the small displacement CogSys test dataset. 56
3.3 Model-fitting experiments using the large displacement CogSys test dataset. 57
3.4 Cross-dataset fitting experiments using the BioID and XM2VTS datasets 
   and independent sequential AAMs with equal number of pixels. ......... 58
3.5 Timings for the training and testing of generative and discriminative AAMs. 59
3.6 Cumulative results for independent sequential AAMs on CogSys dataset. 59
4.1 Fitting results for L-SIC, G-SIC and G-OptSIC AAMs on CogSys dataset. 76
4.2 Timings per iteration for L-SIC, G-SIC and G-OptSIC algorithms. .... 77
5.1 Emotion recognition results on Cohn-Kanade dataset using variational 
   GPLVM algorithms. ............................................................... 94
5.2 Emotion recognition results on Cohn-Kanade dataset using related meth- 
   ods. ...................................................................................... 94
List of Figures

2.1 Thin-plate splines and piecewise affine warps. ........................................ 20
2.2 Area coordinates. ....................................................................................... 21
2.3 Generative AAM, residual image generation. .............................................. 28

3.1 Shape model fitting using a discriminative AAM algorithm. ....................... 35
3.2 Haar features and the integral image. .......................................................... 36
3.3 Haar feature vectors. .................................................................................. 36
3.4 Image pyramids. .......................................................................................... 37
3.5 Steerable pyramid features. ......................................................................... 39
3.6 Piecewise constant weak learner. ............................................................... 40
3.7 Gaussian process regression. ...................................................................... 44
3.8 Sequential AAM training. .......................................................................... 50
3.9 CogSys dataset. .......................................................................................... 51
3.10 XM2VTS and BioID datasets. ................................................................... 52

4.1 Latent space visualisation of fitting experiment on an image from the CogSys dataset using G-SIC, G-OptSIC and L-SIC AAM with 2D texture models. ........................................................................ 78
4.2 Textures after fitting 10D texture models to an image from the CogSys dataset using L-SIC, G-SIC and G-OptSIC algorithms. ................................................................. 79

5.1 Graphical models of Bayesian GPLVMs. .................................................... 83
5.2 Cohn-Kanade dataset. ................................................................................ 91
5.3 Inverse lengthscales of RBF-ARD kernel and latent space obtained from emotion recognition experiment using Cohn-Kanade dataset and a supervised GPLVM. ...................................................... 96
5.4 Prior covariance matrices $K_Y$ obtained from emotion recognition experiments using Cohn-Kanade dataset. ................................................................. 96
## Glossary

### Acronyms

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAM</td>
<td>Active Appearance Model</td>
</tr>
<tr>
<td>ARD</td>
<td>Automatic Relevance Determination</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian Process</td>
</tr>
<tr>
<td>GPLVM</td>
<td>Gaussian Process Latent Variable Model</td>
</tr>
<tr>
<td>IOD</td>
<td>Inter-Ocular Distance</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum a-posteriori</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>POIC</td>
<td>Project-Out Inverse-Compositional</td>
</tr>
<tr>
<td>PPCA</td>
<td>Probabilistic Principal Component Analysis</td>
</tr>
<tr>
<td>RF</td>
<td>Random Forest</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>SIC</td>
<td>Simultaneous Inverse-Compositional</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>TPS</td>
<td>Thin-Plate Spline</td>
</tr>
</tbody>
</table>
## Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>A scalar.</td>
</tr>
<tr>
<td>$\mathbf{a}$</td>
<td>A vector.</td>
</tr>
<tr>
<td>$A$</td>
<td>A matrix.</td>
</tr>
<tr>
<td>$A^T$</td>
<td>The transpose of matrix $A$.</td>
</tr>
<tr>
<td>$A^{-1}$</td>
<td>The inverse of matrix $A$.</td>
</tr>
<tr>
<td>$I$</td>
<td>The identity matrix.</td>
</tr>
<tr>
<td>$L_A$</td>
<td>The lower triangular Cholesky factor of the symmetric, positive definite matrix $A$.</td>
</tr>
<tr>
<td>$X$</td>
<td>$\mathbb{R}^{N \times Q}$ matrix whose rows contain latent vectors ${x_n}_{n=1}^N$.</td>
</tr>
<tr>
<td>$Y$</td>
<td>$\mathbb{R}^{N \times D}$ matrix whose rows contain data vectors ${y_n}_{n=1}^N$.</td>
</tr>
<tr>
<td>$Z$</td>
<td>$\mathbb{R}^{N \times K}$ matrix whose rows contain label vectors ${z_n}_{n=1}^N$ in 1-of-$K$ encoding.</td>
</tr>
<tr>
<td>$</td>
<td>a</td>
</tr>
<tr>
<td>$</td>
<td>A</td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$\bar{a}$</td>
<td>The sample mean $\frac{1}{N} \sum_{i=1}^N a_i$ of a set of vectors $a_1, \ldots, a_N$.</td>
</tr>
<tr>
<td>$\mathbb{E}[x]$</td>
<td>Expectation of a random variable $x$, $\int x p(x) dx$.</td>
</tr>
<tr>
<td>$\langle \cdot \rangle_{p(x)}$</td>
<td>Expectation of expression $\cdot$ with respect to probability density $p(x)$, $\int \cdot p(x) dx$.</td>
</tr>
<tr>
<td>$\delta_{ij}$</td>
<td>Kronecker’s delta: $\delta_{ij} = 1$, iff $i = j$, 0 otherwise.</td>
</tr>
<tr>
<td>$G(r)$</td>
<td>Pixel value at position $r$ in image $G$.</td>
</tr>
<tr>
<td>$\nabla G(r)$</td>
<td>Gradient $\left( \frac{\partial G(r)}{\partial x}, \frac{\partial G(r)}{\partial y} \right)$ at pixel location $r$ in image $G$.</td>
</tr>
<tr>
<td>$\sum_{i=1}^N$</td>
<td>Sum from $i = 1$ to $i = N$.</td>
</tr>
<tr>
<td>$\prod_{i=1}^N$</td>
<td>Product from $i = 1$ to $i = N$.</td>
</tr>
<tr>
<td>$1.9(\Delta)$</td>
<td>The error $\Delta$ is to be read as a multiple of the last digit location in the number, e.g. in this case $1.9 \pm 0.1 \Delta$.</td>
</tr>
<tr>
<td>$n_s$</td>
<td>Number of shape eigenvectors retained in linear shape model.</td>
</tr>
<tr>
<td>$n_t$</td>
<td>Number of texture eigenvectors retained in linear texture model.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In order to facilitate communication between individuals, humans are equipped with a multiplicity of channels through which they interact with the outside world. Examples are the auditory channels of speech and intonation and the visual channel that carries facial expressions [Pantic and Rothkrantz, 2000]. When analysing the channels used in interpersonal communication, the appearance of the face has been found repeatedly to be the most important channel for conveying emotional state and intent during conversation [Stephenson et al., 1978]. In fact, on an emotional level, facial expressions appear to have considerably more effect on the listener than the words spoken [Mehrabian, 1968; Pantic and Rothkrantz, 2000]. The established importance of facial expressions in human interaction has stimulated research in many fields, including psychology, where Ekman and Friesen [1978] set out to create a taxonomy of the facial expressions relating to the basic emotions of fear, anger, happiness, sadness, surprise and disgust by investigating their origin in the action of different facial muscles. Although it is uncertain whether the full space of facial representations of human emotions is spanned by these six prototypical expressions, there is an increasing body of evidence that shows the facial expressions associated with these emotions to be universal among humans [Wagner and Manstead, 1989]. The fact that humans rely so strongly on facial expressions to convey information in social interactions partly explains why many people find the day to day use of computers difficult. Despite extraordinary advances in computational power, the human-computer interface has barely evolved in decades and even today makes no use of non-verbal cues provided by the user. Thus, a central aim of computer vision research is the development of systems that understand human visual cues and which are capable of responding by generating images of faces that con-
vey emotional intent. The successful development of such systems requires three major problems to be solved [Pantic and Rothkrantz, 2000]:

1. detecting the face in images
2. extracting and representing the face
3. classifying the facial expression.

While the work of Viola and Jones [2001] is the canonical solution for the first problem, and a plethora of Machine Learning solutions exist for approaching the last problem, the most challenging task lies in finding a mathematical representation which may be accurately and efficiently extracted from images, is expressive enough to capture the subtleties of facial expressions, and from which images of facial expressions may be synthesised. The most successful attempt at providing such a representation is given by a group of algorithms referred to as “active appearance models” (AAMs) [Cootes et al., 2001] which represent the face in terms of parametric statistical models of shape and texture and provide algorithms for efficiently determining the best-fit model parameters from the image data. Nevertheless, the original formulation of AAMs has drawbacks which include lacking robustness under change of identity, lighting variation and occlusion. Furthermore, linear models are used to capture the non-linear processes that underlie the generation of the face image. Therefore, subsequent research has focused on rectifying these issues and has mainly followed two routes. In the first, the scope of the representation is reduced by dropping the ability to generate images and the focus is placed on developing more robust, highly efficient fitting algorithms for the linear shape models. In recent years several research contributions have taken this approach [Saragih and Goecke, 2007; Sauer et al., 2011; Tresadern et al., 2010; Zhou and Comaniciu, 2007], which we refer to as the “discriminative” AAM\(^1\). A related discriminative approach which has seen recent development is referred to as the “constrained local model” and was introduced by Cristinacce and Cootes [2006] and further explored by Cootes et al. [2012]; Saragih et al. [2011]. The second approach maintains the generative nature of the representation, but aims at increasing its expressiveness by replacing the statistical models employed. This “generative” AAM has seen less development over recent years, with the most notable contribution the work on multilinear models by Lee and Kim [2009].

The work presented in this thesis contains novel contributions to both the “discriminative” and “generative” AAM approaches outlined above. The first contribution

\(^1\)The term “shape” AAM has also been used, but we prefer the inclusion of the term “discriminative”, since it describes the class of fitting algorithms used in this context.
discusses the application of previously unused non-linear regression strategies in the
discriminative AAM setting and discusses the use of sequences of models to improve
performance. Following this, we focus our attention on adding non-linearity into the
generative AAM framework by investigating the use of Gaussian process latent variable
models (GPLVMs) [Lawrence, 2005]. Within this context, we present a non-linear AAM
algorithm which results in lower-dimensional, more natural-looking representations of
face images when compared to the original linear AAM. We conclude the thesis by pre-
senting a novel supervised Gaussian process latent variable model algorithm which we
apply to the problem of facial expression recognition.

1.1 Outline

Chapter 2 gives a review of the concepts used for building statistical models of shape
and texture and presents the most commonly cited generative AAM algorithms before
discussing their benefits and drawbacks.

Chapter 3 presents discriminative AAMs and discusses different regression algo-
rithms and features which are used for fitting such models. Two sequential training
algorithms are introduced and their performance in combination with the different dis-
criminative AAMs is evaluated and discussed based on a set of model-fitting experi-
ments.

Chapter 4 gives a formal introduction to Gaussian process latent variable models
(GPLVMs) which may be viewed as a non-linear type of principal component analysis
(PCA). Following this, a novel generative AAM algorithm is developed in which a non-
linear GPLVM replaces the more typically used linear texture model. The results of
several qualitative and quantitative experiments are presented and discussed.

Chapter 5 extends previous work by incorporating a novel prior distribution and
developing a method that allows the use of GPLVMs in supervised learning problems.
The resulting models are evaluated and discussed based on a set of experiments in which
we attempt to classify the seven basic human emotions surprise, anger, fear, contempt,
disgust, sadness and happiness using images from the Cohn-Kanade facial expression
database.

Chapter 6 contains a summary of the work presented in the thesis and concludes
with some ideas for further work.
Chapter 2

Active Appearance Models

The term “active appearance model” (AAM) was coined by Cootes et al. [2001] and refers to the combination of a statistical model which captures the appearance of a category of objects and a fitting algorithm which allows the determination of the best-fit model parameters given a digital image from the category. In the original work of Cootes et al. [2001], the statistical models are linear and the term “appearance” is understood to refer to the combination of an object’s shape and texture. The starting point for the statistical analysis of the shape and texture of a group of 2D objects is the provision of a dataset

\[ T = \{(I_1, P_1), \ldots, (I_N, P_N)\}, \]  

(2.1)

which contains images \(I_i\) and landmark annotations

\[ P_i = \{(x_{i1}, y_{i1}), \ldots, (x_{in}, y_{in})\}, \]  

(2.2)

which consist of \(n\) points which are placed at locations that correspond across objects of the category [Dryden and Mardia, 1998]. Given the dataset \(T\), the term “texture” is understood to refer to a set of pixels sampled from a uniform grid which covers the convex hull of the landmark annotations in the image. The canonical definition of “shape” is that of Kendall [1984], where it is defined as the geometric information of an object that is invariant with respect to the group of similarity transforms. In other words, the shape is the geometric information that remains after accounting for translation, rotation and scaling of the landmark points. Thus, in order to be able to compare the shapes of a set of objects, the landmark annotations must be aligned into a common reference frame using similarity transforms whose parameters define the pose of
the objects. The determination of the normalised landmark point set \( P_{\text{ref}} \) which defines this reference frame is a key step in the construction of the “model” components of linear active appearance models. Although analytic methods exist for two-dimensional shapes [Horn, 1987], in the context of active appearance models it is more common to use generalised Procrustes analysis [Dryden and Mardia, 1998], which we discuss in the following section. After discussing texture warping, we then introduce the concept of a “model” before moving on to describing the models and fitting algorithms introduced by Cootes et al. [2001] and subsequent developments.

2.1 Procrustes Analysis

In generalised Procrustes analysis, the point set \( P_{\text{ref}} \) which defines the shape reference frame is determined using a method that iteratively removes scaling, rotation and translation from the collection of landmark annotations by following the procedure shown in Algorithm 1. The algorithm is initialised by assigning the first landmark point set to \( P_{\text{ref}} \). The Procrustes mean \( P_{\text{ref}} \) is then iteratively refined by repeating the steps of aligning all landmark point sets in \( P \) to \( P_{\text{ref}} \) before assigning the mean of the aligned landmark point set collection \( P \) to \( P_{\text{ref}} \). The procedure terminates when the Euclidean distance between the new and the old reference point sets (Error) falls below a minimum threshold \( \epsilon \). The rigid transform used for aligning the 2D point sets is a similarity transform, for which we will use the same parameterisation as in [Matthews and Baker, 2004]. The mathematical details relating to the application of similarity transforms to landmark point sets are described in Appendix A.
2.2 Texture Warping

In order to be able to deal with textures independently from their associated shapes, it is necessary to define a separate texture reference frame. Typically, this is created by rescaling the shape reference frame until it covers a pre-specified number of pixels. The correspondence between the landmark points in the image and in the texture model reference frame implies a warp field that contains both rigid and non-rigid components. In practice, the most commonly used methods for implementing such warps are thin-plate splines [Bookstein, 1997] and piecewise affine warps [Wolberg, 1994], with the help of which textures are sampled from the images into the texture reference frame. Thin-plate splines use the location of the source and target points to set up a warp field that minimises an energy measure analogous to the bending energy of a clamped, thin steel plate. Solving the resulting minimisation problem yields a warp function that splits into an affine and a non-affine component. The non-affine component is generally infinitely differentiable and produces smooth warps. The smoothness of thin-plate splines allows non-linear deformations to be easily performed and thin-plate splines are therefore widely used in computer graphics. However, the advantages of thin-plate splines come at the cost of solving the energy minimisation problem each time the source or target points of the warp change, which is the case when fitting active appearance models. Furthermore, numerical issues may appear in extreme cases which can lead to unpredictable results, as shown in Figure 2.1. Here, the texture model reference frame on the left is defined by a closed-mouth shape whereas the image contains a wide-open mouth. The textures are sampled to the texture reference frame using both piecewise affine and thin-plate spline warps. As is obvious from the resulting textures, in this extreme case the piecewise affine warp leads to a more predictable result. Although more robust spline warping methods have been developed [Rohr et al., 2001], the computationally more efficient, continuous but less smooth piecewise affine warps are more commonly used for active appearance models [Cootes et al., 2001; Matthews and Baker, 2004], a convention which we follow in this thesis.

2.2.1 Piecewise Affine Warps

The starting point for setting up a piecewise affine warp is the Delaunay triangulation of the landmark annotations [Delaunay, 1934; Fortune, 1987; Guibas and Stolfi, 1985].

\footnote{For reasons of efficiency, in the context of active appearance models the number of pixels is often chosen to be much lower than the total number of pixels contained in the convex hull of the landmark points.}
Figure 2.1: Extreme case comparison of thin-plate splines (TPS) and piecewise affine warps. The image on the left shows the triangulated texture reference points superimposed over a regular grid. The two images in the middle show the deformations in the grid caused by computing the piecewise affine and thin-plate spline warps between the corresponding landmark points in the texture and image reference frames. Backwards-sampling the image pixels into the texture reference frame using the computed warps yields the images on the right. It is obvious from the result how unpredictable thin-plate spline warps can be in extreme cases.
Figure 2.2: The barycentric coordinates \((\lambda_1, \lambda_2, \lambda_3)\) of point \(r\) are defined by the ratios of the areas of the triangles inscribed around \(r\) with respect to the full triangle area. Affine invariance follows immediately from the invariance of the area ratios under affine transformations.

Given the triangulation, the dense correspondence between coordinates in the model reference frame and the image is achieved by moving to barycentric coordinates which are invariant with respect to affine transformations.

### 2.2.2 Barycentric Coordinates

Following Bradley [2007], given a triangle \(T\), the coordinates of any point \(r\) in the plane defined by \(T\) may be expressed as a convex linear combination of the triangle vertices \(r_1, r_2, r_3\), i.e.

\[
\begin{align*}
\mathbf{r} &= \lambda_1 \mathbf{r}_1 + \lambda_2 \mathbf{r}_2 + \lambda_3 \mathbf{r}_3 \\
\text{s.t.} & \quad \lambda_1 + \lambda_2 + \lambda_3 = 1
\end{align*}
\]  

(2.3)

Noting that \(\lambda_3 = 1 - \lambda_1 - \lambda_2\), this may be rewritten in matrix form as

\[
\mathbf{M} \lambda = \mathbf{r} - \mathbf{r}_3,  
\]  

(2.4)

where

\[
\mathbf{M} = \begin{pmatrix}
x_1 - x_3 & x_2 - x_3 \\
y_1 - y_3 & y_2 - y_3
\end{pmatrix},
\]  

(2.5)
and \( \{(x_i, y_i)\}_{i=1}^3 \) are the Cartesian coordinates of the triangle vertices. The inverse of \( M \) is easily computed using Cramer’s rule [Strang, 1988], so that the barycentric coordinates \( \lambda \) of \( r \) are given by

\[
\lambda = M^{-1}(r - r_3),
\]

where

\[
M^{-1} = \frac{1}{|M|} \begin{pmatrix} y_2 - y_3 & x_3 - x_2 \\ y_3 - y_1 & x_1 - x_3 \end{pmatrix}
\]

so that

\[
\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}
\]

(2.8)

\[
\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}
\]

(2.9)

\[
\lambda_3 = 1 - \lambda_1 - \lambda_2.
\]

(2.10)

Given the values of the barycentric coordinates, it is easy to determine whether \( r \) lies inside, on an edge, on a vertex or outside \( T \). In fact, we have

\[
\begin{align*}
\text{inside} & \quad T, \text{ iff } \lambda_i \in (0, 1), \ i \in \{1, 2, 3\} \\
\text{on an edge of} & \quad T, \text{ if } \exists! i : \lambda_i = 0, \ i \in \{1, 2, 3\} \\
\text{on a vertex of} & \quad T, \text{ if } \exists i, j, i \neq j : \lambda_i = 0, \lambda_j = 0, \ i, j \in \{1, 2, 3\} \\
\text{outside} & \quad T, \text{ otherwise.}
\end{align*}
\]

(2.11)

This is easily verified using the area interpretation of barycentric coordinates shown in Figure 2.2. Given the machinery provided by barycentric coordinates, the piecewise affine warp and the sampling shown in Figure 2.1 are easily implemented. For every point in the texture reference frame, we determine the identity of the triangle within which it lies, as well as its barycentric coordinates. Since the barycentric coordinates are invariant to affine transformations, sampling is performed by finding the corresponding triangle in the image and recovering the Cartesian point coordinates from the barycentric coordinates. The associated pixel value is then obtained by bilinear interpolation and written back to the texture reference frame. Since the texture reference frame is static, the sampling procedure may be made more efficient by creating a lookup table.
that associates each pixel in the texture reference frame with its enclosing triangle and barycentric coordinates.

2.3 Dimensionality Reduction

In the context of statistical shape analysis, the observed data is given by a set of Procrustes-aligned landmark annotations. Assuming a category of 2D shapes is annotated using $n$ landmark points, an instance of the category is mathematically described by a $2n$-dimensional vector of real values. However, while the real space $\mathbb{R}^{2n}$ contains all possible configurations of the $n$ landmark points, the physical constraints inherent to the category, as well as the constraints added by Procrustes analysis, define a submanifold of $\mathbb{R}^{2n}$ within which valid shapes of the category reside. In most practical applications, this submanifold may be described in terms of a Riemannian manifold which is known as the shape space [Dryden and Mardia, 1998]. Typically, no closed-form equations for the constraints exist and therefore the structure of the shape space is hidden, although it is known that it is low-dimensional and curved\(^1\). Since the shape space contains all the morphological information about a category of shapes, the aim is thus to construct a model of this space which defines mappings that allow the projection of an observed data vector into the low-dimensional space as well as the reconstruction of the observed data, given a point from the model space.

Although the shape space is known to be curved, the most well-known and widely used model is principal component analysis (PCA), which constructs low-dimensional linear approximations to the shape manifold and which we present in the following.

2.3.1 Principal Component Analysis

Principal component analysis (PCA) is a linear, unsupervised dimensionality reduction method which approximates the manifold of the data by a hyperplane. PCA may be derived in several different ways, although the most common involves determining the linear projection that maximises the variance of the data in the low-dimensional space [Tipping and Bishop, 1999]. In this sense, PCA reduces to finding a coordinate\(^1\)The curvature is a direct consequence of Procrustes analysis (cf. [Stegmann and Gomez, 2002]), and is potentially increased by hidden non-linear constraints.
2.3 Dimensionality Reduction

representation of the $N$ observed vectors $y_1, \ldots, y_N$ in $\mathbb{R}^D$, with respect to which the $D \times D$ sample covariance matrix

$$ S = \frac{1}{N} \sum_n (y_n - \bar{y})(y_n - \bar{y})^T $$

(2.12)

becomes diagonal and the data is decorrelated. This representation is obtained by solving the eigenvalue problem [Jolliffe, 2002]

$$ SW = W \Lambda, $$

(2.13)

where $\Lambda$ is the diagonal matrix containing the eigenvalues $\lambda_j, j = 1, \ldots, D$ and $W$ is an orthogonal matrix which contains the corresponding eigenvectors $w_j$ in its columns.\(^1\) Without loss of generality, the eigenvalues along the diagonal of $\Lambda$ may be taken to be sorted by magnitude. Dimensionality reduction is then performed by retaining the $Q < D$ eigenvectors whose corresponding eigenvalues sum to a pre-specified proportion of the sum of all eigenvalues. Using the projection matrix $W' \in \mathbb{R}^{D \times Q}$ which contains the first $Q$ eigenvectors in its columns, the observed data vectors $y_1, \ldots, y_N$ are represented by the low-dimensional vectors

$$ x_i = W'^T(y_i - \bar{y}), \quad i = 1, \ldots, N. $$

(2.15)

The original data is approximately reconstructed by inverting Eqn. (2.15), yielding

$$ \hat{y}_i = W'x_i + \bar{y}. $$

(2.16)

An equivalent derivation of PCA views the problem as that of finding the linear embedding of the data which is optimal under linear reconstruction in terms of the quadratic loss function [Jolliffe, 2002; Lawrence and Quiñonero Candela, 2006],

$$ \mathcal{L} = \sum_{n=1}^{N} \| y_n - \hat{y}_n \|. $$

(2.17)

\(^1\)Here we have assumed $N \geq D$, i.e. that there are more data vectors than data dimensions, and that the $D \times D$ covariance matrix $S$ has full rank. In the case that $N < D$, there will be zero-valued eigenvalues and it is more efficient to compute the eigenvectors of the $N \times N$ matrix

$$ \tilde{S} = \frac{1}{N} \sum_n (y_n - \bar{y})^T(y_n - \bar{y}). $$

(2.14)

In appendix B.5, we show how the eigenvectors corresponding to the non-zero eigenvalues of $S$ may be recovered from those of $\tilde{S}$. 
Depending on the curvature of the manifold, PCA may only provide a poor approximation. However, its mathematical simplicity makes PCA particularly appealing and it is used for building the linear shape and texture models employed in the original work on active appearance models [Cootes et al., 2001; Matthews and Baker, 2004] as well as for constructing “eigenfaces” in earlier work by Turk and Pentland [1991] and Belhumeur et al. [1997].

2.4 Linear Shape Model

Given a training dataset $\mathcal{T}$ and using the concepts discussed in the previous sections, applying PCA to the collection of Procrustes-aligned 2D landmark annotations yields the linear shape model

$$s = s_0 + \sum_{i=1}^{n_s} p_i s_i.$$  \hspace{1cm} (2.18)

Here $s$ is a $2n$-dimensional vector of landmark points representing a normalised shape which is synthesised by linearly combining the shape eigenmodes $s_i$ using the shape parameters $p_i$ and adding the mean shape $s_0$. Note that the shape eigenmodes $s_i$ are equivalent to the eigenvectors contained in the columns of the projection matrix $W'$ introduced in Section 2.3.1.

2.5 Linear Texture Model

Following the methods outlined in Section 2.2, a set of textures is generated by sampling from the images contained in a training dataset $\mathcal{T}$ into the texture model reference frame. Since the sampled textures consist of a set of pixels on which the one-dimensional operations of scaling and translation are defined, the textures may be interpreted as one-dimensional shapes and their analysis follows the same procedure used for 2D shapes. In particular, scaling and translation are removed by applying Procrustes analysis and aligning the texture samples to the Procrustes mean. Applying PCA (cf. Section 2.3.1) to the set of normalised texture samples, we obtain a linear texture model

$$G(r; \lambda) = G_0(r) + \sum_{j=1}^{n_t} \lambda_j G_j(r),$$  \hspace{1cm} (2.19)
where $G(r; \lambda)$ is the normalised texture at pixel $r$ obtained by linearly combining the texture eigenmodes $G_j(r)$ using the texture parameters $\lambda_j$ and adding the mean texture $G_0(r)$.

2.6 Linear Appearance Model

Rather than treating shape and texture independently, the original AAM of Cootes et al. [2001] models shape and texture jointly in a linear appearance model. Thus, the original AAM is referred to as the combined AAM by Matthews and Baker [2004]. After building independent shape and texture models, the shape and texture parameters $(p, \lambda)$ associated with each instance in the training dataset $\mathcal{T}$ are collected to form a dataset $\hat{\mathcal{T}}$, which consists of zero-mean training vectors of the type

$$b = \left( Ap \right) \lambda$$

(2.20)

where $A$ is a weighting that makes the units of $p$ and $\lambda$ commensurate. Following this, a PCA model is constructed using $\hat{\mathcal{T}}$ so that

$$b = Qc$$

(2.21)

where

$$Q = \begin{pmatrix} Q_s \\ Q_t \end{pmatrix}$$

(2.22)

is composed of two blocks which refer to the shape and texture components and where $c$ are the appearance parameters. The diagonal weighting matrix $A$ is learned from the training data by displacing the shape from the ground truth and recording the changes in magnitude of the sampled texture. Writing Eqns. (2.18) and (2.19) in terms of the appearance parameters $c$ yields the shape and texture model equations

$$s = s_0 + P_s A^{-1} Q_s c$$

(2.23)

$$G(r; c) = G_0(r) + P_t Q_t c$$

(2.24)

where $P_s$ and $P_t$ are the matrices that contain the shape and texture eigenmodes $s_i$ and $G_j(r)$ in their columns.
2.7 Fitting Generative AAMs

Active appearance models that contain both shape- and texture models are referred to as *generative* models, where the term “generative” refers to the fact that novel appearances may be synthesised from the model. Both the original active appearance model [Cootes et al., 2001] as well as the group of algorithms developed by Matthews and Baker [2004] are generative models. In a generative model, the problem of model fitting reduces to finding the pose, shape and texture model parameters with respect to which the appearance generated by the model optimally reproduces the data sampled from the image. The procedure for fitting generative AAMs is best described within the framework of an optimisation problem, which we present in the following.

2.7.1 Objective Function

Generative models allow the computation of the residual error, which is given by the difference of the sampled image data and the texture generated by the model. The problem of fitting generative models is thus reduced to an optimisation problem with the objective of minimising the residual error with respect to the model parameters. The construction of the residual image is illustrated in Figure 2.3 and the objective function is formalised in the following definition:

**Definition 1 Generative AAM Objective Function**

Given a texture model $G$ parameterised by $\lambda$, a warp field $W$ induced by a shape model $S$ with parameters $p$ and a global pose transform $N$ with parameters $q$, the problem of fitting a generative active appearance model may be posed as the minimisation of the sum of squared errors between the pixels sampled from the image $I$ using the warp $N \circ W$ and the pixels synthesised using the texture model $G$:

$$\min_{\lambda, p, q} \sum_{r \in R_t} \| E(r; \lambda, p, q) \|^2,$$

(2.25)

where $R_t$ is the texture model reference frame and

$$E(r; \lambda, p, q) = G(r; \lambda) - I(N(W(r; p); q))$$

(2.26)

is referred to as the residual image.

Early work on generative active appearance models such as [Sclaroff and Isidoro, 1998], where they were dubbed “active blobs” or [Jones and Poggio, 1998], where they were called “multidimensional morphable models”, investigated standard as well as stochastic gradient descent algorithms for jointly optimising objective functions of this type.
2.7 Fitting Generative AAMs

Figure 2.3: Computing the objective function of Eqn. (2.25) for generative AAM fitting requires backwards sampling from the image into the texture reference frame $R_t$ using the warp $N(W(r, p); q)$, generating the model texture $G(r; \lambda)$ and taking the difference to obtain the residual image $E(r; \lambda, p, q)$. The texture reference frame $R_t$ is usually defined by scaling the mean shape $s_0$ so that it covers $O(1000)$ pixels.

with respect to the parameters. At each iteration of a gradient descent algorithm, the objective function $F$ is approximated by a quadratic function and the unique minimum of this approximation is found, resulting in additive updates to the model parameters

$$p_{i+1} = p_i + \Delta p_i$$

$$\lambda_{i+1} = \lambda_i + \Delta \lambda_i$$

where $\Delta p_i = f \left( \frac{\partial F(r)}{\partial p} \right)$ and $\Delta \lambda_i = g \left( \frac{\partial F(r)}{\partial \lambda} \right)$ and $f, g$ are linear functions of the shape and texture gradients.\(^1\) Unfortunately, this procedure is computationally inefficient, as the gradients depend on the current parameter values and therefore must be recomputed at each iteration. Much research has therefore been devoted to developing efficient and practical generative AAM algorithms. Sclaroff and Isidoro [1998] investigated the use of the difference decomposition approach introduced by Gleicher [1997], while Cootes et al. [1998] developed an ad-hoc fitting algorithm based on multilinear regression before moving to an efficient gradient descent algorithm in [Cootes et al., 2001]. More recently, starting from the Lucas-Kanade image alignment algorithm [Lucas and Kanade, 1981], Matthews and Baker [2004] introduced a group of efficient fitting algorithms which

\(^1\)The pose parameters $q$ are updated compositionally and are omitted here for the sake of simplicity.
apply compositional, as opposed to additive updates to the shape model parameters. Out of the large body of work on active appearance models, the most commonly cited publications are [Cootes et al., 2001] and [Matthews and Baker, 2004]. Noting that subsequent work by Batur and Hayes [2005], Papandreou and Maragos [2008] or Amberg et al. [2009] are closely related extensions, we therefore limit the discussion on generative AAMs to the three most commonly cited algorithms, the combined AAM of [Cootes et al., 2001], the simultaneous inverse-compositional AAM of [Baker et al., 2003] and the project-out inverse-compositional AAM of [Matthews and Baker, 2004].

2.8 Combined AAM

An efficient algorithm for optimising the objective function in Eqn. (2.25) in the context of combined AAMs was introduced by Cootes et al. [2001, 1998]. In order to fit the appearance model efficiently, Cootes et al. [2001] make the simplifying assumption that since the residual image (cf. Eqn. (2.26)) is calculated in the reference frame of the texture model $G$, the linear function associated with the parameter updates may be taken to be constant for small displacements from the ground truth and may thus be pre-computed. Therefore, rather than performing the expensive update step used in gradient descent, the parameter updates at iteration $i$, $\Delta c_i$ are obtained from the residual image $e_i$ using a constant update matrix $U_c$.

$$\Delta c_i = U_c e_i. \quad (2.29)$$

While the matrix $U_c$ is obtained using linear regression in [Cootes et al., 1998], in [Cootes et al., 2001] the objective function is linearised around a small parameter displacement and the update matrix is obtained after approximating the gradients of the residual error using a finite differencing scheme and solving the resulting linear equation in the least-squares sense. Thus, the algorithm by Cootes et al. [2001] is a gradient descent algorithm in which the gradients are assumed to be constant w.r.t. the parameter values. For completeness, it is noted that the updates $\Delta q_i$ to the pose parameters $q_i$ of the pose transform $N$ are obtained equivalently using a constant update matrix $U_q$. However, letting $N_{q_i}$ denote the pose transform at iteration $i$ with parameters $q_i$, the parameters $q_{i+1}$ at the next iteration are obtained compositionally (cf. Section A.2)

$$N_{q_{i+1}} = N_{q_i} \circ N_{\Delta q_i} \quad (2.30)$$
2.9 Simultaneous Inverse Compositional AAM

In practice, the combined AAM fitting algorithm has been shown to work well as long as the shape is sufficiently close to the ground truth, where the constant approximation holds. However, Matthews and Baker [2004] use a counterexample to disprove the constant approximation hypothesis and introduce a group of principled gradient-descent fitting algorithms for appearance models that contain the shape and texture models separately. In the following we outline the concepts behind two algorithms, the simultaneous- and project-out inverse-compositional AAM, a full discussion of which is given in [Baker et al., 2003] and [Matthews and Baker, 2004], respectively. For the sake of simplicity, we omit the rigid shape transform $N$ in the following. As shown in [Matthews and Baker, 2004], this does not represent a restriction of generality, as the shape model may be generalised to include rigid transformations.

2.9 Simultaneous Inverse Compositional AAM

Inverse compositional AAM fitting optimises a slightly modified objective function. At a given step of the iterative procedure, we have the shape parameters $p$ and texture model parameters $\lambda$ and the incremental updates $\Delta p$ and $\Delta \lambda$ are found by linearising the objective

$$\min_{\Delta p, \Delta \lambda} \sum_{r \in R_t} \left[ G(W(r, \Delta p); \lambda + \Delta \lambda) - I(W(r; p)) \right]^2. \quad (2.31)$$

Here, the shape parameter updates $\Delta p$ are applied to the texture model reference frame rather than to the landmark points in the image as in Eqn. (2.25). In practice, this means that $\Delta p$ is applied in the “inverse” direction. In [Baker and Matthews, 2001], it is shown that to first order in $\Delta p$, this objective function is equivalent to Eqn. (2.25). In fact, expanding in a Taylor series and retaining first order terms, the objective becomes

$$\min_{\Delta p, \Delta \lambda} \sum_{r \in R_t} \left[ G(r; \lambda) + \frac{\partial G(r; \lambda)}{\partial \lambda} \Delta \lambda + \frac{\partial G(r; \lambda)}{\partial r} \frac{\partial W}{\partial p} \bigg|_{p=0} \Delta p - I(W(r; p)) \right]^2. \quad (2.32)$$

By combining the shape and texture parameter updates into a vector

$$\Delta c = (\Delta p, \Delta \lambda), \quad (2.33)$$

this simplifies to

$$\min_{\Delta c} \sum_{r \in R_t} \left[ E(r) - D(r) \Delta c^T \right]^2. \quad (2.34)$$
Here,

\[ E(r) = G(r; \lambda) - I(W(r; p)) \]  \hspace{1cm} (2.35)

is the error image given the current parameters \( p \) and \( \lambda \) and

\[ D(r) = \left( \frac{\partial G(r; \lambda)}{\partial r} \frac{\partial W}{\partial p}, \frac{\partial G(r; \lambda)}{\partial \lambda} \right) \Bigg|_{p=0} \]  \hspace{1cm} (2.36)

are the steepest descent images which encode the direction with respect to which local changes in the parameters \( p \) and \( \lambda \) cause the greatest reduction in fitting error. The derivative of the warp is computed at \( p = 0 \) and may therefore be precomputed. Taking the derivative of Eqn. (2.34) w.r.t. \( \Delta c \), setting to zero and solving for \( \Delta c \), we have

\[ \sum_r D(r)^T D(r) \Delta c = \sum_r D(r)^T E(r). \]  \hspace{1cm} (2.37)

Defining the Hessian matrix

\[ H = \sum_r D(r)^T D(r), \]  \hspace{1cm} (2.38)

we recover the parameter update vector

\[ \Delta c_* = H^{-1} \sum_r D(r)^T E(r). \]  \hspace{1cm} (2.39)

Following this, the texture parameters \( \lambda \) are updated additively

\[ \lambda \leftarrow \lambda + \Delta \lambda_* \]  \hspace{1cm} (2.40)

and the shape parameters \( p \) are updated \textit{inverse-compositionally} using

\[ W(r; p) \leftarrow W(r; p) \circ W(r; \Delta p_*)^{-1}, \]  \hspace{1cm} (2.41)

where to first order in \( \Delta p \) \cite{Matthews2004},

\[ W(r, \Delta p)^{-1} \equiv W(r, -\Delta p). \]  \hspace{1cm} (2.42)

Although the inverse-compositional framework allows the warp derivative \( \frac{\partial W}{\partial p} \) to be precomputed, the simultaneous optimisation of \( p \) and \( \lambda \) remains inefficient, as the steepest descent images and the Hessian must be computed at each iteration. In the following, we discuss a more efficient inverse-compositional fitting algorithm first derived in
2.10 Project-Out Inverse Compositional AAM

Starting from the objective function in Eqn. (2.25), the idea in [Matthews and Baker, 2004] is to “project-out” the effect of texture variation on the value of the objective function. This is done by performing an orthogonal decomposition of the objective function with respect to the space spanned by the texture model eigenmodes:

$$\lambda_*, p_* = \arg \min_{\lambda, p} \| G_0(r) + \sum_i \lambda_i G_i(r) - I(W(r, p)) \|$$

$$= \arg \min_{\lambda, p} \left[ \| G_0(r) - I(W(r, p)) \|_{\text{span}(G_i)} + \| G_0(r) + \sum_i \lambda_i G_i(r) - I(W(r, p)) \|_{\text{span}(G_i)} \right]$$

The first term (Eqn. (2.44)) is projected onto the orthogonal complement of the linear texture space and therefore only the constant mean texture is retained in this term. The key insight is that because the second term (Eqn. (2.45)) is projected onto the subspace collinear with the linear texture space, the texture parameters $\lambda_i$ may always be chosen such that this term becomes zero. This facilitates a sequential optimisation strategy where only the first term is optimised to determine the shape parameters $p$ using the strategy presented in Section 2.9, after which the optimal texture parameters may be determined by setting the second term to zero. The objective function in Eqn. (2.44) is implemented indirectly by projecting out the texture components of the steepest descent images for the shape parameters $p_j$:

$$\tilde{D}_j(r) = \frac{\partial G_0(r)}{\partial r} \frac{\partial W}{\partial p_j} \bigg|_{p_j=0} - \sum_i \left[ \sum_r G_i(r) \cdot \frac{\partial G_0(r)}{\partial r} \frac{\partial W}{\partial p_j} \bigg|_{p_j=0} G_i(r) \right]$$

These modified steepest descent images $\tilde{D}_j(r)$ are then used in Eqns. (2.38) and (2.39) to compute the parameter update, and since the inner product is computed with the error image, this is tantamount to projecting the residual error directly. Because the optimisation does not include the texture parameters, and the warp gradients are calculated at the position $p = 0$, the modified steepest descent images $\tilde{D}_j(r)$ and the Hessian $H$ may be precomputed so that a highly efficient algorithm results. Unfortunately, however,
it has been shown that the added efficiency associated with the project-out algorithm comes at the price of reduced generic fitting accuracy [Gross et al., 2005]. The problem is that by removing texture variation from the optimisation problem, the residual error is only measured with respect to the mean texture, regardless of the actual sampled texture. While fitting the shape model remains successful if the sampled texture only deviates slightly from the mean, this approach breaks down for larger deviations, which may be caused by colour variation or varying lighting conditions in the images.

2.11 Conclusions

In this chapter we presented basic concepts of statistical shape analysis and discussed how principal component analysis is used for building generative linear models of shape and texture which may be combined to form models of appearance. We introduced the term “active appearance models” and discussed how it refers to a combination of shape and texture models and a model-fitting algorithm which allows instances of these models to be fitted to images. We discussed the three most commonly cited AAMs, the combined AAM, the simultaneous- and the project-out inverse-compositional AAM, and showed how the different model-fitting algorithms may be described in terms of the optimisation of objective functions which compare the texture generated by the appearance models to the texture sampled from the image. We showed how gradient descent is used to optimise such objective functions and discussed how a canonical approach using exact gradients leads to accurate but inefficient algorithms such as the simultaneous inverse-compositional AAM. Finally, we presented the approximation schemes used by the combined AAM and project-out inverse-compositional AAM to make model fitting efficient and discussed how they represent a trade-off between accuracy and efficiency.
Chapter 3

Discriminative AAM

The active appearance models described in the previous chapter are composed of generative models of shape and texture which allow model fitting to be cast into the optimisation problem formalised in Definition 1, where the residual between the texture sample and the generated texture is used to drive the update of the shape model. As discussed in the previous chapter, the gradient descent algorithms employed to optimise generative models either lack efficiency or make simplifying assumptions which have the potential to reduce fitting performance. In an alternative approach termed the “discriminative” AAM, the texture model is therefore removed and fitting is done by directly using the information conveyed by the texture sample to update the shape model parameters. More specifically, a regression model is used to predict updates to the pose and shape model parameters based on a set of features obtained from the texture sample taken at the current shape model location. This is illustrated in Figure 3.1 which shows the typical workflow in a discriminative AAM. A number of methods exist for constructing and fitting such regression models, of which we describe boosting [Friedman, 2001], which was used for fitting shape models in [Zhou and Comaniciu, 2007], [Saragih and Goëcke, 2007], [Cristinacce and Cootes, 2007] and [Tresadern et al., 2010], random forests [Breiman, 2001], which were used in [Sauer et al., 2011], and Gaussian processes [Rasmussen and Williams, 2006], which, to our knowledge, have not been used previously in the context of discriminative model fitting. Furthermore, rather than applying discriminative models individually, we propose two training procedures for the construction of sequences of discriminative models, building on previous work in [Saragih and Goëcke, 2007; Tresadern et al., 2010; Zimmermann et al., 2009] and [Dollár et al., 2010]. The lack of a texture model makes discriminative AAMs more efficient and thus a common application is tracking faces in video, where problems such
as lighting variation, occlusion and the need to deal with multiple identities is common. In generative AAMs these situations are especially problematic, since they must be accounted for by the texture model in order to make fitting robust under such changes. Since the regression models used in discriminative AAMs work with the texture directly, a common approach to increasing fitting robustness is to move from image pixels to using higher-level features which are efficient to compute and which are invariant under changes such as lighting variation and changing identity. In the following, we introduce the features and regression models we have used with discriminative AAMs, before discussing the construction of sequences of discriminative AAMs and presenting results.

Figure 3.1: Shape model fitting using a discriminative AAM algorithm.

3.1 Features

Following their successful use in many applications, we employ features based on two gradient-type multiresolution approaches, *Haar features* [Viola and Jones, 2001] and *steerable pyramids* [Freeman and Adelson, 1991].

3.1.1 Haar Features

Haar features came to prominence in the face detection algorithm of Viola and Jones [2001]. The different Haar templates in Figure 3.2a indicate how Haar features encapsulate intensity gradients across the vertical, horizontal and diagonal orientations and at different scales, depending on the chosen shape of the template. Figure 3.2c shows how Haar features at any scale may be computed very efficiently using the integral image $I(N(W(r,p),q))$. 

Figure 3.2a: Haar features in a face image. The different Haar templates show how Haar features encapsulate intensity gradients across different orientations and scales.

Figure 3.2c: Efficient computation of Haar features using the integral image $I(N(W(r,p),q))$. 

Figure 3.3: Discriminative AAM fitting process. The algorithm takes an image $I$, extracts Haar features, computes feature vectors $h$, and updates the shape model $\Delta p = f(h)$. 

3.3 Regression Models
Figure 3.2: Haar features are gradient-like features and are computed by summing the pixels under the templates shown in (a). By precomputing the integral image, which is illustrated in (b), the computation of the Haar features reduces to a constant-time operation regardless of the template shape and size. An example is shown in (c).

Figure 3.3: When using Haar features for fitting discriminative AAMs, the prediction of the shape model location is based on Haar feature vectors which are determined by first computing the integral image of the sampled texture, placing the set of $n$ pre-selected Haar templates onto the texture and concatenating the calculated Haar feature values $h_i$ into a vector.
3.1 Features

described in Figure 3.2b. Furthermore, due to their similarity to multi-scale gradients, they incorporate invariance to global lighting variation, whilst the use of different Haar template shapes promotes scale invariance. These properties have resulted in Haar features being widely adopted as low-level descriptors. However, the number of possible Haar features in an image of size $N \times N$ scales as $O(N^4)$, and in order to identify salient features in the image, an effective means of feature selection is required. To this end, Haar features have been used successfully in combination with boosting classifiers as in Viola and Jones [2001], or with boosting regression within the context of discriminative face model fitting in [Zheng et al., 2006; Zhou and Comaniciu, 2007] and [Cristinacce and Cootes, 2007]. Figure 3.3 shows the computation of a feature vector from a set of Haar features selected during training of a boosting regression model.

3.1.2 Steerable Pyramid Features

Multiresolution analysis forms an effective means of capturing image features at different scales. Typically, the starting point for multiresolution analysis is the recursive application of a Gaussian low-pass filter $G$ followed by downsampling of the image, leading to the construction of a multiresolution image “pyramid”. However, in many applications, multi scale edge features are of interest, which are more suitably highlighted by computing the Laplacian of the image. In practice, this is done by subtracting adjacent levels of the Gaussian pyramid as shown in Figure 3.4a. The reason why this difference of Gaussian pyramid is equivalent to the Laplacian pyramid lies in the fact

![Figure 3.4: Image pyramids](image_url)
3.1 Features

that a Gaussian kernel, the covariance of which is linearly parameterised by the time \( t \), represents a fundamental solution to the Heat equation

\[
\frac{\partial T(r, t)}{\partial t} - \alpha \nabla^2 T(r, t) = 0.
\]  

(3.1)

Thus, by construction, the difference of Gaussian pyramid represents a finite difference approximation to the temporal derivative in Eqn. (3.1) and is thus approximately equal to the Laplacian at multiple scales. In applications such as face model fitting, however, further knowledge about the orientation of image features is useful and this is not provided by the isotropic Gaussian filters used to compute the Laplacian pyramid. A general method for performing multi-resolution analysis of images at arbitrary orientations and scales using only a set of “steerable” basis filters was introduced by Freeman and Adelson [1991] and further developed by Simoncelli and Freeman [1995].

The definition of steerability is reproduced in the following:

**Definition 2 Steerability [Freeman and Adelson, 1991]**

A function \( f : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) is steerable if it satisfies the steerability constraint

\[
f^\theta(x, y) = \sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(x, y),
\]  

(3.2)

where \( f^\theta \) is an arbitrary rotation of \( f \), \( \{f^{\theta_j}\}_{j=1}^{M} \) is a steerable basis set consisting of \( M \) rotated versions of \( f \), and \( \{k_j(\theta)\}_{j=1}^{M} \) is a set of interpolation functions.

This result carries over to sampled 2D functions, and the simplest example of a “steerable” basis is given by a set of \( n + 1 \) \( n \)th order derivatives of a radially symmetric function. For instance, given the first order derivatives of a Gaussian filter \( g(x, y) \)

\[
\nabla g(x, y) = \left( \frac{\partial g(x, y)}{\partial x}, \frac{\partial g(x, y)}{\partial y} \right),
\]  

(3.3)

the derivative at orientation \( \theta \), \( g^\theta \) may be obtained by projecting onto the unit vector \( e_\theta = (\cos \theta, \sin \theta) \) so that

\[
g^\theta(x, y) = \nabla g(x, y) \cdot e_\theta
\]

\[
= \cos \theta \frac{\partial g(x, y)}{\partial x} + \sin \theta \frac{\partial g(x, y)}{\partial y}.
\]  

(3.4)
Given the steerable basis filters, the oriented gradient of an image may then be computed simply by convolving with the filter defined in Eqn. (3.4). Figure 3.4b gives an illustration of a 2-level steerable pyramid decomposition using a set of four steerable filters $B_i$.

In the context of face model fitting we are interested in extracting features that encapsulate information about the pose and expression of faces. To this end, we employ a steerable pyramid with four orientations and compute summary statistics on the pixel values taken from overlapping regions in response images as shown in Figure 3.5. The sample statistics used are a set of linear and non-linear statistics composed of the $\text{max}$, $\text{min}$, $\text{median}$, $\text{mean}$, $\text{variance}$, $\text{skewness}$ and $\text{kurtosis}$. A similar set of features is discussed in the context of pattern classification problems in [Sauer, 2008].

3.2 Regression Models

In a discriminative AAM, a shape model is fitted using a set of regression models $f_i$ which predict updates to the shape model parameters $p_i$ directly based on features $h$ calculated from texture samples $x$:

$$dp_i = f_i(h(x)), \quad i = 1, \ldots, n_s. \tag{3.6}$$

The regression models $f_i$ are built from a separate training dataset

$$\tilde{T} = \left\{ (x_1, \delta p^{(1)}), \ldots, (x_j, \delta p^{(j)}), \ldots, (x_n, \delta p^{(n)}) \right\}, \tag{3.7}$$

39
the \( j \)th sample of which is created by randomly displacing the landmark point sets by fitting an instance of the shape model to the ground truth and perturbing the pose and shape parameters. Following this, the texture \( x_j \) is sampled from the displaced position, and the shape parameter increment vector \( \delta p^{(j)} = (dp_1^{(j)}, \ldots, dp_{ns}^{(j)}) \) required to move the landmark points back to their ground truth position is recorded.

### 3.2.1 Boosting

The boosting algorithm used in this thesis follows the gradient boosting method of Friedman [2001], which additively combines a set of \( M \) weak learners \( \varphi \) into a “strong” regression function \( f \), such that the update to the \( i \)th shape model parameter is given by

\[
dp_i = \sum_{m=1}^{M} \alpha \varphi_{im} \left( h_m(x) \right).
\]  

(3.8)

In this equation, \( h_m(x) \) represents a feature computed on the current texture sample \( x \), and \( \alpha \) is a “shrinkage” parameter, which is included to reduce overfitting [Hastie et al., 2003]. Typically, the weak learners \( \varphi_{im} \) are chosen to be simple linear or piecewise constant functions, as illustrated in Figure 3.6. The training stage of the boosting regression for the \( i \)th shape parameter is described in detail in Algorithm 2. The weak learners

![Figure 3.6: Piecewise constant function \( \varphi_{im} \) used as weak learner in a boosting regression model. The weak learner predicts an update value to the \( i \)th shape model parameter \( p_i \) given the value of the feature \( h_m \) computed from the texture sample \( x \). The black dots represent the training data \( \left\{ \left( dp_i^{(1)}, h_m(x_1) \right), \ldots, \left( dp_i^{(n)}, h_m(x_n) \right) \right\} \) from which the function \( \varphi_{im} \) is efficiently determined by computing the histogram and calculating the mean in each bin.](image-url)
3.2 Regression Models

are selected in a greedy fashion so that at the first iteration the feature which results in the best prediction of the \( n \) displacements of the \( i \)th shape parameter \( \{d_{p_i}^{(k)}\}_{k=1}^n \) is chosen. The predictions made by the weak learner are weighted by the shrinkage parameter and subtracted from the parameter displacements before the strong learner is updated and the procedure iterates. Boosting has been used previously in the context of shape model fitting, such as in [Cristinacce and Cootes, 2007; Tresadern et al., 2010; Zhou and Comaniciu, 2007], or in Sauer et al. [2011] where it was compared to random forest regression.

**Algorithm 2** Gradient boost training [Friedman, 2001] for shape model regression [Tresadern et al., 2010]

<table>
<thead>
<tr>
<th>Input: ( \tilde{T} = {(x_n, d_{p_i}^{(n)})}_{n=1}^N ), ( f_i(x) = 0, \alpha \ll 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result: ( f_i(x), {h_m}_{m=1}^M ) ( \triangleright ) “strong learner”, features</td>
</tr>
</tbody>
</table>

1: for \( m \leftarrow 1, M \) do
2: \( (h_m, \varphi_{im}) = \arg\min_{\varphi,h} \sum_{k=1}^N \left[ d_{p_i}^{(k)} - \varphi(h(x_k)) \right]^2 \)
3: for \( k \leftarrow 1, N \) do
4: \( d_{p_i}^{(k)} \leftarrow d_{p_i}^{(k)} - \alpha\varphi_{im}(h_m(x_k)) \) \( \triangleright \) update \( d_{p_i}^{(k)} \)
5: end for
6: \( f_i(x) \leftarrow f_i(x) + \alpha\varphi_{im}(h_m(x)) \)
7: end for

3.2.2 Random Forest

In recent years random forests [Breiman, 2001] have become increasingly popular and have been used successfully in many classification and regression problems. The popularity of random forests stems from the fact that they represent a simple and efficient algorithm with few free parameters and have been shown to resist overfitting in many applications. Furthermore, both the training and testing stages of random forests follow “embarrassingly parallel” procedures, a fact which may be exploited to obtain high performance on parallel computer architectures. Random forests are constructed by building a set of \( N \) binary trees (a forest) on bootstrap samples of the training dataset. The trees are constructed recursively, such that starting from the root node, a random subset of features is selected after which the training data is split by setting a thresh-
old on one of the feature variables. This creates two new leaf nodes and allows the computation of the sum of squared errors

\[ E_{sc} = \sum_{l \in L} \sum_{n \in l} (dp_{in} - m_{il})^2 = \sum_{l \in L} N_l \text{var}(l), \]  

(3.9)

where \( L \) is the total number of leaf nodes and \( m_{il} = \frac{1}{N_l} \sum_{n=1}^{N_l} dp_{in} \) and \( \text{var}(l) \) are the mean and variance of the \( N_l \) displacements of the \( i \)th shape parameter contained in leaf \( l \). Among all features available in the subset, the feature that minimises \( E_{sc} \) is chosen. Various stopping criteria for the recursion have been proposed in the literature [Breiman et al., 1984]. In our case, the trees are built until each leaf node contains a single sample and as recommended by Breiman [2001], the size of the random feature subset is chosen to be roughly equal to the square root of the total number of features. The trees are similar to the piecewise constant functions used as weak learners in Section 3.2.1, as each leaf outputs the mean of the data it contains. When presented with test data, the mean over the outputs of each tree in the forest is returned as the forest prediction. This ensemble prediction allows for the gathering of statistics on the outputs by inspecting the level of “agreement” among the individual tree regression functions.

The tree building procedure described above represents the standard procedure for scalar outputs. If, however, we wish to predict all \( n_s \) updates at once, this method must be extended to allow for vectorial outputs. As discussed in [Sauer et al., 2011], this is done either by extending the sum of squared errors to all output dimensions, i.e.

\[ E_{vec} = \sum_{i=1}^{n_s} \sum_{l \in L} \sum_{n \in l} (dp_{in} - m_{il})^2 = \sum_{i=1}^{n_s} \sum_{l \in L} N_l \text{var}_i(l), \]  

(3.10)

such that at each binary split the feature that gives the best joint prediction of the outputs is chosen, or by simply training a scalar random forest for each output dimension. Using the first method, we arrive at a regression forest capable of directly predicting vectorial outputs. This makes for a more efficient algorithm when compared to the set of scalar forests and the joint feature selection may also allow for correlations in the outputs to be learned. However, in the case of high-dimensional outputs where the individual dimensions are uncorrelated, this also represents a significant shortcoming as a single feature is required to discriminate between the uncorrelated phenomena.
3.2.3 Gaussian Processes

In the terminology of statistics, a stochastic process is defined as a set of random variables which are indexed by elements from an index set, usually time or space. Common examples of stochastic processes include Markov chains, where the index set is made up of discrete time points and Markov random fields, where the index set contains points from space. Following Rasmussen and Williams [2006], a Gaussian process may be defined as follows:

**Definition 3 Gaussian Process**

A Gaussian process is a set of random variables such that the joint distribution over any finite subset is a multivariate Gaussian distribution.

Due to the marginalisation property of multivariate Gaussians (cf. C.2), it follows directly from the definition that a Gaussian process is completely defined by a mean function \( m(x) \) and a covariance function \( k(x, x') \) which is also referred to as a *kernel*:

\[
 f(x) \propto \mathcal{GP}(m(x), k(x, x')) \tag{3.11}
\]

where

\[
 m(x) = \mathbb{E}[f(x)] \tag{3.12}
\]
\[
 k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{3.13}
\]

In practice, the mean function is often taken to be zero, as this simplifies the mathematics and does not represent a restriction of generality.

**Function Priors** In the context of Bayesian statistics, a Gaussian process is often interpreted as a prior distribution over functions [Rasmussen and Williams, 2006]. Although functions are infinite dimensional objects, the marginalisation property of multivariate Gaussians allows us to model a function by considering the joint distribution over a finite set of function values. The type of kernel determines the space of functions modelled by the prior. One of the most commonly used kernels is the radial basis function (RBF)

\[
 k(x_i, x_j) = \sigma_f \exp \left( -\frac{1}{2l} \sum_{m=1}^{D} (x_{im} - x_{jm})^2 \right) \tag{3.14}
\]

which models the space of smooth functions. The two hyperparameters \( \theta_f = (\sigma_f, l) \) correspond to the variance and the characteristic length scale of the functions modelled.
by the prior. The length scale determines the amount of correlation between the inputs \((x_1, \ldots, x_N)\) and may be thought of intuitively as the degree of flexibility of the functions. Functions may be sampled from the prior by using the following recipe:

1. specify a set of \(N\) input locations \(x \in \mathbb{R}^D\)
2. evaluate the elements of the covariance matrix
   \[ K_{ij} = k(x_i, x_j), \quad i, j = 1, \ldots, N \]  
   (3.15)
3. sample function values from the Gaussian with covariance \(K\):
   \[ f \propto \mathcal{N}(0, K) \]  
   (3.16)

Figure 3.7a shows six independent samples drawn from a Gaussian process prior using an RBF kernel. It is clear from the sampling recipe that the variance \(\sigma_f\) of the kernel is a gain parameter that influences the magnitude of the sampled function values. If the function is referred to as a signal, the variance may thus be thought of as a signal strength.

Figure 3.7: (a) shows six samples drawn from a GP prior using an RBF kernel with \(\sigma_f = 2, l = 2\). (b) contains the mean (red line) and variance (shaded region) of the posterior process obtained after conditioning the joint prior on the training data points, marginalising and optimising the kernel hyperparameters. A similar illustration is given in [Rasmussen and Williams, 2006, chap. 2].
Regression Models

Regression

In a regression problem, the task is to uncover the function \( f(x) \) given a set of noisy observations \( y = (y_1, \ldots, y_N) \) at inputs \( X = (x_1, \ldots, x_N) \). In other words, we are trying to learn the model

\[
y = f(x) + \epsilon,
\]

where usually \( \epsilon \propto N(0, \sigma^2) \). Once fitted, the regression model allows one to predict outputs \( f_\star \) given test inputs \( x_\star \). However, given only the training data \( (y, X) \), and without making any prior assumptions about the form of \( f \), both the variables \( f \), which correspond to the values of the regression function at the inputs \( X \), as well as the predictions \( f_\star \) at \( X_\star \) are unknown. In a Bayesian treatment of the regression problem this uncertainty must be captured in the form of a prior distribution \( p(f, f_\star) \) over the latent variables \( f \) and \( f_\star \). Combined with the data likelihood \( p(y|f) \), this allows the predictive posterior distribution \( p(f_\star|y) \) to be inferred using Bayes’ theorem. In the case of Gaussian process regression, the prior distribution is a Gaussian process, and the observed data \( y \) is modelled by the likelihood

\[
p(y|f) \propto N(f, \sigma^2 I).
\]

Invoking Bayes’ rule, the joint posterior becomes

\[
p(f, f_\star|y) = \frac{p(f, f_\star)p(y|f)}{p(y)},
\]

where \( p(y) \) is the marginal likelihood which is obtained by integrating out the latent variables \( f \):

\[
p(y) = \int p(y, f)d\mathbf{f} = N(y|0, K_{NN} + \sigma^2 I).
\]

Here we have used the notation \( K_{NN} = K(X, X) \) to denote the kernel matrix evaluated at all points \( \mathbf{x}_n \in X \). As is often the case in Gaussian computations, this integral is most easily solved by inspection starting from the joint distribution and using the identities given in appendix C. Noting that \( p(y, f) \) is a Gaussian process, the result immediately follows by applying the marginalisation property. The joint posterior of Eqn. (3.19) still contains the latent variables \( f \) associated with the training input data \( X \). Following the sum rule of probability theory, the predictive posterior distribution \( p(f_\star|y) \) is obtained
by marginalising the set of variables $f$ in the joint posterior, the result of which is also Gaussian:

$$p(f_*|y) \propto N(m_*, C_*),$$  \hspace{1cm} (3.21)

where

$$m_* = K_{sN} \left( K_{NN} + \sigma_e^2 I \right)^{-1} y$$  \hspace{1cm} (3.22)

$$C_* = K_{ss} - K_{sN} \left( K_{NN} + \sigma_e^2 I \right)^{-1} K_{Ns}$$  \hspace{1cm} (3.23)

are the predictive mean $m_*$ and covariance $C_*$. A full derivation of Eqns. 3.20 and 3.22 is given in appendix C.6. These equations depend on the training and testing inputs $X$ and $X_*$ via the kernels, where we have used the shorthand notation $K_{Ns} = K(X, X_*)$, as well as on the kernel hyperparameters $\theta_f$ and the noise hyperparameter $\sigma_e$. The hyperparameters are not treated in a Bayesian manner, since placing prior distributions over the hyperparameters, computing the hyperposteriors and marginalising is often not analytically tractable or leads to a drastic increase in complexity. Therefore, the values of the hyperparameters must be determined using a procedure known as “empirical Bayes” or “type-II maximum likelihood”, where the marginal likelihood of Eqn. (3.20) is optimised with respect to the hyperparameters. A discussion and motivation of the empirical Bayes method for model selection is given by Rasmussen and Williams [2006, chap. 5]. Figure 3.7b shows the predictive mean function of Eqn. (3.22) together with two standard deviations of variance (the shaded area) for 10 noisy observations of a sinusoid. The marginal likelihood was optimised with respect to the hyperparameters using the conjugate gradient optimiser of Møller [1993]. As expected, the variance of the posterior process is small around the observed data points, but quickly increases to the prior variance in regions where no data is available. The noise parameter was fixed to zero for this illustration and therefore the regression function perfectly interpolates the data points. In the case of non-zero noise variance, part of the signal is explained by noise and consequently the regression function no longer exactly fits the data. The ratio of the variance of the regression function and the variance of the noise is referred to as the signal-to-noise ratio (SNR) and provides a measure for the goodness-of-fit of the regression function.

### 3.2.4 Sparse Gaussian Process Regression

In non-parametric kernel-based models such as Gaussian processes, the number of parameters increases linearly with the size of the training data. In the case of Gaussian
3.2 Regression Models

process regression, the presence of the inverse of the kernel matrix $K_{NN}$ in the predictive mean Equation 3.22 results in a computational complexity of $O(N^3)$, where $N$ is the number of training data. Unfortunately, this complexity renders Gaussian process regression unusable for datasets containing more than a few thousand samples. Many methods have been proposed to reduce the computational burden and a unifying review is given by Quiñonero Candela and Rasmussen [2005]. The general idea is based on the observation that often the training data $(y, X)$ contains redundant information about the function $f(x)$. Therefore, a carefully chosen subset of the training data should suffice to correctly infer $f(x)$. To this end, an additional set of $M \ll N$ inducing variables $u = [u_1, \ldots, u_M]$ is introduced. These are drawn from the same Gaussian process prior as $f$ (and $f^*$) but the covariance function is restricted to a set of $M$ inducing input locations $X_u$, which, ideally, are chosen so that the prior over $u$ captures all the information about $f$. The crucial step is the assumption that the inducing variables $u$ represent a sufficient statistic for $f$, i.e. that $f$ and $f^*$ are conditionally independent given $u$. With this assumption, the joint prior $p(f, f^*)$ may be approximated as\footnote{Note that the exact expression would be}

$$q(f, f^*) = \int q(f^*|u)q(f|u)p(u|X_u)du.$$ (3.25)

The choice of the term “inducing variable” becomes clearer from this equation, where both $f$ and $f^*$ are conditioned on (the same) $u$. Therefore, $f$ and $f^*$ can only communicate via $u$ and thus $u$ induces the dependencies between training and test data. The various sparsification algorithms discussed in Quiñonero Candela and Rasmussen [2005] make further assumptions about the inducing conditionals $q(f|u)$ and $q(f^*|u)$ and typically result in a much more manageable complexity of $O(NM^2)$. However, finding a set of inducing variables that are sufficient statistics for $f$ is hard, and in sparse approximations such as by Snelson and Ghahramani [2006], the inducing variables are selected by optimising approximations to the exact GP prior with respect to the inducing inputs $X_u$. As noted by Titsias [2009], this may lead to unexpected behaviour of the resulting model, since no bounds are provided to guarantee closeness of the result to the theoretical ground truth given by the full GP model. In our work, we therefore use the variational sparse GP approach derived by Titsias [2009], which uses a variational

$$p(f, f^*) = \int p(f^*, f|u)p(u|X_u)du$$ (3.24)

and that the inducing inputs $X_u$ are variational hyperparameters rather than random variables, i.e. Eqn. (3.24) holds regardless of their values. However, their values do influence the likelihood of the data given the model.
lower bound $F_V$ (cf. [Bishop, 2006, chap. 10]) to the marginal likelihood for selecting the inducing inputs $X_u$, the kernel hyperparameters $\theta_f$ and the inverse noise variance $\beta = \frac{1}{\sigma_\epsilon^2}$.

$$F_V(X_u, \theta_f, \beta) = \log \mathcal{N}(y|0, \tilde{K}_{NN} + \sigma_\epsilon^2 I) - \frac{1}{2\sigma_\epsilon^2} \text{Tr}(K_{NN} - \tilde{K}_{NN}). \tag{3.26}$$

The first term, which is seen to be formally equivalent to Eqn. 3.20, contains the Nyström approximation (cf. [Rasmussen and Williams, 2006, chap. 8])

$$\tilde{K}_{NN} = K_{NM} K^{-1}_{MM} K_{MN} \tag{3.27}$$

of the full covariance matrix $K_{NN}$. The trace term becomes zero if $M$ inducing inputs $X_u$ are found such that the inducing variables represent a sufficient statistic for $f$ and the Nyström approximation becomes exact. Therefore, the trace term ensures that maximising $F_V$ leads to a good sparse approximation of the full Gaussian process. The details of the derivation are given in [Titsias, 2009], but for completeness the predictive mean and covariance become

$$m_{vs}^* = K_{*M} K^{-1}_{MM} \mu \tag{3.28}$$

$$C_{vs}^* = K_{**} - K_{*M} K^{-1}_{MM} K_{M*} + K_{*M} K^{-1}_{MM} A K^{-1}_{MM} K_{M*}, \tag{3.29}$$

where

$$\mu = \frac{1}{\sigma_\epsilon^2} K_{MM} \left( K_{MM} + \frac{1}{\sigma_\epsilon^2} K_{MN} K_{NM} \right)^{-1} K_{MN} y \tag{3.30}$$

$$A = K_{MM} \left( K_{MM} + \frac{1}{\sigma_\epsilon^2} K_{MN} K_{NM} \right)^{-1} K_{MM}. \tag{3.31}$$
3.3 Sequential Discriminative Models

As discussed in the previous section, discriminative AAMs update the shape model parameters directly by applying the learned regression model to features extracted from the image at the current shape model location. This is more efficient than the generative fitting procedure outlined in Section 2.7, in which the shape model parameter updates are obtained by iteratively minimising an error measure such as Eqn. (2.25) which relates the generated texture to the texture sampled from the image. However, an important advantage of the generative fitting algorithm is that a well-defined objective function is optimised. At each iteration of the optimisation procedure, the update of the texture model parameters leads to a change in the value of the objective function and thus information is fed back into the next iteration of the search. Such reinforcement is lacking in discriminative AAM fitting where a simple, “one-shot” update to the shape model parameters is made, based on the features computed from the current texture sample. This drawback becomes especially apparent when applying a single discriminative model iteratively, such that the same regression model is used at each iteration regardless of whether the location of the model has changed. It is obvious that the accuracy of the regression procedure is limited by the training data used to train the regression models. If mainly large displacements from the ground truth are used to train the model, it can be expected to perform well when tested on examples showing similar displacements. However, when applied in an iterative fashion, the model is expected to move closer to the ground truth, and thus further from the type of examples provided in the training data, making the ensuing predictions approximate at best. One way of addressing this issue, termed the “sequential” AAM in the following, builds on similar ideas published in [Dollár et al., 2010], which were developed in [Tresadern et al., 2010] and [Sauer et al., 2011]. In order to mimic the reinforcement provided by the texture model in the generative fitting algorithm, we investigate sequential discriminative AAMs in which sequences of discriminative AAMs are constructed by adapting the training data as well as the model complexity according to the position of the model in the sequence. The exact training procedure is chosen using suitable heuristics but, typically, the earlier stages of the sequence only model pose variation whilst increasing amounts of shape variation are introduced in the later stages. Furthermore, the magnitude of the displacements used for training the shape models is gradually reduced as the sequence progresses. In the following, we present two training algorithms for sequential AAMs, which were first published in [Sauer et al., 2011].
3.3 Sequential Discriminative Models

Figure 3.8: Training procedures for the sequential AAM models [Sauer et al., 2011]. (a) At step $n$ of the coupled training procedure, the partial sequence $\text{SEQ}(n-1)$ is tested on a dataset $T(n)$ which contains random displacements of the same range as $T(1)$. The predicted landmark points are then used to train a discriminative AAM consisting of a regression model (REG) and a shape model (SM). The model is then appended to the sequence and the procedure iterates. (b) The independent sequential AAM consists of a set of discriminative AAM models which are trained on independent datasets $T(n)$ whose displacement range is reduced from step to step.

3.3.1 Coupled Sequential AAM

In the coupled sequential AAM, the first model in the sequence is trained using a dataset generated by perturbing the landmark points from the ground truth locations as described in Section 3.2. The model is then used to predict the ground truth locations in a separate test dataset and the resulting point locations are recorded to form the training dataset for the next model in the sequence. This procedure is illustrated in Figure 3.8a, and is repeated until the end of the sequence is reached. Since the magnitude of the displacements in the training data decreases from stage to stage, and the complexity of the shape models increases, each stage may be seen to represent a discriminative AAM specialised to a certain range of displacements from the ground truth. As was shown in [Sauer et al., 2011], a downside to this procedure is the use of the outputs of the previous stages as training data for the following stage, which may cause overfitting to occur.

3.3.2 Independent Sequential AAM

In order to assess the degree of overfitting caused by the training procedure for the coupled sequential AAM, we propose an alternative training procedure we refer to as the independent sequential AAM, in which the models in the sequence are trained independently using training data whose range of displacements is reduced by an empirically chosen factor from stage to stage. This procedure is illustrated in Figure 3.8b.
3.4 Experiments

In order to compare the discriminative AAM algorithms presented in the previous sections to existing generative algorithms, as well as to measure the performance of the sequential discriminative models, we performed a set of experiments using landmark annotated images from three different face image databases. The CogSys\(^1\) database (cf. Figure 3.9) contains video sequences of individuals performing cognitive tasks; in our experiments we used images taken from sequences of four individuals reciting the alphabet from memory. The XM2VTS database ([XM2], cf. Figure 3.10a) contains static images of many individuals with neutral facial expressions, and the BioID database ([Bio], cf. Figure 3.10b) is made up of web cam images taken in an office environment. For each image database, we created separate training and testing datasets containing 50 images each. Since the landmark annotations provided with the datasets do not coincide, we chose a common subset containing 17 points shown in red in Figures 3.9, 3.10a and 3.10b. This subset is commonly used for face analysis and corresponds to the annotation scheme of the BioID database.

![Figure 3.9: Four images from the CogSys database including the 68-point landmark annotations. The 17-point subset used for our experiments is shown in red.](image)

3.4.1 Error Measure

The performance of the algorithms is measured in terms of the fitting accuracy which is quantified by computing the mean point-to-point error between the ground truth points and the shape model points after fitting. More specifically, if

\[
s_m = (x_1, y_1, \ldots, x_n, y_n) \quad (3.32)
\]

\[
s_{gt} = (\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_n, \tilde{y}_n) \quad (3.33)
\]

\(^1\)This dataset was the result of a joint effort between the University of Manchester and Manchester Metropolitan University. At the time of writing, this dataset is not publicly available.
are the landmark point vectors associated with the model and the ground truth, the error

\[ E = \frac{100}{n \delta_{gt}} \sqrt{\sum_{i=1}^{2n} (s_{mi} - s_{gti})^2} \]  

(3.34)

is computed as a percentage of the inter-ocular distance (IOD) \( \delta_{gt} \) in the ground truth shape.

### 3.4.2 Train/Test Dataset Generation

The training data for the discriminative AAMs were generated by rigidly aligning the shape model reference points with the ground truth points before randomly perturbing the pose and shape model parameters within predefined bounds \([dp_{i}^{\min}, dp_{i}^{\max}]\). For each dataset, a total of 4000 instances were generated and, in practice, the random displacements \( dp_{i} \) were computed by sampling from a truncated standard normal distribution before multiplying by the maximum displacement range. The same procedure was followed for generating the test data for both the discriminative and generative AAMs. Two test datasets were generated for each image dataset, one initialised using small perturbations of up to 6% IOD and the other using large perturbations of up to 15% IOD.
3.4 Experiments

3.4.3 Generative AAMs

A common method of increasing the robustness of generative AAM models is to use a multiresolution approach in which the search is started at the lowest resolution before moving to the higher resolution levels. Thus in our experiments, the combined AAM, simultaneous inverse-compositional AAM (SIC) and project-out inverse-compositional AAM (POIC) algorithms were trained as 2-level multiresolution models, using texture samples containing 10000 pixels at the highest resolution. This resolution is of the same order of magnitude as the models used in Baker and Matthews [2004]. During testing, a maximum of 20 iterations per model were used which is usually sufficient to achieve convergence. A further increase in robustness in generative AAMs may be obtained by generalising the texture model to work with higher-level image features such as gradient responses. However, in such models the reconstruction of the image data is no longer possible and we therefore chose to use the standard pixel-based approach in these experiments. Results obtained from testing these models on the CogSys dataset are shown at the bottom of tables 3.2 and 3.3.

<table>
<thead>
<tr>
<th>Models</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_{pix}</td>
<td>1000</td>
<td>1000</td>
<td>3000</td>
<td>4000</td>
<td>5000</td>
</tr>
<tr>
<td>dq(%)</td>
<td>15</td>
<td>10</td>
<td>7</td>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td>n_{shape}</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>n_{lim}</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>dp(\sigma)</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dp_{lim}(\sigma)</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>0.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3.1: Training procedure for independent AAM sequences containing 5 models with increasing pixel numbers n_{pix}. The training data for each discriminative AAM model is generated by randomly perturbing the pose parameters q by a maximum of dq percent and the shape model parameters p by a maximum of dp standard deviations. The first two models only model pose variation, whereas the following models include shape models containing 4, 8 and 12 shape parameters. Assuming convergence, and to increase robustness, from the 4th model onwards, the perturbation of the first n_{lim} shape model parameters is constrained to bounds specified in dp_{lim}.

3.4.4 Discriminative AAMs

In order to investigate the contributions to fitting accuracy made by the different features and regression models as well as the sequence training algorithms, we performed a series of model-fitting experiments using sequences of 5 discriminative AAMs using the
boosting, random forest and variational GP regression algorithms in combination with both Haar features and steerable pyramid features. Furthermore, a separate experiment was carried out to assess the performance of the variational GP compared to the full GP algorithm. Experimental details were as follows:

**Sequences** The independent AAM sequences (cf. 3.3.2) were trained according to the heuristic detailed in table 3.1. The first two models in the sequence accounted for rigid pose variation and non-rigid shape variation was added in models 3, 4 and 5. The number of parameters of the shape models increased throughout the sequence and because convergence in the shape parameters was expected to occur, the later stages applied bounds to the shape model parameters in order to encourage robustness. The same model settings were used for the coupled AAM sequences (cf. 3.3.1) and the training data for the first model in the sequence was generated as specified in table 3.1. In order to measure the effect of increasing pixel resolution, the sequence experiments were repeated but the number of pixels used for the texture samples was set to 1000 for all the models in the sequences.

**Features** The Haar features and steerable pyramid features were generated as described in Section 3.1.1 and 3.1.2. We used the Haar templates shown in Figure 3.2a and the steerable pyramid features were generated as shown in Figure 3.5.

**Boosting** For the boosting regression models (cf. 3.2.1), we used 200 piecewise-constant functions as weak learners and the shrinkage factor was set to 0.05.

**Random Forests** Following results published in [Sauer et al., 2011], we used scalar random forests containing 100 trees for each output dimension rather than the vectorial forests discussed in Section 3.2.2. Each tree was built to completion during training.

**Gaussian Processes** Since the GP regression procedures discussed in Section 3.2.3 do not provide a means for feature selection, we used the features selected by the boosting algorithm to train the GP models. The variational GP models (cf. 3.2.4) were trained using an RBF kernel and 100 inducing inputs. The noise and kernel variance were set to 0.1 and 1, respectively, leading to an initial signal-to-noise ratio of 10. The number of iterations of the non-linear optimiser was limited to 1000.

The results of the experiments on the CogSys dataset are reported in tables 3.2 and 3.3. Although these experiments were carried out on independent training and test datasets, the images were nevertheless sampled from video sequences of the same individuals with
a constant background. The experiments therefore allowed only limited conclusions regarding overfitting and the generalisation capability of the algorithms used. We therefore added a set of cross-dataset experiments using the XM2VTS and BioID datasets, the results of which are shown in table 3.4.

3.4.5 Implementation

The software for all the experiments contained in this chapter was implemented in C++. Parts of Tim Cootes’s non-public software were used to implement the shape and appearance models, as well as for the boosting and random forest regression algorithms. The Gaussian process algorithms were implemented from scratch in C++ and tested against a non-public Matlab implementation by Michalis Titsias. All the experiments could have been run on a standard desktop machine, but in order to speed up the collection of data, the CSF (Computational Shared Facility) servers at the University of Manchester were used.
<table>
<thead>
<tr>
<th>Error Quantile</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Displacement</td>
<td>2.11</td>
<td>2.99</td>
<td>3.92</td>
<td>6.19</td>
</tr>
</tbody>
</table>

### Discriminative AAM

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Increasing Resolution</th>
<th>Equal Resolution</th>
<th>Coupled Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Haar</td>
<td>1.39(1)</td>
<td>1.62(1)</td>
<td>1.93(2)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.84(1)</td>
<td>0.99(1)</td>
<td>1.16(1)</td>
</tr>
<tr>
<td>Boost</td>
<td>Haar</td>
<td>0.68(1)</td>
<td>0.83(0)</td>
<td>0.97(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.80(1)</td>
<td>1.02(1)</td>
<td>1.27(1)</td>
</tr>
<tr>
<td>VarGP</td>
<td>Haar</td>
<td>0.19(0)</td>
<td>0.24(1)</td>
<td>0.31(0)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.33(0)</td>
<td>0.41(0)</td>
<td>0.53(1)</td>
</tr>
<tr>
<td>FullGP</td>
<td>Haar</td>
<td>0.13(0)</td>
<td>0.18(0)</td>
<td>0.25(0)</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.71(1)</td>
<td>0.85(1)</td>
<td>1.02(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Increasing Resolution</th>
<th>Equal Resolution</th>
<th>Coupled Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Haar</td>
<td>0.70(1)</td>
<td>0.87(1)</td>
<td>1.15(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.70(0)</td>
<td>0.81(0)</td>
<td>0.95(1)</td>
</tr>
<tr>
<td>Boost</td>
<td>Haar</td>
<td>0.57(1)</td>
<td>0.71(1)</td>
<td>1.15(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.61(1)</td>
<td>0.74(1)</td>
<td>0.92(1)</td>
</tr>
<tr>
<td>VarGP</td>
<td>Haar</td>
<td>0.18(0)</td>
<td>0.24(0)</td>
<td>0.31(0)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.29(0)</td>
<td>0.33(0)</td>
<td>0.41(0)</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.50(1)</td>
<td>0.62(1)</td>
<td>0.82(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Increasing Resolution</th>
<th>Equal Resolution</th>
<th>Coupled Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Haar</td>
<td>0.75(1)</td>
<td>0.93(1)</td>
<td>1.27(2)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.89(1)</td>
<td>1.11(1)</td>
<td>1.46(2)</td>
</tr>
<tr>
<td>Boost</td>
<td>Haar</td>
<td>0.69(1)</td>
<td>0.87(1)</td>
<td>1.05(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.78(1)</td>
<td>0.97(1)</td>
<td>1.15(1)</td>
</tr>
<tr>
<td>VarGP</td>
<td>Haar</td>
<td>0.80(0)</td>
<td>0.94(0)</td>
<td>1.11(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.68(1)</td>
<td>0.87(1)</td>
<td>1.20(2)</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.77(1)</td>
<td>0.95(1)</td>
<td>1.21(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Increasing Resolution</th>
<th>Equal Resolution</th>
<th>Coupled Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Haar</td>
<td>0.74(1)</td>
<td>0.92(1)</td>
<td>1.27(2)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.82(1)</td>
<td>1.00(1)</td>
<td>1.29(2)</td>
</tr>
<tr>
<td>Boost</td>
<td>Haar</td>
<td>0.68(1)</td>
<td>0.89(1)</td>
<td>1.06(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.78(1)</td>
<td>0.97(1)</td>
<td>1.15(1)</td>
</tr>
<tr>
<td>VarGP</td>
<td>Haar</td>
<td>0.84(0)</td>
<td>0.97(0)</td>
<td>1.15(1)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.75(1)</td>
<td>0.95(1)</td>
<td>1.13(1)</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.77(1)</td>
<td>0.95(1)</td>
<td>1.18(1)</td>
</tr>
</tbody>
</table>

### Generative AAM

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Increasing Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combined AAM</td>
<td>0.09(0)</td>
<td>0.13(1)</td>
</tr>
<tr>
<td>POIC AAM</td>
<td>0.19(2)</td>
<td>1.71(14)</td>
</tr>
<tr>
<td>SIC AAM</td>
<td>0.32(3)</td>
<td>1.64(8)</td>
</tr>
<tr>
<td>Mean</td>
<td>0.29(2)</td>
<td>1.16(8)</td>
</tr>
</tbody>
</table>

Table 3.2: Model-fitting experiments using the small displacement CogSys test dataset. The numbers refer to the quantiles of the error distribution after fitting with the specified model/feature combinations. The error quantiles before fitting are listed under “Initial Displacement”. The 95% confidence interval of the Maritz-Jarrett standard error [Maritz and Jarrett, 1978] is included in parentheses and is given as a multiple of the last digit location, e.g. $2.45(6) = 2.45 \pm 0.06$. 
<table>
<thead>
<tr>
<th></th>
<th>Error Quantile</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial Displacement</strong></td>
<td></td>
<td>4.62</td>
<td>6.54</td>
<td>8.66</td>
<td>11.58</td>
</tr>
<tr>
<td><strong>Discriminative AAM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>1.68(2)</td>
<td>2.66(16)</td>
<td>8.39(12)</td>
<td>13.11(25)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.95(1)</td>
<td>1.38(4)</td>
<td>4.64(18)</td>
<td>9.76(32)</td>
</tr>
<tr>
<td></td>
<td>Boost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.80(1)</td>
<td>1.18(5)</td>
<td>7.52(31)</td>
<td>12.64(24)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.95(1)</td>
<td>1.32(3)</td>
<td>3.13(15)</td>
<td>8.68(29)</td>
</tr>
<tr>
<td></td>
<td>VarGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.21(0)</td>
<td>0.30(1)</td>
<td>5.59(63)</td>
<td>12.07(23)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.36(0)</td>
<td>0.50(1)</td>
<td>1.26(16)</td>
<td>9.42(57)</td>
</tr>
<tr>
<td></td>
<td>FullGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.15(0)</td>
<td>0.22(1)</td>
<td>3.49(98)</td>
<td>12.19(34)</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>0.83(1)</td>
<td>1.22(5)</td>
<td>5.08(26)</td>
<td>10.95(32)</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.87(2)</td>
<td>1.64(8)</td>
<td>8.81(12)</td>
<td>12.74(10)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.79(1)</td>
<td>1.26(5)</td>
<td>4.36(14)</td>
<td>9.61(31)</td>
</tr>
<tr>
<td></td>
<td>Boost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.68(1)</td>
<td>1.11(4)</td>
<td>7.33(29)</td>
<td>12.38(26)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.72(1)</td>
<td>1.03(2)</td>
<td>3.39(22)</td>
<td>10.40(31)</td>
</tr>
<tr>
<td></td>
<td>VarGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.21(0)</td>
<td>0.30(1)</td>
<td>4.96(67)</td>
<td>12.32(21)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.31(0)</td>
<td>0.42(1)</td>
<td>5.13(40)</td>
<td>12.63(38)</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>0.60(1)</td>
<td>0.96(4)</td>
<td>5.67(31)</td>
<td>11.68(26)</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>1.36(4)</td>
<td>3.83(32)</td>
<td>8.02(14)</td>
<td>11.65(21)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>1.77(5)</td>
<td>3.62(11)</td>
<td>6.87(14)</td>
<td>11.70(22)</td>
</tr>
<tr>
<td></td>
<td>Boost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>1.05(3)</td>
<td>3.16(17)</td>
<td>7.57(17)</td>
<td>11.82(30)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>1.34(3)</td>
<td>3.10(11)</td>
<td>5.99(13)</td>
<td>9.59(21)</td>
</tr>
<tr>
<td></td>
<td>VarGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.95(1)</td>
<td>1.73(17)</td>
<td>6.95(20)</td>
<td>11.09(26)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.86(1)</td>
<td>1.40(4)</td>
<td>5.32(24)</td>
<td>10.95(40)</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>1.22(3)</td>
<td>2.81(15)</td>
<td>6.78(17)</td>
<td>11.13(27)</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>1.34(4)</td>
<td>3.78(15)</td>
<td>7.87(13)</td>
<td>11.61(26)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>1.53(5)</td>
<td>3.45(11)</td>
<td>6.24(12)</td>
<td>10.41(25)</td>
</tr>
<tr>
<td></td>
<td>Boost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>1.06(3)</td>
<td>3.21(16)</td>
<td>7.57(16)</td>
<td>11.80(30)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>1.06(1)</td>
<td>2.37(16)</td>
<td>5.67(14)</td>
<td>10.21(26)</td>
</tr>
<tr>
<td></td>
<td>VarGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haar</td>
<td>0.99(1)</td>
<td>1.72(18)</td>
<td>6.93(20)</td>
<td>11.11(26)</td>
</tr>
<tr>
<td></td>
<td>StPy</td>
<td>0.97(1)</td>
<td>1.69(14)</td>
<td>6.50(14)</td>
<td>10.21(26)</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>1.16(3)</td>
<td>2.70(15)</td>
<td>6.80(15)</td>
<td>10.89(27)</td>
</tr>
<tr>
<td><strong>Combined AAM</strong></td>
<td></td>
<td>0.21(17)</td>
<td>8.28(13)</td>
<td>13.39(14)</td>
<td>17.62(16)</td>
</tr>
<tr>
<td><strong>POIC AAM</strong></td>
<td></td>
<td>5.07(27)</td>
<td>8.89(11)</td>
<td>11.41(11)</td>
<td>14.34(25)</td>
</tr>
<tr>
<td><strong>SIC AAM</strong></td>
<td></td>
<td>4.21(10)</td>
<td>6.79(13)</td>
<td>9.19(12)</td>
<td>12.42(11)</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td></td>
<td>3.16(18)</td>
<td>7.99(12)</td>
<td>11.33(12)</td>
<td>14.79(17)</td>
</tr>
</tbody>
</table>

Table 3.3: Model-fitting experiments using the large displacement CogSys test dataset. The numbers refer to the quantiles of the error distribution after fitting with the specified model/feature combinations. The error quantiles before fitting are listed under “Initial Displacement”. The 95% confidence interval of the Maritz-Jarrett standard error [Maritz and Jarrett, 1978] is included in parentheses and is given as a multiple of the last digit location, e.g. 0.86(1) ≡ 0.86 ± 0.01.
Table 3.4: Cross-dataset fitting experiments using the large displacement BioID and XM2VTS datasets. All experiments use a 5-stage independent discriminative AAM sequence, built using the procedure specified in table 3.1 except that 1000 pixels are used throughout the sequences. The error quantiles before fitting are listed under “Initial Displacement”. The 95% confidence interval of the Maritz-Jarrett standard error [Maritz and Jarrett, 1978] is included in parentheses and is given as a multiple of the last digit location, e.g. 7.11(105) ≡ 7.11 ± 1.05.
3.4 Experiments

<table>
<thead>
<tr>
<th>Timings</th>
<th>Combined</th>
<th>SIC</th>
<th>POIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (min)</td>
<td>5.9</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Test (ms)</td>
<td>225.2</td>
<td>1.1E4</td>
<td>157.9</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Timings</th>
<th>RF Haar</th>
<th>RF StPy</th>
<th>Boost Haar</th>
<th>Boost StPy</th>
<th>VarGP Haar</th>
<th>VarGP StPy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (h)</td>
<td>392.5</td>
<td>0.6</td>
<td>60.1</td>
<td>1.5</td>
<td>120.6</td>
<td>8.8</td>
</tr>
<tr>
<td>Test (ms)</td>
<td>18.9</td>
<td>141.4</td>
<td>9.2</td>
<td>139.5</td>
<td>381.6</td>
<td>509.1</td>
</tr>
</tbody>
</table>

(b)

Table 3.5: (a) Timings for training and testing 2-level generative AAM models, where 20 iterations per level were used for testing. (b) Timings for 5-stage independent discriminative AAM sequences using Haar features. The test timings represent the mean over all instances in the test dataset.

<table>
<thead>
<tr>
<th>Models</th>
<th>RF Haar</th>
<th>RF StPy</th>
<th>Boost Haar</th>
<th>Boost StPy</th>
<th>VarGP Haar</th>
<th>VarGP StPy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94.82</td>
<td>95.33</td>
<td>96.05</td>
<td>95.18</td>
<td>97.23</td>
<td>97.03</td>
</tr>
<tr>
<td>2</td>
<td>96.92</td>
<td>97.90</td>
<td>98.00</td>
<td>98.15</td>
<td>98.56</td>
<td>98.87</td>
</tr>
<tr>
<td>3</td>
<td>96.56</td>
<td>98.05</td>
<td>98.41</td>
<td>98.67</td>
<td>98.67</td>
<td>99.13</td>
</tr>
<tr>
<td>4</td>
<td>96.51</td>
<td>98.05</td>
<td>98.36</td>
<td>98.67</td>
<td>98.77</td>
<td>99.23</td>
</tr>
<tr>
<td>5</td>
<td>96.51</td>
<td>98.05</td>
<td>98.36</td>
<td>98.67</td>
<td>98.82</td>
<td>99.23</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Models</th>
<th>RF Haar</th>
<th>RF StPy</th>
<th>Boost Haar</th>
<th>Boost StPy</th>
<th>VarGP Haar</th>
<th>VarGP StPy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44.64</td>
<td>44.97</td>
<td>45.03</td>
<td>45.90</td>
<td>53.90</td>
<td>53.23</td>
</tr>
<tr>
<td>2</td>
<td>53.95</td>
<td>60.05</td>
<td>66.00</td>
<td>66.10</td>
<td>72.82</td>
<td>70.62</td>
</tr>
<tr>
<td>3</td>
<td>52.51</td>
<td>65.90</td>
<td>72.00</td>
<td>70.62</td>
<td>78.82</td>
<td>78.82</td>
</tr>
<tr>
<td>4</td>
<td>52.00</td>
<td>65.95</td>
<td>72.10</td>
<td>74.00</td>
<td>80.00</td>
<td>71.33</td>
</tr>
<tr>
<td>5</td>
<td>51.85</td>
<td>65.95</td>
<td>72.10</td>
<td>74.00</td>
<td>80.82</td>
<td>71.54</td>
</tr>
</tbody>
</table>

(b)

Table 3.6: Cumulative results for independent sequential AAMs. The values correspond to the percentage of errors that are below 3% IOD after applying between one and all five models to the small (a) and large (b) CogSys test dataset.

3.4.6 Discussion

Generative AAMs

**Comparison** The results using the generative AAMs on the CogSys datasets are shown at the bottom of tables 3.2 and 3.3. On the large displacement dataset, the POIC algorithm shows the worst overall performance. However, as was noted in Section 2.10, this is unsurprising given that the CogSys dataset contains four individuals of varying skin texture. As expected, the SIC algorithm performs somewhat better due to the joint shape and texture model optimisation. However, this comes at the cost of
3.4 Experiments

greatly increased fitting times (cf. table 3.5a). While the combined AAM shows the best fitting accuracy, it is also notable for the fact that, more so than in the case of the other algorithms, there appears to be a cutoff point after which fitting performance deteriorates rapidly. This reflects the breakdown of the constant linear update approximation (cf. 2.8) in cases where the initial position is too far from the ground truth.

Accuracy Overall, the generative algorithms perform poorly on the larger displacement dataset, demonstrating the susceptibility of the gradient descent algorithms to becoming trapped in local optima, especially when the initialisation is further from the ground truth. Although the same may be said for the higher quantiles of the results on the smaller displacement dataset shown in table 3.2, the lower quantiles are associated with very low error values. This may be attributed to the fact that fitting generative models involves optimising a well-defined non-linear objective function, and the closer the initial points are to the ground truth, the more likely the optimisation is to succeed.

Efficiency Table 3.5a shows the mean training and test timings for the three generative AAM models. Although the POIC algorithm is the most efficient, the combined AAM appears to represent the best trade-off between accuracy and speed. After taking into account the number of iterations, the timings coincide with those reported in [Matthews and Baker, 2004]. As expected, training time is not an issue, and in testing, the SIC algorithm is two orders of magnitude slower than the combined and POIC algorithms.

Discriminative AAMs

Sequences Comparing the mean of the fitting errors for the different sequence training experiments in tables 3.2 and 3.3, it is clear that independent training outperforms coupled training by a large margin. This coincides with the experiments carried out in [Sauer et al., 2011]. It also appears that training sequences using a greater number of pixels for the models at each stage as shown in table 3.1 does not appear to lead to a significant increase in fitting accuracy. Although this seems surprising, it is an encouraging result as training sequences with increasing pixel numbers is often very time-consuming, especially when Haar features are used (cf. table 3.5b).

Regression Models Focusing on the independent sequences and comparing the performance of the regression models across all features, it is clear that the boosting approach outperforms the random forest in most instances but that the best results are
obtained using the variational GP model. Since the VarGP models are trained using the features selected by the boosting algorithm, this implies that the piecewise-constant regression functions used to build the strong learners are unable to capture the full information provided by the selected features. Comparing the results of the single full GP experiment to those of the variational GP experiments, a small but significant increase in performance is evident in both Table 3.2 and Table 3.3. The fact that the results of the variational GP are so close to those of the full GP is a testament to the quality of the variational sparse GP algorithm of Titsias [2009], especially when considering that only 100 inducing inputs were used for the sparse algorithm but the full GP model was trained using 4000 samples. Evidently, the optimisation results in inducing inputs which represent a concise summary of the training data.

Cumulative sequence results  Table 3.6 shows the results after cumulative application of the models in the sequences. The fitting accuracy of the sequences using the random forest and boosting regression algorithms appears to converge at the third model, and in some cases deteriorates slightly after this. However, this is not the case when using the Gaussian process regression models where the value of the success statistic increases monotonously throughout the sequences.

Cross-dataset tests  The results of the cross-dataset experiments reported in Table 3.4 show that although there is a significant deterioration in performance between the within- and cross-dataset experiments, the VarGP algorithm results in the lowest errors and also has the best generalisation performance. Interestingly, we observe a large difference in performance for the random forest algorithm depending on whether it is trained on the BioID or on the XM2VTS dataset.

Features  When comparing the performance of the Haar features to the steerable pyramid features in Table 3.2 and Table 3.3, the general trend shows the Haar features to be superior, especially in the experiments involving the variational GP algorithm. Given the ad hoc nature of the steerable pyramid features, this is unsurprising, especially as the shape and location of the Haar features are optimised whereas the steerable pyramid features are computed at fixed locations. However, in most experiments the values lie within $\sim 5\%$ of each other and there are several instances where the results are close.

Efficiency  Table 3.5b shows the time required for training and testing the different discriminative AAM algorithms. The timings refer to the sequence experiments with
increasing pixels shown in tables 3.2 and 3.3. The long training times reported in the case of Haar features are a consequence of the greatly increased amount of possible feature template shapes and locations in the higher resolution texture samples. The training data contained 4000 instances and in the case of the random forest algorithm, training required almost 17 days. Not shown is the training time for the experiments which use 1000 pixels throughout the sequences. Here, the training of the random forest algorithm only took \(\sim 30\)h. Since the number of steerable pyramid features is small and fixed, training times are shorter, although testing is slower due to the required filtering operations. The extremely long fitting times for the FullGP algorithm are explained by the large number of training instances and highlight the necessity for sparse GP models such as the VarGP algorithm, which yields a substantial increase in performance at only a modest loss in accuracy.

**Comparison**

Comparing the results for the generative AAM algorithms presented in tables 3.2 and 3.3 to the results obtained for the independent discriminative AAM sequences, it is clear that, overall, the discriminative AAM algorithms outperformed the generative AAMs. This is especially true in the case of large displacements. As expected, however, comparing the lower quantiles of the small displacement experiments, the generative AAM models performed as well or better than the discriminative models. Comparing table 3.5a and 3.5b, the random forest and boosting-based models that use Haar features were between one and two orders of magnitude faster when compared to the fastest generative algorithms. However, the comparatively large number of 20 iterations per multiresolution level was used for the generative AAMs where, in practice, often 5 iterations per level are sufficient. The variational sparse GP algorithm was substantially slower, and although it is not suitable for performance-critical applications such as tracking faces in video sequences, its added accuracy makes it attractive for static model-fitting tasks.

### 3.5 Conclusions

In this chapter we discussed a group of regression-based discriminative fitting methods in which the texture information is used directly to update the shape model parameters. To this end, we presented three different regression models based on boosting, random forests and Gaussian processes. Furthermore, we investigated two types of features: Haar features, whose use is commonplace in the field of Computer Vision, as well as a group of features derived from steerable pyramid filters. Inspired by the principled fitting algorithms for generative AAMs discussed in the previous chapter, we developed
two training procedures to build sequences of discriminative AAMs in a bid to increase fitting accuracy and robustness. We evaluated the different combinations of regression models, features and sequence training algorithms in a set of experiments which employed data from three different image databases, and contrasted the results with those obtained using the three most commonly cited generative AAMs, the combined AAM, the simultaneous- and the project-out inverse compositional AAM. Overall, the sequential discriminative AAM algorithms outperformed the generative AAM algorithms, especially in the case of large displacements from the ground truth where the generative AAMs consistently failed. The differences were less extreme for small displacements and the experiments showed that when initialised close to the ground truth, the principled generative AAM algorithms performed as well or better than the discriminative algorithms. The independent sequence training algorithm provided the best results for all discriminative AAMs, outperforming the proposed coupled algorithm by a large margin. The boosting algorithm outperformed the random forest algorithm in most experiments while also taking significantly less time to train and resulting in more compact models. Overall, the variational Gaussian process algorithm resulted in the best fitting accuracy, although fitting took substantially longer when compared to the other algorithms. The use of the proposed steerable pyramid features greatly reduces training time when compared to Haar features. However, fitting accuracy is slightly reduced and the required filtering operations result in slower runtime performance. Haar features are therefore favoured due to their added accuracy and faster runtime performance but at times their advantage over steerable pyramid features is only a slight one, showing there is scope for further improvement. The conclusions drawn from the experiments regarding robustness and accuracy of the models suggest that a hybrid approach to model-fitting could be beneficial. Thus, future work could investigate the use of discriminative AAM sequences prior to applying generative AAMs for added accuracy.
Chapter 4

Non-Linear Probabilistic Models for Face Modelling

The discriminative AAM models discussed in the previous chapter employ non-linear regression to drive the fitting of a linear shape model. Although this approach shows promising results and is the subject of active research, it is not entirely satisfactory from a mathematical standpoint, as it is unclear what is being optimised during fitting. Conversely, the fitting of generative AAM models is grounded in the optimisation of a non-linear objective function which relates the parameters of linear shape and texture models. As a complement to the discriminative AAM, an interesting and comparatively unstudied problem is the use of non-linear dimensionality reduction methods to replace the linear PCA models in generative AAMs. Building on the successful use of Gaussian processes in the previous chapter, we investigate the use of non-linear latent variable models which are based on Gaussian process regression and which were first described by Lawrence [2005]. In the following we present the theory and apply variants of the original Gaussian process latent variable model (GPLVM) to the problem of face modelling with generative AAMs, as well as to understanding facial expressions.

4.1 Probabilistic Unsupervised Dimensionality Reduction

Latent variable models are used for probabilistic linear and non-linear dimensionality reduction. Popular models such as probabilistic PCA [Tipping and Bishop, 1999], factor analysis [Ghahramani and Beal, 2000], generative topographic mapping [Bishop et al., 1998] and Gaussian process latent variable models (GPLVM) [Damianou et al., 2011; Lawrence, 2005; Titsias and Lawrence, 2010] belong to this class of models. In the fol-
4.1 Probabilistic Unsupervised Dimensionality Reduction

Following, we describe the principles unifying such models providing background material that we will build on in Section 5.1.

Assume a multivariate dataset \( Y = \{y_{ni}\}_{n,i=1}^{N,D} \), where \( y_n \in \mathbb{R}^D \) is a data vector.\(^1\) We assume that the high dimensional vector \( y_n \) is produced through a low dimensional space that governs the data generation process. More precisely, there exists a latent vector \( x_n \in \mathbb{R}^Q \), with \( Q \ll D \), that gives rise to the observed vector \( y_n \) according to [Bishop et al., 1998; Lawrence and Moore, 2007; Tipping and Bishop, 1999]

\[
y_{ni} = f_i(x_n; w_i) + \epsilon_{ni}, \quad i = 1, \ldots, D. \tag{4.1}
\]

Here, \( y_{ni} \) is the \( i \)th dimension of the \( n \)th observed data vector, \( f_i \) is a mapping parameterised by \( w_i \) and \( \epsilon_{ni} \) is independent noise. There are three unknowns: i) the latent vectors \( \{x_n\}_{n=1}^N \) contained in the rows of \( X \), ii) the parameters \( \{w_i\}_{i=1}^D \) contained in the rows of \( W \) and iii) the noise term \( E = \{\epsilon_{ni}\}_{n,i=1}^{N,D} \). In a fully probabilistic Bayesian framework all these unknowns are assigned prior distributions and inference over them proceeds by applying Bayes’ rule. In partially Bayesian probabilistic approaches either the parameters \( W \) or the latent vectors \( X \) are estimated using maximum likelihood. This is the approach of most probabilistic unsupervised dimensionality reduction methods, which can be further divided into linear and non-linear methods.

Linear methods such as probabilistic PCA (PPCA) and factor analysis assume a linear mapping such that \( f_i(x_n; w_i) = w_i^T x_n \), where the parameters \( w_i \) are often referred to as factor loadings. They assign a standard normal prior to each latent vector \( x_n \), i.e. \( p(x_n) = N(x_n|0, I) \), and assume white Gaussian noise \( \epsilon \sim N(\epsilon|0, \Sigma) \) with isotropic covariance (i.e. \( \Sigma = \sigma^2 I \)) in the case of PPCA, and diagonal covariance (i.e. \( \Sigma = \text{diag}(\psi) \)) in factor analysis. Standard training approaches estimate the parameters \( W \) by maximum likelihood. More precisely, the latent variables are integrated out so as to obtain the likelihood

\[
p(Y|W) = \prod_{n=1}^{N} \int \prod_{i=1}^{D} p(y_{ni}|x_n, w_i)p(x_n)dx_n, \tag{4.2}
\]

where \( p(y_{ni}|x_n, w_i) = N(y_{ni}|w_i^T x_n, \Sigma) \). Typically this likelihood is maximised with respect to the parameters using the EM algorithm [Dempster et al., 1977; Bishop, 2006, chap.9]. In the case of PPCA the likelihood is maximised by an eigenvalue problem which is equivalent to principal component analysis (PCA) [Tipping and Bishop, 1999].

\(^1\)In our notation, the matrix \( Y \) contains the vectors \( y_n \) in its rows.
Fully Bayesian inference in linear probabilistic dimensionality reduction models is not feasible since assigning a prior $p(W)$ on the parameters $W$ and marginalising is analytically intractable. However, approximate Bayesian solutions based on variational inference [Bishop, 1999; Ghahramani and Beal, 2000] are feasible and unlike the maximum likelihood techniques, they allow for automatic selection of the latent dimensionality.

In many practical problems we are interested in investigating non-linear latent variable models, where the mapping $f_i$ is taken to be a non-linear function. In most cases however, this causes the integral over $\mathbf{x}_n$ in Eqn. (4.2) to become analytically intractable, and it is therefore infeasible to obtain the standard maximum likelihood solution for the parameters $W$. In the following section, we discuss a more flexible non-linear and unsupervised dimensionality reduction model which is based on Bayesian non-parametrics and particularly Gaussian processes [Rasmussen and Williams, 2006].

### 4.2 Gaussian Process Latent Variable Model

We introduced Gaussian processes and discussed their application to regression problems in Section 3.2.3. Within the context of Gaussian process latent variable models, non-linear Gaussian process regression models are used to model the set of functions $\{f_i(\mathbf{x})\}_{i=1}^D$. More precisely, each mapping function $f_i(\mathbf{x})$ is assumed to be an independent random draw from a Gaussian process, so that

$$f_i(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')),$$

(4.3)

where $k(\mathbf{x}, \mathbf{x}')$ is a covariance function that quantifies the smoothness of the random draws. During training, these functions are evaluated on a finite set of latent input vectors $\{\mathbf{x}_n\}_{n=1}^N$ contained in the rows of $\mathbf{X}$, and the Gaussian process prior assumption on the mapping functions induces the following Gaussian prior distributions on the vectors of function values $\{f_{ni}\}_{i=1}^D$, with $\mathbf{F} = \{f_{ni}\}_{n,i=1}^{N, D}$ and $f_{ni} = f_i(\mathbf{x}_n)$:

$$p(\mathbf{F}|\mathbf{X}) = \prod_{i=1}^D N(f_i|0, \mathbf{K}_X),$$

(4.4)

where $\mathbf{K}_X$ is a covariance matrix such that $(\mathbf{K}_X)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. This approach, introduced by Lawrence [2005], is referred to as the Gaussian process latent variable model (GPLVM) and has been used extensively in applications [Huang et al., 2007; Urtasun and Darrell, 2007; Urtasun et al., 2008; Wang et al., 2008].
4.3 Generative AAM with GPLVM

Most current training methodologies for GPLVMs are based on carrying out maximum likelihood or MAP estimation over the latent variables \( \mathbf{X} \). Specifically, both the mapping variables \( \mathbf{F} \) and the latent vectors \( \mathbf{X} \) are unknown quantities which in a fully Bayesian training procedure should be treated as random variables and should be integrated out. However, given the prior over \( \mathbf{F} \) in Eqn. (4.4), and using a standard normal distribution for the prior over \( \mathbf{X} \) as in PPCA and factor analysis, the simultaneous marginalisation of both \( (\mathbf{F}, \mathbf{X}) \) is analytically intractable. Moreover, marginalisation of the latent variables \( \mathbf{X} \) which would allow subsequent maximum likelihood training as in PPCA and factor analysis is also intractable, as the latent variables \( \mathbf{X} \) can appear non-linearly inside the covariance matrix \( K_X \) in Eqn. (4.4). Fortunately, however, the marginalisation of \( \mathbf{F} \) is feasible. This motivated the training procedure proposed by Lawrence [2005], where the mapping \( \mathbf{F} \) is marginalised out to obtain the likelihood

\[
p(Y|X) = \prod_{i=1}^{D} \int \prod_{n=1}^{N} p(y_{ni}|f_{ni})p(f_{i}|X) df_i,
\]

\[
= \prod_{i=1}^{D} \mathcal{N}(Y_{ni}|0, K_X + \Sigma). \tag{4.5}
\]

Point estimates for the latent inputs \( \mathbf{X} \) are then found by maximising the likelihood using a gradient-based optimisation algorithm. When a prior \( p(\mathbf{X}) \) is added as a regularisation penalty in this optimisation, the whole approach reduces to a MAP estimation over \( \mathbf{X} \). Also, by considering more structured priors over \( \mathbf{X} \), interesting latent space regularisation mechanisms may be added [Lawrence and Moore, 2007; Urtasun et al., 2008; Wang et al., 2008]. If linear covariance functions are used for the Gaussian process mappings, the resulting GPLVM model implements linear dimensionality reduction, and is shown to be equivalent to PCA in [Lawrence, 2005]. This leads us to investigate the use of GPLVMs as a replacement for PCA in generative AAMs.

4.3 Generative AAM with GPLVM

Revisiting the generative AAM algorithms discussed in Section 2.7, it is clear that in order for an alternative dimensionality reduction algorithm to be successful, it must provide an explicit mapping into and out of the latent space preserving data proximity in both directions. In other words, the mapping should be bijective. Spectral dimensionality reduction techniques such as Isomap [Tenenbaum et al., 2000] or locally linear embeddings [Roweis and Saul, 2000] preserve data proximity, however they do not
provide a mapping into the latent space. On the other hand, the probabilistic nature of the GPLVM allows the determination of latent space points given a data point via optimisation of the posterior. However, no explicit mapping exists, and the optimisation is susceptible to local optima. Furthermore, as shown by Lawrence and Quiñonero Candela [2006], the GPLVM algorithm encourages the learning of a mapping that preserves data proximity when moving from the latent space to the data space, but that unfortunately the converse is not true. Thus, data points that are close in data space may not be associated with close latent projections. An ad hoc solution to these issues termed the “back-constrained” GPLVM was proposed by [Lawrence and Quiñonero Candela, 2006], where instead of optimising the latent variables $X$, they are parameterised using a radial basis function network (cf. [Bishop, 2006, chap. 6])

$$ x_{ij} = \sum_{n=1}^{N} A_{jn} g(y_i, y_n), $$

whose parameters $A$ are then optimised. Here, $x_{ij}$ is the $i, j$th entry of $X$, which contains the $N$ latent vectors $x_i$ associated with the data vectors $y_i$ in its rows. The values of the RBF kernel $g$ encode the proximity structure around data point $y_i$ and thus the linear combination of the kernel values in Eqn. (4.6) encourages the latent space projections $x_{ij}$ to reflect the structure of the data space. After training, the RBF network mapping may then be used to project data points into the latent space.

The back-constrained GPLVM fulfills the requirements needed for a replacement of the PCA models in a generative AAM algorithm. However, as opposed to linear PCA models, where fitting the model to an instance of a shape or a texture is a convex optimisation problem with a closed-form solution (cf. Eqn. (2.15)), fitting the non-linear GPLVM model to a shape or texture instance represents a non-convex optimisation problem which must be solved using an iterative optimisation algorithm. Since shape and texture are coupled in the combined AAM, using GPLVMs would lead to very complicated training and testing algorithms requiring several stages of optimisation. On the other hand, shape and texture are modelled separately by the inverse-compositional model discussed in Section 2.9, making this group of algorithms more suitable for exploring the use of non-linear GPLVM models. Previous work such as [Batur and Hayes, 2005] and [van der Maaten and Hendriks, 2010] has linked the poor fitting performance of generative AAM algorithms in the generic scenario to the use of linear PCA models for modelling the fundamentally non-linear texture manifold. Therefore, in the following we explore the effects of replacing the linear texture model in an inverse-compositional
AAM with a GPLVM, retaining a linear shape model since the addition of a non-linear shape model is problematic, as it is unclear how the compositional update of the shape parameters could be performed in that case.

### 4.3.1 GPLVM Texture Model

The linear texture model used in the generative AAM algorithms of Section 2.7 allows textures \( G(r; \lambda) \) to be generated as a linear superposition of the texture eigenmodes \( G_j(r) \), as shown in Eqn. (2.19). From a latent variable model perspective, the texture parameters \( \lambda \) are the latent variables associated with the output texture and are embedded in the linear latent space spanned by the eigenmodes \( G_j(r) \). When replacing the linear texture model with a trained GPLVM, the texture image \( G_{nl} \) is a non-linear function \( \zeta \) of the latent variable \( \tilde{x} \):

\[
G_{nl}(\tilde{x}) = \zeta(\tilde{x}) \tag{4.7}
\]

Thus, the latent variable \( \tilde{x} \) may be seen to be the analogue of the linear texture parameters \( \lambda \). In the following, we describe the training of a non-linear GPLVM texture model and derive a closed-form expression for the function \( \zeta \) and explain how to recover the latent variables associated with a given texture. We adhere to the notation used for latent variable models, and consider the matrix \( Y \) to contain flattened texture images in its rows.

**Training**

For the GPLVM texture model, we use an RBF kernel

\[
k(x_i, x_j) = \gamma \exp \left( -\frac{\alpha}{2} \sum_{k=1}^{Q} (x_{ik} - x_{jk})^2 \right), \tag{4.8}
\]

where \( x_i, x_j \) refer to the \( i, j \)th latent vectors, and assume isotropic Gaussian noise with covariance \( \Sigma = \beta^{-1} I \), where the parameter \( \beta = \frac{1}{\sigma^2} \) is the inverse variance. As noted in Lawrence [2005], there is a scale redundancy between the free variables \( X \) and the inverse lengthscale parameter \( \alpha \) of the RBF kernel which may be removed by placing an independent unit Gaussian prior

\[
p(X) = \prod_{n=1}^{N} N(x_n|0, I) \tag{4.9}
\]
over the variables $x_n$. Multiplying with the likelihood of Eqn. (4.5), we obtain the joint distribution

$$p(Y|X, \beta, \theta)p(X) \sim \prod_{i=1}^{D} N(Y_{:,i}|0, K + \beta^{-1} I) \prod_{n=1}^{N} N(x_n|0, I),$$

(4.10)

where we have made the dependency on the kernel hyperparameters $\theta = (\gamma, \alpha)$ and the noise parameter $\beta$ explicit. Taking the negative logarithm, and setting $\tilde{K} = K + \beta^{-1} I$, we obtain the negative log-posterior (without the normalisation)

$$F(X, \theta, \beta) = \frac{DN}{2} \log 2\pi + \frac{D}{2} \log |\tilde{K}| + \frac{1}{2} \text{Tr}(\tilde{K}^{-1}YY^T) + \frac{1}{2} \sum_{n=1}^{N} x_n^T x_n$$

(4.11)

which is used for training the parameters $\{X, \theta, \beta\}$ of the back-constrained GPLVM. For simplicity, we continue to refer to the latent variables $X$ directly, although in actual fact the parameters that are optimised are the coefficients of the back-constraint mapping $\alpha$ shown in Eqn. (4.6). In our implementation, the model parameters are determined using the scaled conjugate gradient algorithm developed by Moller [1993]. In order for the optimisation to be efficient, analytic formulas for the gradients with respect to the parameters must be available and are discussed in the following.

Gradients

The model parameters appear in the objective function via the kernel matrix $\tilde{K}$ and therefore the gradients split into terms according to the chain rule. The inner-most term of the chain rule

$$\frac{\partial F}{\partial \tilde{K}} = \frac{1}{2} \left( \tilde{K}^{-1} Y Y^T \tilde{K}^{-1} - D \tilde{K}^{-1} \right)$$

(4.12)

only depends on the kernel matrix $\tilde{K}$ and may be computed independently of the exact type of kernel used [cf. Lawrence, 2005]. The gradients with respect to the model parameters are given in the following.
Gradient w.r.t. $\gamma$

\[
\frac{\partial F}{\partial \gamma} = \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \frac{\partial \tilde{K}_{ij}}{\partial \gamma} = \frac{1}{\gamma} \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \tilde{K}_{ij}
\]  

(4.13)

(4.14)

Gradient w.r.t. $\alpha$

\[
\frac{\partial F}{\partial \alpha} = \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \frac{\partial \tilde{K}_{ij}}{\partial \alpha} = \frac{1}{2} \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \sum_{k} (x_{ik} - x_{jk})^2 \tilde{K}_{ij}
\]

(4.15)

(4.16)

Gradient w.r.t. $x_{lm}$

\[
\frac{\partial F}{\partial x_{lm}} = \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \frac{\partial \tilde{K}_{ij}}{\partial x_{lm}} = \alpha \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} ((x_{im} - x_{jm}) \delta_{il} - (x_{im} - x_{jm}) \delta_{jl}) \tilde{K}_{ij} = 2\alpha \sum_{n} \frac{\partial F}{\partial \tilde{K}_{ln}} (x_{nm} - x_{lm}) \tilde{K}_{ln}
\]

(4.17)

(4.18)

(4.19)

Here we have used the Kronecker delta $\delta_{ij}$, defined as

\[
\delta_{ij} = \begin{cases} 
1, & i = j \\
0, & \text{otherwise}
\end{cases}
\]

(4.20)

Gradient w.r.t. $\beta$

\[
\frac{\partial F}{\partial \beta} = \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \frac{\partial \tilde{K}_{ij}}{\partial \beta} = -\frac{1}{\beta^2} \sum_{ij} \frac{\partial F}{\partial \tilde{K}_{ij}} \delta_{ij}
\]

(4.21)

(4.22)
4.3 Generative AAM with GPLVM

4.3.2 Texture generation

Given a latent variable \( x_\ast \), the associated output texture \( y_\ast \) is determined by extending the likelihood of the GPLVM of Eqn. (4.5). Including the test data pair \((y_\ast, x_\ast)\), we have

\[
p(Y, y_\ast | X, x_\ast, \beta, \theta) \sim \prod_{i=1}^{D} \mathcal{N}(Y_{N_\ast,s}|0, K_{N_\ast} + \beta^{-1} I),
\]

where

\[
Y_{N_\ast} = \begin{pmatrix}
y_{1,1} & y_{1,2} & \cdots & y_{1,D} \\
y_{2,1} & y_{2,2} & \cdots & y_{2,D} \\
\vdots & \vdots & \ddots & \vdots \\
y_{s,1} & y_{s,2} & \cdots & y_{s,D}
\end{pmatrix}
\]

(4.24)

is the extended data matrix and

\[
K_{N_\ast} = \begin{pmatrix}
k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_\ast) \\
k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_\ast) \\
\vdots & \vdots & \ddots & \vdots \\
k(x_\ast, x_1) & k(x_\ast, x_2) & \cdots & k(x_\ast, x_\ast)
\end{pmatrix}
\]

(4.25)

is the extended kernel matrix. Conditioning on the training data \( Y \) using Eqn. C.7 yields

\[
p(y_\ast | Y, X, x_\ast, \beta, \theta) \sim \mathcal{N}(y_\ast | Y^T K_{NN}^{-1} k_{N_\ast}, (k_{\ast\ast} - k_{N_\ast}^T K_{NN}^{-1} k_{N_\ast}) I_D)
\]

(4.26)

from which we obtain the predictive mean

\[
\mu_\ast = Y^T K_{NN}^{-1} k_{N_\ast}
\]

(4.27)

which represents the best estimate of the output texture \( y_\ast \) given the latent space point \( x_\ast \). Due to the presence of the non-linear RBF kernel, the predictive mean is a non-linear function of the latent variables \( \{X, x_\ast\} \) and is equivalent to the function \( \zeta \) of Eqn. (4.7). Furthermore, since the kernel \( K \) is shared across the \( D \) output dimensions and due to the factorised likelihood, the covariance matrix is spherical with variance \( k_{\ast\ast} - k_{N_\ast}^T K_{NN}^{-1} k_{N_\ast} \). Here, \( k_{\ast\ast} \) refers to the bottom right entry, and \( k_{N_\ast} \) to the right column of the kernel matrix \( K_{N_\ast} \).
4.3 Generative AAM with GPLVM

4.3.3 Latent space projection

As a precursor to fitting a generative AAM using gradient descent, the initial values of the shape and texture model parameters must be chosen. While the shape is commonly initialised to the mean, using the “back-constrained” GPLVM for the texture model allows the latent space point $x_*$ to be determined using the learned back-constraint mapping of Eqn. (4.6). Furthermore, this initialisation may be improved upon by optimising the negative log-posterior

$$
\hat{F}(x_*) = \frac{DN}{2} \log 2\pi + \frac{D}{2} \log |\tilde{K}_{N*}| + \frac{1}{2} Tr(\tilde{K}_{N*}^{-1} Y_{N*} Y_{N*}^T) + \frac{1}{2} \sum_{n=1}^{N} x_n^T x_n + \frac{1}{2} x_*^T x_* \quad (4.28)
$$

with respect to $x_*$, while keeping the trained parameters $\{X, \theta, \beta\}$ fixed.

4.3.4 Non-Linear AAM

Replacing the linear texture model with the non-linear GPLVM texture model, the generative AAM objective function of Eqn. (2.25) becomes

$$
\min_{x, p, q} \sum_{r \in s_0} \| E(r; x, p, q) \|^2; \quad (4.29)
$$

where

$$
E(r; x, p, q) = G_{nl}(r; x) - I(N(W(r; p); q)) \quad (4.30)
$$

and the remaining variables are as defined in Section 2.7.1.

In order to optimise the objective function in Eqn. (4.29), we adapt the SIC algorithm of Section 2.9 to work with the GPLVM texture model. To this end, we require the steepest descent images

$$
D_{nl}(r) = \left( \frac{\partial G_{nl}(r; \hat{x})}{\partial r}, \frac{\partial W}{\partial p} \bigg|_{p=0}, \frac{\partial G_{nl}(r; x)}{\partial \hat{x}} \right) \quad (4.31)
$$

with respect to the non-linear texture model $G_{nl}$ and the latent space point $\hat{x}$. The first term in Eqn. (4.31) requires no modification, since it is composed of the gradient of the texture image $G_{nl}$ and the Jacobian of the warp $W$, which is induced by the linear shape model. The second term contains the gradients with respect to the latent space point $\hat{x}$ which are obtained from the definition of the non-linear texture model.
4.3 Generative AAM with GPLVM

in Eqn. (4.7) and Eqn. (4.27). Given the latent space vector \( \hat{x} \), the associated texture vector \( \hat{y} \) is

\[
\hat{y} = Y^T K_{NN}^{-1} k_N \hat{x}.
\]  

(4.32)

Noting that we are using an RBF kernel, the derivative of the \( m \)th component of the texture vector with respect to the \( i \)th component of \( \hat{x} \) becomes

\[
\frac{\partial y_m}{\partial \hat{x}_i} = \sum_{p,l} Y_{lm}^T K_{lp}^{-1} \frac{\partial}{\partial \hat{x}_i} \gamma \exp \left( -\frac{\alpha}{2} \sum_k (x_{pk} - \hat{x}_k)^2 \right)
\]  

(4.33)

\[
= \alpha \sum_{p,l} Y_{lm}^T K_{lp}^{-1} (x_{pi} - \hat{x}_i) k_p
\]  

(4.34)

Reassembling the derivative vectors \( \frac{\partial y}{\partial \hat{x}} \) into the texture reference frame, we obtain the steepest descent images \( \frac{\partial G_{nl}(r; x)}{\partial \hat{x}} \). Given the steepest descent images \( D_{nl} \), the fitting of a non-linear AAM may be performed according to the SIC algorithm of Section 2.9. As an extension to this algorithm, the log posterior of Eqn. (4.28) may be optimised in a separate step at each iteration to improve convergence. In order to ease notation, in the following the linear SIC AAM algorithm is referred to as “L-SIC”, the GPLVM SIC algorithm is called “G-SIC” and the extension using optimisation of the posterior is named “G-OptSIC”.

4.3.5 Experiments

In order to investigate the performance of the non-linear GPLVM algorithms, we performed a set of quantitative experiments on the CogSys dataset used in Section 3.4.

Training  In all the experiments, the dimensionality of the linear shape models was set to 20 dimensions. The GPLVM models associated with the texture models were trained on a total of 200 images of four individuals from the CogSys dataset and separate models were constructed using latent spaces of 2, 5 and 10 dimensions. The latent spaces were initialised using linear PCA and the signal-to-noise ratio (SNR) of the model was initialised to 10 by setting the kernel and noise hyperparameters relative to the variance of the data. Furthermore, we placed bounds on the SNR in order to avoid over- and underfitting of the data, so that the hyperparameters were constrained to remain within a factor of 10 of their initial values. Following initialisation, the non-linear optimiser was run for 1000 iterations. For the sake of comparison, L-SIC models were built using the same dimensionality settings.
4.3 Generative AAM with GPLVM

Testing  The test data was generated from an independent, randomly selected set of 200 frames taken from the video sequences used to generate the training data. The behaviour of the GPLVM-based algorithms was evaluated and compared to the L-SIC AAM in two experiments. In the first experiment, the test data was generated by rigidly aligning the mean shape to the ground-truth points. This resulted in an initial error distribution which was very close to the ground-truth and which was used to measure the stability of the non-linear AAM algorithms. Small perturbations of the ground-truth pose parameters were used to generate the test data for the second experiment, resulting in an initial error distribution similar to the experiments of Section 3.4. All algorithms used 100 fitting iterations, and the shape and texture errors for the first fitting experiments are shown in tables 4.1a and 4.1c. The shape error was calculated as in Eqn. (3.34), whereas the texture error was given as the percentage of mean deviation per pixel from the ground truth texture. While all the L-SIC models improved on the initial error distribution, the shape and texture error after fitting was increased for a significant proportion of the test data in the case of the 2D and 5D G-SIC and G-OptSIC models. Although this remained true for the 10D G-SIC algorithm, the best fitting results in this experiment were obtained for the 10D G-OptSIC algorithm. This suggests that when near the ground truth position, the fitting performance of the non-linear AAM models improves with the number of dimensions used for the non-linear texture model. However, increasing the dimensionality beyond 10 dimensions did not result in any further significant improvement in these experiments. Comparing tables 4.1b and 4.1d, we find a similar pattern, although it is clear that a significant proportion of fitting experiments diverged. In particular, this happened when using the non-linear G-SIC and G-OptSIC algorithms, where divergence was drastic and resulted in undefined behaviour in the texture models when the shape shrunk to zero or expanded to infinity. Timings of the algorithms used in the experiments are shown in table 4.2.

The 2D latent spaces associated with the texture models used in the experiments are visualised in Figure 4.1, where 4.1a shows the latent space associated with the GPLVM models and where the latent space associated with a 2D linear texture model is illustrated in 4.1b. The continuous latent space visualisation in Figure 4.1a is obtained by evaluating the GPLVM mapping at the points of a regular grid and recording the negative logarithm of the variance. Thus, red areas correspond to locations near the training data, where the variance of the GPLVM mapping is low. Both latent spaces clearly separate the textures of the four individuals. The similarity of the latent spaces is apparent and is a direct result of using the linear latent space to initialise the training of the GPLVM. While optimising the latent space, the back-constraint mapping encourages
errors are included in the tables. The corresponding measure of Eqn. (3.34). Due to the presence of outliers, the 95% confidence interval (b) show the algorithms where the AAMs contained texture models of 2, 5 and 10 dimensions. (a) and Table 4.1: Fitting results for the Linear SIC AAM and the non-linear GPLVM algorithms where the AAMs contained texture models of 2, 5 and 10 dimensions. (a) and (d), where the entries represent the mean percentage of deviation from the ground truth pixel values. As is obvious from the quantiles of the initial error distribution, the experiments in (a) and (c) start very close to the ground truth, whereas small pose displacements are used in (b) and (d).

<table>
<thead>
<tr>
<th>L-SIC</th>
<th>Error Quantiles</th>
<th>G-OptSIC</th>
<th>Error Quantiles</th>
<th>G-SIC</th>
<th>Error Quantiles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25%</td>
<td>50%</td>
<td>75%</td>
<td>95%</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td>Init</td>
<td>0.30</td>
<td>0.38</td>
<td>0.46</td>
<td>0.56</td>
</tr>
<tr>
<td>2</td>
<td>0.14(1)</td>
<td>0.19(2)</td>
<td>0.23(1)</td>
<td>0.31(4)</td>
<td>0.25(2)</td>
</tr>
<tr>
<td>5</td>
<td>0.08(0)</td>
<td>0.11(1)</td>
<td>0.14(2)</td>
<td>0.23(3)</td>
<td>0.11(1)</td>
</tr>
<tr>
<td>10</td>
<td>0.08(1)</td>
<td>0.10(1)</td>
<td>0.13(1)</td>
<td>0.22(4)</td>
<td>0.10(1)</td>
</tr>
<tr>
<td>G-SIC</td>
<td>2</td>
<td>0.17(83)</td>
<td>0.26(83)</td>
<td>0.38(83)</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.10(1)</td>
<td>0.16(1)</td>
<td>0.22(2)</td>
<td>0.41(8)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.08(1)</td>
<td>0.12(1)</td>
<td>0.19(3)</td>
<td>0.52(23)</td>
</tr>
<tr>
<td>G-OptSIC</td>
<td>2</td>
<td>0.37(6)</td>
<td>0.73(15)</td>
<td>1.66(63)</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.07(1)</td>
<td>0.09(1)</td>
<td>0.15(2)</td>
<td>15.50(33)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.06(0)</td>
<td>0.09(1)</td>
<td>0.12(1)</td>
<td>0.20(2)</td>
</tr>
</tbody>
</table>

Table 4.1: Fitting results for the Linear SIC AAM and the non-linear GPLVM algorithms where the AAMs contained texture models of 2, 5 and 10 dimensions. (a) and (b) show the shape error after fitting, where the values are generated using the error measure of Eqn. (3.34). Due to the presence of outliers, the 95% confidence interval of the Maritz-Jarrett standard error was large in some cases so that not all standard errors are included in the tables. The corresponding texture errors are shown in (c) and (d), where the entries represent the mean percentage of deviation from the ground truth pixel values. As is obvious from the quantiles of the initial error distribution, the experiments in (a) and (c) start very close to the ground truth, whereas small pose displacements are used in (b) and (d).
4.3 Generative AAM with GPLVM

<table>
<thead>
<tr>
<th>L-SIC</th>
<th>G-SIC</th>
<th>G-OptSIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Time (ms/iteration)</td>
<td>3.8</td>
<td>4.7</td>
</tr>
</tbody>
</table>

Table 4.2: Timings in milliseconds per iteration for the L-SIC, G-SIC and G-OptSIC algorithms used in the experiments detailed in the tables in Figure 4.1.

the latent space points to be placed according to the similarity of the training data. However, contrary to linear PCA, the objective function is non-linear, and therefore the construction of the latent space is subject to the effect of local optima.

The latent points obtained from an individual fitting experiment of the type used to generate tables 4.1a and 4.1c are shown as sequences of points and crosses in Figures 4.1a and 4.1b, where white corresponds to the first iteration and black to the last. The ground-truth texture of the experiment is shown on the right in Figure 4.1c, and is similar to frame 89 of the training dataset, which is the top-right image in Figure 4.1a. The sequence of round greyscale points in Figure 4.1a corresponds to the G-SIC algorithm, and it is clear that although the search is started very near the ground truth, the algorithm converges to a local optimum which results in the rather poor reconstruction shown on the left in 4.1c. Fitting using the L-SIC algorithm converges quickly, however the latent space points shown in Figure 4.1b remain far from the ground truth. Although the L-SIC reconstruction in Figure 4.1c is an improvement on the G-SIC reconstruction, the facial traits appear slightly distorted and the loss of high-frequency information caused by the low dimensionality of the PCA latent space makes the texture look wooden and somewhat artificial. The most successful fit is obtained using the G-OptSIC algorithm, of which the corresponding latent points are displayed as crosses in Figure 4.1a and the associated reconstruction is given in the second image from the right in Figure 4.1c.

Discussion Although this experiment proves that a 2D GPLVM texture model is able to produce more realistic images when compared to a linear model of the same dimensionality, it must be noted that this comes at the price of much reduced fitting robustness, even when the search starts close to the ground truth (cf. table 4.1a). The experiments in table 4.1 show that increasing the dimensionality of the non-linear models also increases their robustness. This behaviour is echoed in Figure 4.2, which shows the reconstructed image for the different AAM algorithms after increasing the number of dimensions in the texture model to 10. Now, the G-SIC algorithm also converges to

77
Figure 4.1: (a) Latent texture space visualisation of a 2D GPLVM model and (b) the corresponding linear texture space after training on 100 texture samples from video sequences of four individuals from the CogSys dataset. An image corresponding to a particular frame in the video is shown for each of the four individuals. The colouring in figure (a) represents the negative logarithm of the predictive variance of the GPLVM when evaluated at the latent points shown (cf. Eqn. 4.26). The plots include a fitting test, where the AAMs were fitted starting from the mean shape. The greyscale points and crosses in (a) correspond to the latent points associated with each iteration of the G-SIC and G-OptSIC fitting algorithms, where white represents the first and black the last iteration. Equivalently, the greyscale points in (b) represent the latent points observed during fitting of a linear AAM with the L-SIC algorithm. The reconstructed images corresponding to the final (i.e. the black) latent points are shown in (c) and the original image is included on the right for comparison.
Figure 4.2: Employing 10 dimensional texture models in the fitting experiments of Figure 4.1 results in the reconstructions shown above. Comparing to Figure 4.1c, the performance of the G-SIC and L-SIC algorithms is much improved.

a good approximation of the ground truth as does the linear reconstruction, although on closer inspection the linear reconstruction still appears smoothed and less natural when compared to the non-linear GPLVM-based reconstruction. The poor robustness of the GPLVM-based algorithms for low-dimensional latent spaces may be explained by noting that the optimisation of the GPLVM models attempts to provide the best possible reconstruction of the output data given the latent space. If the latent space dimensionality is low, depending on the complexity of the output data this can only be achieved by creating complex non-linear mappings which increase the difficulty of fitting to unseen test data. Furthermore, it must be noted that the values and bounds chosen for the kernel and noise hyperparameters of the GPLVM are critical to successful model fitting. While the variance of the training data provides a strong heuristic for setting the values of these parameters, the bounds are determined empirically for the experiments in this chapter. Setting the bounds too loosely, e.g. by only requiring positivity of the parameters, results in the exact interpolation of the data by the GPLVM and leads to very poor generalisation performance. On the other hand, setting the bounds too strictly results in a rigid model incapable of expressing significant non-linearities. In practice, we found that allowing one order of magnitude of deviation from the parameter initialisation provided the best results. As was noted in the case of GP regression models in Chapter 3.2, the non-parametric nature of GP-based models leads to poor scaling of the computational complexity with the number of training data. Inspection of the predictive mean equation reveals the presence of the inverse of the kernel matrix, $K_{NN}^{-1}$. Luckily, however, this only depends on the training data and may be precomputed for performing predictions. Unfortunately, this is not the case when training the model, and computation of the gradients of the objective function requires several potentially
large kernel matrices to be inverted, so that training quickly becomes infeasible on larger datasets. In Chapter 5.1 we extend the idea of “back-constrained” GPLVMs and make use of recently proposed sparse formulations of the GPLVM which help to overcome this problem.

**Implementation** All the models used in this chapter were implemented in C++. For the linear inverse-compositional AAM, the same implementation as described in Section 3.4.5 was used. The GPLVM models were implemented from scratch in C++ as part of the framework which was developed for Gaussian process regression described in Section 3.4.5. Parts of Tim Cootes’s non-public shape and appearance modelling libraries were used in the implementation of the GPLVM AAM. In order to implement bound constraints using the gradient descent optimiser by Møller [1993] which only solves unconstrained optimisation problems, we expressed every constrained variable as a function of an unconstrained variable. Therefore, if a variable $\theta$ was constrained to lie within the bounded interval $[a, b]$, it was parameterised by an unconstrained variable $\phi$ via the invertible transformation

$$\theta = a + (b - a) \logit^{-1}(\phi) \quad (4.35)$$

which uses the logistic function $\logit^{-1}(x) = \frac{1}{1 + \exp(-x)}$. Before each iteration of the optimisation, the constrained parameter $\theta$ which was used to evaluate the objective function was then transformed to $\phi$ using the inverse transformation

$$\phi = \logit \left( \frac{\theta - a}{b - a} \right) \quad (4.36)$$

where $\logit(x) = \log \left( \frac{x}{1 - x} \right)$. Although $\phi$ was passed to the optimiser, the analytic gradient equations were calculated with respect to $\theta$ and thus the transform had to be accounted for in the gradients. This was easily accomplished by using the chain rule to include the extra term $\frac{\partial \theta}{\partial \phi}$. This procedure, a description of which is given in Lawrence [2005], is commonly used within the field of Machine Learning to solve constrained optimisation problems. In our experience with Gaussian processes and Gaussian process latent variable models, it worked at least as well as more advanced interior point optimisers with which we performed some early experiments.

**Conclusions** The GPLVM framework provides a flexible vehicle for investigating non-linear appearance models and in this chapter we showed how GPLVMs can be adapted
to facilitate the development of non-linear simultaneous inverse-compositional AAM models, analogous to the linear models described by Matthews and Baker [2004]. The natural appearance of the GPLVM-based models is a significant advantage when compared to linear models and could benefit applications where human-computer interaction takes place via avatars, since it is often the subtle inaccuracies in facial appearance that disconcert the user. On the other hand, the experiments in this chapter also highlight the innate difficulties of working with non-linear models, and show that in their current form these algorithms are unlikely to replace linear AAMs. However, the probabilistic nature of the GPLVM opens avenues for further work and an interesting direction of further research could be the integration of non-trivial prior distributions over the latent space as has been proposed in the GPLVM literature [cf. Urtasun and Darrell, 2007; Wang et al., 2008]. More specifically, both improved temporal coherence [Wang et al., 2008] and semantic clustering [Urtasun and Darrell, 2007] could be employed to increase robustness of fitting provided a good latent space initialisation is given.
Chapter 5

Supervised dimensionality reduction using Gaussian processes

The original GPLVM framework that was employed in the previous chapter suffers from several shortcomings. Firstly, the latent variables $X$ are free parameters of the model which must be optimised during training. This results in the computational cost of training becoming prohibitive as the number of training data increases. Furthermore, the dimensionality of $X$ is a tunable free parameter of the model, the value of which must be found empirically. This is a limitation of the maximum likelihood (ML) or maximum a posteriori (MAP) methods used to train the model. Secondly, although label information is often available in experiments such as those described in the previous chapter, the GPLVM in its original form is an unsupervised dimensionality reduction algorithm and therefore does not integrate label information into the construction of the latent space. Some extensions such as [Urtasun and Darrell, 2007] and [Gao et al., 2011] have been proposed for supervised or discriminative training. However, these methods employ the MAP or ML framework for training the model and are prone to the problems mentioned above. Lastly, although the use of “back-constraints” adds the required regularity properties to the GPLVM mapping, it represents a purely ad hoc solution which is not grounded in the probabilistic framework of the GPLVM. Recently, Titsias [2009] developed a variational Bayesian approach for Gaussian process latent variable models which allows automatic determination of the dimensionality of the latent space by leveraging the variational sparse Gaussian process regression framework discussed in Section 3.2.4 to approximately integrate out the latent variables $X$. This work was extended by Damianou et al. [2011] where the addition of temporal latent space priors was investigated. In the following, we propose a probabilistic extension of the GPLVM
5.1 Supervised Variational GPLVM

Let us assume that together with the input data \( Y \), we are given some output data \( Z = \{ z_{nk} \}_{n,k}^{N,K} \), where each row \( z_n \) is a \( K \)-dimensional vector that represents the outputs associated with the data input point \( y_n \). \( Z \) can quantify classification labels, regression real values or any other desired labelling information attached to the input data \( Y \). The general structures of the Bayesian GPLVM models discussed in this thesis are represented as graphical models in Figure 5.1, where shaded nodes represent observed variables and unshaded nodes refer to latent variables. The unsupervised GPLVM of Titsias and Lawrence [2010] is shown in Figure 5.1a. Figure 5.1b shows the supervised GPLVM, where \( Z \) represents label information associated with the data \( Y \). This graphical model implies that we seek a latent representation \( X \) that depends on the input data \( Y \) and which subsequently generates or reconstructs the output data \( Z \). If \( Z \) is chosen to be equal to the input data \( Y \), we obtain the “autoencoder” GPLVM shown in Figure 5.1c where the inputs are reconstructed via a low-dimensional set of hidden variables \( X \). This model represents a principled implementation of the ad hoc “back-constraint” idea used in the previous chapter. As is obvious from Figure 5.1, the supervised GPLVM and autoencoder GPLVM are formally equivalent, the only difference being the interpretation of the output data. Thus, no extra treatment is required for the autoencoder GPLVM and in the following we concentrate on the supervised GPLVM. In order to implement the model described by the graphical model in Figure 5.1b, we need to define

Figure 5.1: Graphical models of Bayesian GPLVMs: (a) GPLVM for unsupervised learning, (b) GPLVM for supervised or discriminative learning and (c) autoencoder GPLVM.

which enables supervised learning and which formalises the notion of “back-constraints” by developing a training algorithm inspired by the variational inference procedure of Damianou et al. [2011]. We evaluate the algorithm by applying it to the problem of recognising emotions from facial expressions.
the conditional probability distributions \( p(X|Y) \) and \( p(Z|X) \). In the following, this is achieved by adapting the variational Bayesian framework proposed by Damianou et al. [2011].

To define \( p(X|Y) \), we assume that the latent inputs \( X \) are finite realisations of non-linear latent functions indexed by the input data. More precisely, we define \( Q \) latent functions \( \{x_q(y)\}_{q=1}^{Q} \) which consist of random draws from Gaussian processes indexed by the input data variable \( y \) so that
\[
x_q(y) \sim \mathcal{GP}(0, k(y_i, y_j)).
\]

Thus, for the random variables \( X \) we have \( (X)_{nq} = x_q(y_n) \), and the prior distribution over \( X \) becomes
\[
p(X|Y) = \prod_{q=1}^{Q} N(X_{:,q}|0, K_Y),
\]
where \( X_{:,q} \) is the \( q \)th column of \( X \) and the covariance matrix \( K_Y \) is such that \( (K_Y)_{ij} = k(y_i, y_j) \). Intuitively, the above assumptions say that for two training input data points \( y_i \) and \( y_j \), which are close to one another\(^1\), the corresponding low-dimensional latent vectors \( x_i \) and \( x_j \) must also be close to one another. Furthermore, each \( x_i \) can be viewed as a feature vector extracted from the input data point \( y_i \). Importantly, such features are extracted in a linear or non-linear manner depending on the properties of the kernel function used. The use of a linear kernel,
\[
k(y_i, y_j) = \sigma^2 y_i^T y_j,
\]
may be interpreted as restricting the latent space to a linear manifold [Rasmussen and Williams, 2006]. A very common choice of kernel which allows smooth functions to be sampled from the prior is the radial basis function (RBF), which we discussed in Eqn. (3.14), and which we include here for completeness:
\[
k(y_i, y_j) = \sigma^2 \exp \left(-\frac{\alpha}{2} r_{ij}^2 \right).
\]

Here, \( r_{ij} = ||y_i - y_j||^2 \), and \( \alpha \) is an inverse lengthscale parameter which determines the degree of smoothness of the functions drawn from the prior.

Finally, the set of Matérn covariance functions is a kernel family parameterised by two positive parameters \( l \) and \( \nu \) [Rasmussen and Williams, 2006]. Variation of the

\(^1\)Here, the proximity measure is given by the choice of kernel function.
5.1 Supervised Variational GPLVM

parameter \( \nu \) results in kernel functions of different degrees of smoothness where the RBF kernel is recovered in the case \( \nu \to \infty \). A commonly used kernel is the kernel obtained by setting \( \nu = \frac{3}{2} \):

\[
k_{3/2}(y_i, y_j) = \left( 1 + \frac{\sqrt{3}r_{ij}}{l} \right) \exp \left( -\frac{\sqrt{3}r_{ij}}{l} \right),
\]

(5.5)

The functions drawn from a Gaussian process using this kernel are significantly rougher when compared to the RBF kernel.

The remaining part of the model is quantified by the conditional probability \( p(Z|X) \) and is defined exactly as \( p(Y|X) \) in the unsupervised GPLVM in Eqn. (4.5). However, in this case, the mapping functions \( \{f_k(x)\}_{k=1}^{K} \) generate the output data \( Z \). Given that these output data are real-valued, we have

\[
p(Z|X) = \prod_{k=1}^{K} N(Z_{i,k}|0, K_X + \Sigma),
\]

(5.6)

where \( Z_{i,k} \) is an \( N \)-dimensional vector storing the \( k \)th outputs. Since the model is intended to perform non-linear dimensionality reduction, and we wish to automatically determine the dimensionality of the data, we employ an RBF kernel for \( K_X \) which implements the concept of automatic relevance determination (ARD):

\[
k(x_i, x_j) = \sigma^2 \exp \left( -\frac{1}{2} \sum_{q=1}^{Q} \alpha_q (x_{iq} - x_{jq})^2 \right)
\]

(5.7)

\[
= \sigma^2 \prod_{q=1}^{Q} \exp \left( -\frac{1}{2} \alpha_q (x_{iq} - x_{jq})^2 \right)
\]

(5.8)

Comparing this kernel to the standard RBF kernel shown in Eqn. (3.14), it is clear that the ARD kernel contains an inverse lengthscale hyperparameter \( \alpha_q \) for each latent dimension \( q \) and therefore the exponential factorises across the latent dimensions. These hyperparameters may be interpreted as a measure of “relevance” of the corresponding dimension. If, for instance, the value of the parameter \( \alpha_{q'} \) converges to zero during optimisation of the objective function, the argument of the corresponding exponential function in Eqn. (5.8) becomes zero irrespective of the inputs \( \{x_{iq'}, x_{jq'}\}_{i,j=1}^{N} \). Thus, the exponential evaluates to one and no longer contributes to the covariance function. In other words, the dimension \( q' \) has become irrelevant. Unfortunately, however, automatic relevance determination is not possible within the ML or MAP framework.
where changing the latent dimensionality increases the number of free parameters of the model in proportion to the number of training data. Since adding free parameters always improves the likelihood, optimisation cannot lead to automatic selection of the dimensionality [Damianou et al., 2011]. The approach proposed by Titsias [2009] and Damianou et al. [2011] addresses this problem by approximately marginalising the latent variables \( X \) so that the remaining parameters and hyper-parameters of the model can be determined by type-II maximum likelihood. Marginalising the latent variables greatly increases the amount of data relative to the number of remaining parameters, and it is therefore assumed that type-II maximum likelihood estimation results in accurate point estimates and that a fully Bayesian treatment is unnecessary. In the following, we discuss a similar variational Bayesian training procedure where the use of an RBF-ARD kernel allows the removal of redundant latent dimensions and thus to determine the latent dimensionality automatically.

### 5.2 Variational inference

In order to be able to determine the parameters and hyperparameters of the model shown in Figure 5.1b via type-II maximum likelihood, we require the marginal likelihood which is obtained by integrating out the latent variables \( X \) and the latent function variables \( F \):

\[
p(Z|Y) = \int \prod_{k=1}^{K} \prod_{n=1}^{N} p(z_{nk}|f_{nk})p(F|X)p(X|Y)\,dFdX.
\]  

(5.9)

Here, \( f_{nk} = f_k(x_n) \) is the latent function \( f_k \) instantiated at the latent point \( x_n \), \( p(F|X) \) is the GP prior over the latent functions of Eqn. (4.4), and \( p(X|Y) \) is the prior over the latent variables \( X \) defined in Eqn. (5.2). However, the integration over \( X \) is intractable, since \( X \) appears non-linearly in the inverse of the covariance matrix of the GP prior \( p(F|X) \). Titsias and Lawrence [2010] address this issue by employing the variational sparse Gaussian process framework developed by Titsias [2009] which we discussed in Chapter 3.2.4. Thus, the same “data augmentation” trick is employed according to which the GP prior \( p(F|X) \) is consistently augmented with inducing variables \( U = \{u_k\}_{k=1}^{K} \), where \( u_k \in \mathbb{R}^M \) are auxiliary values drawn from the random function \( f_k(x) \) so that

\[
p(F,U|X) = p(F|U,X)p(U).
\]  

(5.10)
Here, \( p(U) \) is a marginal GP prior that does not depend on the latent variables \( X \) but on a set of \( M \leq N \) free-to-optimise inputs \( X_U \). Following this, variational inference is applied in the space of random variables \((F, X, U)\) as opposed to in the initial space \((F, X)\). The variational distribution \( q(F, X, U) \) is chosen to be

\[
q(F, X, U) = p(F|U, X)q(U)q(X),
\]

where crucially, \( p(F|U, X) \) is the same distribution that appears in the augmented GP prior of Eqn. (5.10),

\[
q(X) = \prod_{i=1}^{Q} N(X_{i}|\mu_i, S_i),
\]

and \( q(U) \) is a free-form distribution which is determined analytically within the variational inference framework. Notice that (as in [Damianou et al., 2011]) the variational distribution \( q(X) \) factorises over the dimensions of the latent variables \( X \) and not over the latent variables themselves. Thus, the prior over \( X \) allows correlations between the latent space points \( X \) to be modelled. After adding the inducing variables \( U \) to Eqn. (5.9), we insert a trivial unity term expressed using the variational distribution of Eqn. (5.11):

\[
p(Z|Y) = \int q(F, U, X)p(Z, F, U, X|Y)q(F, U, X)dFdUdX.
\]

By taking the log and applying Jensen’s inequality\(^1\), we then obtain a lower bound \( F(q, \theta) \) for the log of the marginal likelihood of Eqn. (5.9):

\[
\log p(Z|Y) \geq \int q(F, U, X) \log \frac{p(Z, F, U, X|Y)q(F, U, X)}{q(F, U, X)}dFdUdX.
\]

\(^1\)For a convex function \( f: \mathbb{R} \to \mathbb{R} \), the following holds for all \( x_n \in \mathbb{R}, \lambda_n \in [0, 1] \)

\[
f \left( \sum_n \lambda_n x_n \right) \leq \sum_n \lambda_n f(x_n).
\]

In probability theory, this implies

\[
f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)].
\]
Solving this integral is challenging. However, the steps required are analogous to those detailed in the appendix of [Damianou et al., 2011]. After inserting the variational distribution of Eqn. (5.12), the lower bound splits into a likelihood and a prior term

\[
F(q, \theta) = \int_p(F|U, X) \log p(Z|F) q(U|X) dF dU dX + \int q(U) \log \frac{p(U)}{q(U)} dU
\]

which have an analytic solution

\[
F(q, \theta) = \frac{1}{2} \sum_{d=1}^{D} \log \left( \frac{\beta^{\frac{N}{2}} |K_{MM}|^{\frac{1}{2}}}{(2\pi)^{\frac{N}{2}} |\beta \Psi_2 + K_{MM}|^{\frac{1}{2}}} \right) - \frac{1}{2} y_d^{T} W y_d - \frac{\beta \psi_0}{2} + \frac{\beta}{2} Tr(K_{MM}^{-1} \Psi_2)
\]

\[
- \frac{Q}{2} \log |K_Y| - \frac{1}{2} \sum_{q=1}^{Q} \left[ Tr(K_{Y}^{-1} S_q) + Tr(K_{Y}^{-1} \mu_q \mu_q^{T}) \right] + \frac{1}{2} \sum_{q=1}^{Q} \log |S_q|.
\]

Here, we have

\[
W = \beta I_N - \beta^2 \Psi_1 (\beta \Psi_2 + K_{MM})^{-1} \Psi_1^{T},
\]

and

\[
\psi_0 = Tr \left( (K_{NN})_{q(X)} \right)
\]

\[
\Psi_1 = \langle K_{NM} \rangle_{q(X)}
\]

\[
\Psi_2 = \langle K_{MN} K_{NM} \rangle_{q(X)}.
\]

Furthermore, \(K_{MM}\) and \(K_{NM}\) represent the kernel of the GP prior of Eqn. (4.4) evaluated using the set of the \(M\) inducing inputs \(X_U\) and the \(N\) means of the variational distribution \(q(X)\) of Eqn. (5.12), whereas \(K_Y\) is the kernel of the GP prior of Eqn. (5.2) evaluated at the data points \(Y\). In Eqn. (5.20) we have made the dependency on the parameters \(\theta = \{\theta_f, \theta_p, \theta_{\text{var}}\}\) explicit. Assuming we are using an RBF-ARD kernel for the GPLVM and an RBF kernel for the latent space prior, the hyperparameters, with respect to which the lower bound is optimised, are

\[
\theta_f = \{\sigma_f, \{\alpha_i\}_{i=1}^{Q}, \beta, X_U\}
\]

\[
\theta_p = \{\sigma_p, \alpha_p\}
\]

\[
\theta_{\text{var}} = \{\mu_i, S_i\}_{i=1}^{Q}.
\]
5.2 Variational inference

Here, $\beta$ is the inverse variance of the Gaussian noise, $\{\mu_i, S_i\}$ are the means and covariances of the variational distribution $q(X)$, and the remaining hyperparameters are the variances and inverse lengthscales of the kernel functions. Although the inducing inputs $X_U$ are not strictly kernel parameters, we have included them in $\theta_f$ since they always appear within the context of the kernel. It is important to note that, in practice, a small constant amount of white noise is added to the kernel $K_Y$ in order to guarantee positive definiteness in the presence of numerical errors. This is a common procedure in numerical linear algebra and is referred to as “jitter”. The variational means and covariances are determined by setting the corresponding derivatives of the lower bound to zero, which yields the fixed-point equations

$$S_i = \left( K_Y^{-1} - 2 \frac{\partial \tilde{F}(q, \theta)}{\partial S_i} \right)^{-1} \frac{\partial \tilde{F}(q, \theta)}{\partial \mu_i},$$

$$\mu_i = K_Y \frac{\partial \tilde{F}(q, \theta)}{\partial \mu_i}.$$  \hspace{1cm} (5.28)

This reveals that the $O(N^2)$ mean and covariance parameters are highly correlated via the prior kernel $K_Y$. As discussed by Damianou et al. [2011], this situation may be exploited to reparameterise the means and covariances according to the method described by Opper and Archambeau [2009] where the $N \times N$ diagonal matrix $\Lambda_i = -2 \frac{\partial \tilde{F}(q, \theta)}{\partial S_i}$ and the $N$-dimensional vector $\bar{\mu} = \frac{\partial \tilde{F}(q, \theta)}{\partial \mu_i}$ are used in place of $S_i$ and $\mu_i$. Thus, only the $2(Q \times N)$ variational parameters $(\lambda_i, \mu_i)$ need to be optimised, as the values of $S_i$ and $\mu_i$ are easily reconstructed using the fixed-point equations of Eqn. (5.28). In our work, optimisation of the lower bound is carried out using the iterative conjugate gradient optimiser developed by Møller [1993]. Due to the concavity of the log function, this is guaranteed to increase the likelihood of the model at each iteration. The gradient equations and further implementation details are discussed in appendix D. Following optimisation, the lower bound represents an approximation to the true marginal likelihood. Furthermore, it is easily shown that the integral in Eqn. (5.17) becomes strictly equal to the log of the marginal likelihood if and only if the variational distribution $q(F, U, X)$ is equal to the true posterior distribution $p(F, U, X|Z, Y)$. Therefore, after optimisation the variational distribution is an approximation to the true posterior distribution.
5.3 Classification

A rigorous approach to classification using Gaussian processes is described in [Rasmussen and Williams, 2006] and involves the use of non-Gaussian likelihoods to ensure the outputs may be interpreted as probabilities\(^1\). Rather than following this approach which requires the use of computationally complex approximation procedures, we apply a heuristic which shows similar performance but allows us to retain the Gaussian likelihood of the GPLVM models discussed in the previous sections. More specifically, the class labels are encoded using a 1-of-K coding scheme and \(Z\) is treated as a matrix of real values where each row contains elements from the set \([-1, 1]\). Thus, the model of Section 5.1 is used as a regression model for the data labels and allows the classification of a test data sample \(y_*\) to be carried out using two distinct methods. In the first method the output of the model is used directly:

\[
z_* = \arg\max_k p(f_*|y_*, Y, Z), \; k = 1, \ldots, K \tag{5.29}
\]

where \(p(f_*|y_*, Y, Z)\) is approximated by \(\int p(f_*|u_k, x_*) q(u_k) q(x_*) dx_*\), the expectation over \(q(u_k)\) is analytic and the remaining expectation is approximated by drawing samples from the Gaussian \(q(x_*)\). The latter quantity is given by

\[
q(x_*) = \langle p(x_*|y_*, X) \rangle_{q(X)} \tag{5.30}
\]

and is an approximation of the true posterior over the test point \(x_*\):

\[
p(x_*|y_*, Y, Z) = \int p(x_*|y_*, X) p(X|Y, Z) dX. \tag{5.31}
\]

Here, \(p(x_*|y_*, X)\) is obtained by conditioning the latent prior of Eqn. (5.2) after including the test pair \((x_*, y_*)\). The distribution \(p(X|Y, Z)\) represents the true posterior over the latent input vectors for which we have obtained a variational approximation \(q(X)\) by optimising the lower bound of Eqn. (5.17). Therefore, the true posterior over the test latent variable \(p(x_*|y_*, Y, Z)\) is approximated by a variational distribution \(q(x_*)\) over the test latent point and is given by the expectation of the conditional prior \(p(x_*|y_*, X)\) w.r.t \(q(X)\) as shown in Eqn. (5.30).

\(^1\)E.g. by employing sigmoid functions.
5.4 Complexity

Figure 5.2: The Cohn-Kanade dataset contains video sequences and landmark point annotations for the seven basic emotions anger (45), contempt (18), disgust (59), fear (25), happiness (69), sadness (28) and surprise (83) where the numbers indicate the number of sequences present. Here, the final frames of the video sequences are shown, corresponding to the ‘peak’ of the facial expression. Note the appearance of occluding video timings in the image on the far right.

The second method for prediction is simpler and is based only on the posterior over the test latent point \( x^*_s \). More precisely, we use the mean of \( q(x^*_s) \)

\[
m_i = k_{x^*_s} K^{-1}_Y \mu_i, \quad i = 1, \ldots, Q
\]

(5.32)
as an estimate of the latent point \( x^*_s \) associated with the input data \( y_s \) before performing k-nearest neighbour classification in the latent space.

5.5 Emotion Recognition with
Supervised Variational GPLVMs

To evaluate the models introduced in the previous sections, we performed several classification experiments using data obtained from the extended Cohn-Kanade emotions dataset [Kanade et al., 2000; Lucey et al., 2010].

5.5.1 Cohn-Kanade Dataset

The extended Cohn-Kanade dataset (CK+) [Kanade et al., 2000; Lucey et al., 2010] contains videos of actors performing the emotions anger, contempt, disgust, fear, hap-
ness, sadness and surprise. The first frame of the videos represents a neutral expression, while the final frame is the ‘peak’ of the emotion. All videos contain annotations obtained from active appearance model tracking of 68 landmark points. Figure 5.2 shows examples of the final frames of each type of emotion sequence contained in the dataset, where the number of sequences is given in brackets. As is clear from the figure, far fewer examples of the emotions ‘contempt’, ‘fear’ and ‘sadness’ are provided than for the stronger expressions of ‘anger’, ‘disgust’, ‘happiness’ and ‘surprise’. Incidentally, this provides a good test for our method, as we expect a generative Bayesian framework to outperform purely discriminative classifiers in the small dataset scenario. 10-fold cross-validation was used throughout in our experiments, resulting in a minimum class sample size of 16 in the case of ‘contempt’.

5.5.2 Features

To factor-out pose variation, we rigidly aligned the landmark points into a reference frame defined by the mean of the landmark annotations. The features that were used as input data to train the models were then given by the pose-normalised landmark points, i.e. the ‘shape’, as well as the pixels sampled using the triangulated point mesh, i.e. the ‘texture’. Also, in order to minimise the influence of lighting variation which we did not wish to model here, adaptive histogram equalisation was applied to each texture image [Zuiderveld, 1994]. All experiments used only the final frame of each video sequence. Since there were 68 landmark points, the shape feature was 136D and in our experiments the reference frame was scaled to allow the sampling of 3000 pixels, which resulted in a 3000D texture feature. In all experiments, the shape and texture features were normalised to zero mean, unit variance.

The features used in our work are similar to those used by Lucey et al. [2010]. However, we do not normalise for identity by subtracting the (neutral) first frame of each video from the final frame. Normalising for identity results in an optimistic estimate of the generalisation error, as neither the identity nor the individual’s neutral expression are known in a real-world testing scenario.

5.5.3 Models

Using the feature extraction procedure detailed above, the appearance of the face is decoupled into independent shape and texture components. Emotions resulting in obvious deformations of the shape such as happiness or surprise, which involve opening of the mouth, should be easily distinguished using the shape component alone. On
the other hand, emotions such as *contempt* or *anger*, which result in the appearance of wrinkles, are more likely to be classified correctly based on the texture feature. In our experiments, we trained independent models of the type shown in Figure 5.1b on the shape and texture features, resulting in a set of latent variables $X_s$ and $X_t$ corresponding to the training data samples $Y_s$ and $Y_t$. In general, however, it is the combination of shape and texture, i.e. the appearance which forms a large part of the information that humans use to judge the emotional state of individuals. Therefore we also trained a joint model based on the concatenated latent spaces $Y_c = [X_s, X_t]$ and the autoencoder model shown in Figure 5.1c. After training, the test data was classified using the methods described in Section 5.3.

**Initialisation** The critical parameters in the Bayesian GPLVM models are the variance $\sigma$ of the RBF-ARD kernel 5.7, used in Eqn. (4.4) and the model noise variance $\epsilon = \frac{1}{\beta}$, which together specify the signal-to-noise ratio of the model. A high signal-to-noise ratio indicates that the model ‘fits’ the data and since we wish to capture the structure of the input data, it is imperative that a high signal-to-noise ratio results after training. Since the latent variables $X$ are integrated out in this model and type-II maximum likelihood is used to estimate the remaining parameters, the risk of overfitting the data is much lower than in the ML or MAP GPLVM framework. Bearing in mind that the classification heuristic described in Section 5.3 depends on the model fitting the label data, the SNR ratio is fixed to a high value and the parameters $\sigma$ and $\beta$ are not optimised in the supervised model. The means of the variational distribution $q(X)$ are initialised using PCA, where 20 dimensions are retained in our experiments. Furthermore, care must be taken that the RBF-ARD parameters $\alpha_q$ as well as the length scale of the kernel used in Eqn. (5.2) are initialised based on the ‘radius’ of the input data to ensure that the kernel matrices encode the structure of the data and are not diagonal, since a diagonal kernel matrix corresponds to an uninformative “white noise” process.
Table 5.1: Classification using the shape and texture models described in Section 5.5.3 in a 10-fold cross-validation experiment. The numbers represent the percentage of success for each class. (a) Results using the independent shape and texture models. (b) Results using the combination of shape and texture latent spaces. ‘AutoEnc’ refers to the 1-NN classification in the latent space obtained after training an autoencoder model (cf. Figure 5.1c).

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Shape 1-NN</th>
<th>Posterior</th>
<th>Texture 1-NN</th>
<th>Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIN RBF M32</td>
<td></td>
<td>LIN RBF M32</td>
<td></td>
</tr>
<tr>
<td>An</td>
<td>77.8</td>
<td>80.0</td>
<td>82.2</td>
<td>82.2</td>
</tr>
<tr>
<td>Co</td>
<td>66.7</td>
<td>55.6</td>
<td>55.6</td>
<td>72.2</td>
</tr>
<tr>
<td>Di</td>
<td>91.5</td>
<td>93.2</td>
<td>93.2</td>
<td>94.9</td>
</tr>
<tr>
<td>Fe</td>
<td>84.0</td>
<td>76.0</td>
<td>80.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Ha</td>
<td>97.1</td>
<td>97.1</td>
<td>98.6</td>
<td>97.1</td>
</tr>
<tr>
<td>Sa</td>
<td>75.0</td>
<td>78.6</td>
<td>82.1</td>
<td>75.0</td>
</tr>
<tr>
<td>Su</td>
<td>95.2</td>
<td>97.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>Overall</td>
<td>83.9</td>
<td>82.6</td>
<td>83.8</td>
<td>84.4</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Combined (AutoEnc)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIN RBF M32</td>
</tr>
<tr>
<td>An</td>
<td>93.3</td>
</tr>
<tr>
<td>Co</td>
<td>77.8</td>
</tr>
<tr>
<td>Di</td>
<td>91.5</td>
</tr>
<tr>
<td>Fe</td>
<td>84.0</td>
</tr>
<tr>
<td>Ha</td>
<td>98.6</td>
</tr>
<tr>
<td>Sa</td>
<td>75.0</td>
</tr>
<tr>
<td>Su</td>
<td>97.6</td>
</tr>
<tr>
<td>Overall</td>
<td>88.3</td>
</tr>
</tbody>
</table>

(b)

Table 5.2: Comparison of related methods for emotion recognition with the shape and texture features described in Section 5.5.2. As noted in the text, the SVM results published in Lucey et al. [2010] use identity normalised data, and are therefore an optimistic estimate of the generalisation performance.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Shape DGPLVM SVM RF LDA</th>
<th>Texture DGPLVM SVM RF LDA</th>
<th>Combined LDA SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>An</td>
<td>33.3</td>
<td>35.0</td>
<td>86.0</td>
</tr>
<tr>
<td>Co</td>
<td>50.0</td>
<td>25.0</td>
<td>55.6</td>
</tr>
<tr>
<td>Di</td>
<td>69.5</td>
<td>68.4</td>
<td>90.7</td>
</tr>
<tr>
<td>Fe</td>
<td>48.0</td>
<td>21.7</td>
<td>67.2</td>
</tr>
<tr>
<td>Ha</td>
<td>81.2</td>
<td>98.4</td>
<td><strong>100.0</strong></td>
</tr>
<tr>
<td>Sa</td>
<td>21.49</td>
<td>4.0</td>
<td>66.4</td>
</tr>
<tr>
<td>Su</td>
<td>91.68</td>
<td>50.0</td>
<td><strong>100.0</strong></td>
</tr>
<tr>
<td>Overall</td>
<td>56.5</td>
<td>50.35</td>
<td>80.5</td>
</tr>
</tbody>
</table>
5.5 Emotion Recognition with Supervised Variational GPLVMs

5.5.4 Results

Table 5.1a shows the classification results obtained using the supervised Bayesian GPLVM model of Section 5.1 with the shape and texture features of Section 5.5.2. \( \text{LIN}, \ RBF \) and \( M32 \) (Matérn32) refer to the different kernels used for the latent space prior of Eqn. (5.2). An immediate observation is that while the linear kernel provides similar performance to the non-linear kernels in the case of the shape, in general this is not the case for the texture where the linear kernel performs significantly worse. The texture data may thus be seen to contain more significant non-linearities. Furthermore, as may be expected, the differences between the “Posterior” and “1-NN” methods are small. Figure 5.3b shows the latent space and the inverse lengthscale hyperparameters of the RBF-ARD kernel of an experiment using the supervised Bayesian GPLVM together with the texture data. The latent space shows good separation between the different classes and Figure 5.3a shows that two dominant dimensions remain after optimisation. The results obtained for the combined models are given in table 5.1b. It is clear that combining shape and texture information leads to a significant improvement in performance. The quality of the models is evident in Figure 5.4 which shows the covariance matrices of the GP prior distribution of a shape, texture, and combined model. The covariance matrices of the shape- and texture-based models are computed on the raw input data, and although the correlation within classes is visible, between-class correlations are high. However, in the case of the combined model, whose inputs are the concatenated latent spaces of the trained shape- and texture-based models, the strikingly block-diagonal structure of the prior covariance matrix reflects the separation of the seven emotions and is a clear indication of the effectiveness of the supervised dimensionality reduction model presented in Section 5.1.

5.5.5 Comparison

In order to validate our models, we compared our method to other commonly used supervised Machine Learning techniques such as linear discriminant analysis (LDA), random forests [Breiman, 2001] and discriminative GPLVMs [Urtasun and Darrell, 2007]. Furthermore, we compared to the SVM results published in [Lucey et al., 2010]. No further feature extraction was performed, and all algorithms were trained on the shape and texture features described in Section 5.5.2. The results are given in table 5.2. Since the random forest is a stochastic algorithm, we performed 10 independent 10-fold cross-validation experiments of which we report the mean. In order to apply LDA to the training data, the dimensionality of the input data must first be reduced to ensure
Figure 5.3: (a) Out of the 20 dimensions used to initialise the model, three relevant dimensions remain after training of a supervised Bayesian GPLVM texture model (cf. 5.1), as is evidenced by the values of the inverse length scales of the RBF-ARD kernel which is used in the latent function mapping $p(F|X)$. (b) Optimal 2D projection of the latent variables $X$ corresponding to the training samples contained in the 3D latent space.

Figure 5.4: RBF covariance matrices $K_Y$ of the GP priors $p(X|Y)$ over the latent space of the shape, texture and combined models after training. (a) and (b) take the raw shape and texture features as inputs, whereas (c) shows the covariance matrix of the latent space prior of the combined autoencoder model (cf. 5.5.3), which takes the concatenated latent spaces of the supervised shape and texture models as input.
non-singularity of the covariance matrices used by the LDA algorithm [Martinez and Kak, 2001]. This was done by employing PCA and retaining 20 dimensions, as in our experiments using the supervised Bayesian GPLVM. Following training of the 6D LDA space, 1-NN classification was used to classify the test samples. Furthermore, to compare to the combined model experiments described in Section 5.5.3, the results of which are shown in table 5.1b, a combined shape and texture classifier based on LDA was built in an analogous fashion by training a 6D combined LDA latent space using the 6D LDA shape and texture model latent spaces as inputs. Overall, the LDA classifier outperforms all the other methods in table 5.2 by a considerable margin. Comparing to the supervised GPLVM models in table 5.1b, the LDA model produces comparable results on the shape data, while the non-linear models perform significantly better on the texture and in the combined case. Overall, the best results are obtained using the autoencoder model trained on the combined shape and texture latent spaces using an RBF prior and represent a more than 10 percent improvement over the SVM results published in Lucey et al. [2010].

5.6 Implementation

The various GPLVM models used in this chapter were implemented from scratch in C++ after having worked with Matlab as well as C++ code which is freely available on Neil Lawrence’s website and which was co-written by Andreas Damianou and Michalis Titsias. The LDA algorithm used in the experiments was also implemented from scratch in C++, while the freely available R implementation was used for the random forest classification experiments. Furthermore, the discriminative GPLVM experiments were carried out using the implementation contained in Neil Lawrence’s freely available Matlab code.

5.7 Conclusions

In this chapter we introduced a variational Bayesian GPLVM framework which allows supervised training of GPLVMs. As a fundamental part of this model, we showed how the idea of “back-constraints” may be cast in the probabilistic framework of the GPLVM. Furthermore, we showed how an autoencoder GPLVM model may be trained which reproduces the input data via the “bottleneck” of a low-dimensional latent space. We evaluated these models on the extended Cohn-Kanade emotions dataset and presented competitive results using the supervised GPLVM, which were improved upon by building hierarchical models using the autoencoder GPLVM. However, the supervised GPLVM
we developed in this chapter is no longer a generative model of the input data, and it is therefore not possible to generate texture samples given a position in latent space. This is unfortunate, since supervised training results in semantically clustered latent spaces which could be exploited for generating faces under sparse user constraints. For instance, if trained on video sequences of faces, meaningful sequences could be generated by constraining the trajectory to pass through certain states in the latent space. Further work should therefore focus on including supervision into generative GPLVMs without removing the generative nature of the model, a starting point for which could be recently published work by Damianou et al. [2012]. As part of this work, the idea of autoencoder models could be investigated further, building on the concept of “deep belief networks”, popularised by Hinton and Salakhutdinov [2006]. Furthermore, autoencoder GPLVMs could be used to build a non-linear active appearance model as was done using the MAP-GPLVM framework in the previous chapter. However, it seems unlikely that the use of such models together with the optimisation method employed in the last chapter would lead to a significant improvement, and, unfortunately, the computation of the gradients is significantly more complex as is easily seen by comparing the objective functions in Eqn. (5.20) and Eqn. (4.11).
Chapter 6

Conclusions and Future Work

In this thesis we presented two groups of statistical methods for representing and analysing faces in digital images, and our work may be divided into three main contributions. First, we discussed discriminative fitting algorithms for parametric shape models which directly connect evidence gathered from features extracted at the current model location to parameter updates. This type of model, referred to as a “discriminative” AAM, has been shown previously to be efficient and robust, and contrasts with the generative approach to shape model fitting in which parameter updates are calculated indirectly via the optimisation of an objective function which includes a statistical texture model of the pixel data. In an attempt to improve the accuracy of discriminative AAMs, we investigated two types of features and three different regression models based on boosting, random forests and Gaussian processes. Furthermore, we devised two sequential training algorithms for discriminative AAMs in an attempt to combine the efficiency and robustness of the discriminative approach with the accuracy provided by the minimisation of an objective function. The results of the experiments carried out showed that Gaussian process regression was the least efficient but most accurate fitting method and that the proposed sequence training improved accuracy and robustness when compared to simple application of discriminative AAMs.

In the second contribution of this thesis we investigated replacing the linear texture models in inverse-compositional generative AAMs with non-linear Gaussian process latent variable models. As expected, our experiments showed that the non-linearity of the texture model allows more detailed textures to be encoded using fewer latent dimensions, and that, when properly fitted, the resulting textures appear more natural when compared to linear models of the same dimensionality.
The final contribution focused on developing a supervised training algorithm for GPLVMs which we applied to the problem of classifying facial expressions. This work employed a more recent Bayesian formulation of the GPLVM which also allowed us to formalise the notion of “back-constraints”, an ad hoc implementation of which was used for the non-linear AAM. The resulting supervised GPLVM and autoencoder GPLVM models were put to the test in a series of facial expression recognition experiments in which they outperformed several commonly used and related methods.

6.1 Discriminative AAMs

Although the sequences of discriminative AAM algorithms had a much larger convergence radius, they were still less accurate compared to generative AAM algorithms when the model location was close to the ground truth. This suggests a hybrid approach to model fitting and thus future work could investigate using an efficient discriminative AAM sequence to obtain a first estimate of the ground truth, after which the estimate could be refined using a generative AAM. Furthermore, the fact that the highly tuned Haar features did not perform much better when compared to the more ad hoc steerable pyramid features raises the question whether they provide a sufficiently rich representation of the data. Future work could therefore also focus on improving the steerable pyramid features by introducing optimisation of the shape and location of the patches used for calculating the sample statistics, as well as on investigating the use of successful keypoint detection methods such as SIFT [Lowe, 1999] and SURF [Bay et al., 2008].

6.2 Non-Linear AAM

With non-linear models one faces the difficulty of optimising a non-linear objective function both when training and when testing the model. As was demonstrated by some of the experiments using low-dimensional latent spaces, this has the potential to eliminate any representational advantage over linear models, especially in the case that the initial model location is not close to the ground truth. This is a fundamental problem when dealing with non-linear models and cannot be easily overcome. However, future work could exploit the probabilistic nature of the GPLVM by investigating the use of more informative prior distributions that model temporal coherence and provide a more semantic clustering of the latent space. This could improve robustness when dealing with frames from video sequences or when the identity and appearance of an individual is known in advance, provided that a good latent space initialisation is given.
6.3 Supervised GPLVM

Although it performed well in the experiments, a downside of the supervised GPLVM model devised in this thesis is that it may no longer be considered a generative model of the input data, since it models the labels associated with the data and does not provide a means to generate face textures. Future work should therefore focus on adding supervision to the training of the GPLVM without compromising its generative nature, and could build on a similar effort by Damianou et al. [2012]. In this context, the use of autoencoder models could be investigated more closely by building on recent advances in the field of neural networks, where Hinton and Salakhutdinov [2006] showed that the use of autoencoder models to determine initial estimates of the model parameters in a hierarchical pre-training step before employing supervised training leads to improved accuracy. This was further discussed by Erhan et al. [2010], and an application of this idea to the GPLVM models presented in this chapter seems an interesting topic for further work.
Appendix A

Similarity Transforms

In general, a set of reference points

\[ P_{ref} = (\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_n, \tilde{y}_n), \]  
(A.1)

and a point set

\[ P = (x_1, y_1, \ldots, x_n, y_n) \]  
(A.2)

may be brought into alignment by applying a transformation to the reference points which contains rigid as well as non-rigid components. The rigid component

\[ T_q : P_{ref} \rightarrow \hat{P}, \]  
(A.3)

where the point set \( \hat{P} \) lies within a non-rigid transformation of \( P \), is a similarity transform whose parameters

\[ q = (a, b, t_x, t_y) \]  
(A.4)

are determined by solving the least-squares optimisation problem

\[ q_* = \arg \min_q \sum_i \| r_i - (M \tilde{r}_i + t) \|^2 \]

\[ = \arg \min_q \sum_i \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \left[ \begin{pmatrix} 1 + a & -b \\ b & 1 + a \end{pmatrix} \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \right] \right\|, \]  
(A.5)

where \( M \) is a scale-rotation matrix and \( t \) is a vector of translation parameters.
A.1 Calculating transform parameters

The pose of the landmark point set $P$ is defined by the parameters $q$ of the transformation $T_q$. In this parameterisation, the parameters $(t_x, t_y)$ describe translation and the rotation angle $\theta$ and scale $\alpha$ are combined in the parameters

$$a = \alpha \cos \theta - 1$$
$$b = \alpha \sin \theta.$$

(A.6)

This is the parameterisation used by Matthews and Baker [2004], one of the main reasons being that $q = 0$ represents the identity transform. Furthermore, the use of the parameters $a$ and $b$ defined in Eqn. (A.6) hides the non-linear dependence on the rotation angle and facilitates the linear least-squares fitting method for determining the transform parameters which is described in the following.

A.1 Calculating transform parameters

The parameters $q$ of the similarity transform $T_q$ are found by calculating the parameters $q_*$ with respect to which the error after alignment becomes minimal. A general solution to the problem of rigid alignment of two $m$-dimensional point sets is derived by Umeyama [1991]. However, in the simple case of a 2d similarity transform, the solution may be obtained directly from the linear least-squares objective function $F$ defined in Eqn. (A.5).

A.1.1 Translation

Without loss of generality the centroid of the reference point set $P_{ref}$ may be chosen to lie at the origin so that it is not affected by rotation and scaling of $P_{ref}$. Thus, omitting $M$ and setting the derivative w.r.t. the translation parameters $t = (t_x, t_y)$

$$\frac{\partial F}{\partial t} = -\sum_i [r_i - (\tilde{r}_i + t)]$$

(A.7)

$$= -Nt + \sum_i (r_i - \tilde{r}_i)$$

(A.8)

to zero, we have

$$t = \frac{1}{N} \sum_i r_i,$$

(A.9)

which is just the centroid of $P$. 

103
A.1.2 Scale and Rotation

Once the translational degree of freedom has been removed from both point sets by moving the centroids to the origin, the scale and rotation parameters may be determined from the optimisation problem

\[
\begin{align*}
\begin{pmatrix} \tilde{a} \\ b \end{pmatrix} &= \arg \min_{(\tilde{a},b)} \sum_i \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} \tilde{a} \\ b \end{pmatrix} \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} \right\|^2,
\end{align*}
\]

where we have set \( \tilde{a} = 1 + a \). Setting the derivatives w.r.t. \( \tilde{a} \) and \( b \) to zero and solving the resulting linear systems of equations, we find

\[
a = \frac{\sum_i \tilde{x}_i x_i + \sum_i \tilde{y}_i y_i}{\sum_i \tilde{x}_i \tilde{x}_i + \sum_i \tilde{y}_i \tilde{y}_i} - 1
\]

and

\[
b = \frac{\sum_i \tilde{x}_i \tilde{y}_i - \sum_i x_i \tilde{y}_i}{\sum_i \tilde{x}_i \tilde{x}_i + \sum_i \tilde{y}_i \tilde{y}_i}.
\]

Using the definitions given in Eqn. (A.6), the scale

\[
\alpha = (1 + a)^2 + b^2
\]

and orientation

\[
\theta = \arctan \frac{b}{1 + a}
\]

are easily recovered.

A.2 Composition

Given two similarity transforms \( T_q \) and \( T_q' \), the parameters

\[
q'' = (a'', b'', t_x'', t_y'')
\]

of the composition

\[
T_{q''} = T_q \circ T_q'
\]
A.3 Inverse

are easily computed using the definition of the scale-rotation matrix $M$ and translation vector $t$ in Eqn. (A.5), yielding

\[
a'' = (a + 1)(a' + 1) - bb' - 1 \\
b'' = (a + 1)b' + b(a' + 1) \\
t_x'' = (a + 1)t'_x - bt'_y + t_x \\
t_y'' = bt'_x + (a + 1)t'_y + t_y.
\]  

(A.17) 

(A.18) 

(A.19) 

(A.20)

A.3 Inverse

Given the similarity transform $T_q$ with parameters

\[ q = (a, b, t_x, t_y) \]  

(A.21)

that describes the rigid alignment of the reference point set $P_{ref}$ to the point set $P$, we require the parameters $q'$ of the inverse transform $T_q^{-1}$ which rigidly aligns $P$ to $P_{ref}$. The identity transform $id$ may be expressed as

\[ T_q \circ T_q^{-1} = id, \]  

(A.22)

and therefore the parameters $q' = (a', b', t'_x, t'_y)$ of the inverse transform $T_q^{-1}$ are determined by setting eqs. A.17 - A.20 to zero:

\[
a' = \frac{a + 1}{(a + 1)^2 + b^2} - 1 \]  

(A.23)

\[
b' = -\frac{b}{(a + 1)^2 + b^2} \]  

(A.24)

\[
t'_x = -a't_x + b't_y \]  

(A.25)

\[
t'_y = -b't_x - a't_y. \]  

(A.26)
Appendix B

Matrix Identities and Linear Algebra

In the following, we give a brief summary of some of the matrix identities and methods used in this thesis. A complete treatment including the definition of all the mathematical terms used is beyond the scope of this appendix. However, a good starting point for further reading is [Strang, 1988].

B.1 Matrix product

For $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times p}$, the matrix product is defined as follows:

$$(AB)_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

(B.1)

B.2 Hadamard product

Given two equally sized matrices $A, B \in \mathbb{R}^{n \times m}$, the Hadamard product, denoted by the symbol $\circ$, is defined as the element-wise product:

$$(A \circ B)_{ik} = a_{ik} b_{ik}$$

(B.2)
B.3 Trace

Given a square matrix $A \in \mathbb{R}^{n \times n}$, the trace is defined as the sum of the diagonal elements:

$$Tr(A) = Tr(A^T) = \sum_{i=1}^{n} a_{ii}. \quad (B.3)$$

Cyclic invariance

The trace is invariant to cyclic permutations of its arguments. If $A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times p}$ and $C \in \mathbb{R}^{p \times n}$,

$$Tr(ABC) = Tr(BCA) = Tr(CAB). \quad (B.4)$$

Factorised square matrix

Let $A, B \in \mathbb{R}^{n \times d}$. The trace of the matrix product $C = AB^T$ is given by the sum of the elements of the Hadamard product:

$$Tr(AB^T) = \sum_{i=1}^{n} \sum_{k=1}^{d} a_{ik}b_{kj}^T \delta_{ij} \quad (B.5)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{d} a_{ik}b_{ki}^T \quad (B.6)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{d} a_{ik}b_{ik} \quad (B.7)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{d} (A \circ B)_{ik} \quad (B.8)$$

B.4 Cholesky Factorisation

A symmetric, positive definite matrix $A \in \mathbb{R}^{n \times n}$ may be factorised into a product of a positive definite lower triangular matrix $L$ and its upper triangular transpose $L^T$:

$$A = LL^T \quad (B.9)$$

Cholesky factorisation is the preferred way of solving a linear system of equations with symmetric positive definite matrix $A$:

$$Ax = b. \quad (B.10)$$
Since the matrix $L$ is lower triangular, the system

$$LL^T x = b$$  \hspace{1cm} (B.11) $$

is easily solved in two steps by first solving

$$Ly = b$$  \hspace{1cm} (B.12) $$

using forward substitution to determine $y$, and then solving the system

$$L^Tx = y$$  \hspace{1cm} (B.13) $$

using back-substitution to determine $x$. This approach to solving Eqn. B.10 does not require the direct computation of a matrix inverse, and is therefore much more numerically stable. The only source of numerical instability is the presence of round-off errors, which can cause elements in the original matrix to become negative in the case that $A$ is badly conditioned. This may be avoided by adding a small correction referred to as “jitter” to the diagonal of $A$.

In this thesis it is often necessary to determine the logarithm of the determinant of a symmetric positive definite matrix $A$. This is easily computed using the Cholesky factorisation and the properties of the determinant:

$$\log |A| = \log |L_A L_A^T|$$  \hspace{1cm} (B.14) $$

$$= \log |L_A| + \log |L_A^T|$$  \hspace{1cm} (B.15) $$

$$= 2 \sum_i (L_A)_{ii}$$  \hspace{1cm} (B.16) $$

### B.5 Spectral Decomposition

Given a matrix $A \in \mathbb{R}^{n \times n}$ of full rank $n$ (i.e. whose determinant is non-zero), then its $n$ eigenvectors $v_j$ and corresponding eigenvalues $\lambda_j$ satisfy

$$Av_j = \lambda_j v_j, \quad j = 1, \ldots, n.$$  \hspace{1cm} (B.17) $$

Concatenating the eigenvectors into the columns of a matrix $P$, the matrix $A$ may be written as the product of three factors

$$A = PDP^{-1}.$$  \hspace{1cm} (B.18) $$

where $D$ is a diagonal matrix containing the $n$ eigenvalues $\lambda_i$ on its diagonal.
Symmetric Matrices

If $A$ is symmetric, the eigenvectors are orthogonal, and the factorisation becomes

$$A = PD P^T$$  \hspace{1cm} (B.19)

$$= \sum_{i=1}^{n} \lambda_i v_i v_i^T.$$  \hspace{1cm} (B.20)

Covariance Matrices

Assuming the matrix $Y \in \mathbb{R}^{N \times D}$ contains $N$ $D$-dimensional data vectors in its rows, then the symmetric $D \times D$ sample covariance matrix is given by

$$S = \frac{1}{N} Y^T Y.$$  \hspace{1cm} (B.21)

If $N < D$, this matrix is singular, and the eigenvectors corresponding to non-zero eigenvalues may be more efficiently calculated using the $N \times N$ matrix

$$\tilde{S} = \frac{1}{N} YY^T.$$  \hspace{1cm} (B.22)

In fact, let $v$ be an eigenvector of $\tilde{S}$ with eigenvalue $\tilde{\lambda}$, so that

$$\tilde{S} v = \tilde{\lambda} v.$$  \hspace{1cm} (B.23)

Multiplying from the left with $Y^T$, we find

$$SY^T v = \tilde{\lambda} Y^T v,$$  \hspace{1cm} (B.24)

from which we see that $w = Y^T v$ is an eigenvector of $S$ with eigenvalue $\tilde{\lambda}$.

Numerical Stability and Condition Numbers

The finite precision of floating point arithmetic on a computer means that in general floating point numbers are only determined up to a fixed error $\epsilon_m$ of their true value. Thus, when solving a problem such as a linear system of equations (cf. Eqn. B.10), we are actually analysing a perturbed problem

$$Ax = b + \epsilon,$$  \hspace{1cm} (B.25)
the solution of which,
\[ x + \delta = A^{-1}b, \] (B.26)
deviates from the true solution \( x \) by an amount \( \delta \). The magnitude of this deviation is determined by a numerical property of \( A \) known as the condition number \( \kappa \), which for any non-singular square matrix is given by the ratio of the largest to the smallest eigenvalue:
\[ \kappa(A) = \left| \frac{\lambda_{\max}}{\lambda_{\min}} \right| \] (B.27)
Using a suitable norm, it may be shown that the magnitude of the relative errors are related by the following inequality:
\[ \frac{||\delta||}{||x||} \leq \kappa(A) \frac{||e||}{||b||}. \] (B.28)
From this it follows that a condition number that is close to unity is desirable, since it specifies by how much the initial error is amplified when passed through the linear system of Eqn. B.10.

B.6 Partitioned Inverse

The inverse of a partitioned matrix
\[ \Sigma = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \] (B.29)
is given by the identity [Bernstein, 2009; Bishop, 2006]
\[ \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M^{-1} & -M^{-1}BD^{-1} \\ -D^{-1}CM^{-1} & D^{-1} + D^{-1}CM^{-1}BD^{-1} \end{pmatrix}, \] (B.30)
where
\[ M = (A - BD^{-1}C) \] (B.31)
is the Schur complement with respect to \( D \). Apart from when working with multivariate Gaussian distributions where partitioned inverses often appear, the fact that only \( D \) and \( M \) require inversion for computing the partitioned inverse makes this identity especially useful when \( D \) and \( M \) are small or diagonal matrices whose inversion is simple.
B.7 Matrix Inversion Lemma

The derivation of the partitioned inverse equation leads to another result known as the matrix inversion lemma, which relates the inverse of a rank-$k$ correction to a matrix $A$ to a rank-$k$ correction of the inverse of $A$:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}. \quad (B.32)$$

This identity is useful when the inverse of $A$ is known and $C$ is small, in which case the computation of the inverse of the matrix on the left-hand side only requires the inversion of the smaller matrix $C^{-1} + DA^{-1}B$.

B.8 Matrix Determinant Lemma

The matrix determinant lemma allows the efficient computation of the determinant of a rank-$k$ correction to a matrix $A$ in the case that the inverse and determinant of $A$ are known:

$$|A + BCD^T| = |C^{-1} + D^T A^{-1} B| |C| |A| \quad (B.33)$$

B.9 Matrix Derivatives

The computation of matrix derivatives is often required, especially when determining analytic gradient expressions for objective functions which contain matrices that depend on parameters. In the following, we give some of the most common results, assuming $K$ is a matrix which depends on a parameter $\theta$. Many more results may be found in [Petersen and Pedersen, 2012].

**Inverse**

$$\frac{\partial K^{-1}}{\partial \theta} = -K^{-1} \frac{\partial K}{\partial \theta} K^{-1} \quad (B.34)$$

**Determinant**

$$\frac{\partial |K|}{\partial \theta} = |K| \text{Tr} \left( K^{-1} \frac{\partial K}{\partial \theta} \right) \quad (B.35)$$

$$\frac{\partial \log |K|}{\partial \theta} = \text{Tr} \left( K^{-1} \frac{\partial K}{\partial \theta} \right) \quad (B.36)$$
Appendix C

Gaussian Identities

This appendix contains results for multivariate Gaussian distributions which may be found in standard Statistics and Machine Learning textbooks such as [Mardia et al., 1979, chap. 3] and [Bishop, 2006, chap. 2].

C.1 Multivariate Gaussian Distribution

A multivariate random variable $\mathbf{x}$ is said to be Gaussian if its associated probability distribution $p(\mathbf{x})$ is a member of the family of multivariate Gaussian distributions

$$
N(\mathbf{x}|\mu, \Sigma) = \frac{1}{\sqrt{2\pi} |\Sigma|} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right),
$$

which is parameterised by a mean vector

$$
\mu = \mathbb{E}[\mathbf{x}] = \int \mathbf{x} N(\mathbf{x}|\mu, \Sigma) d\mathbf{x}
$$

and a symmetric covariance matrix

$$
\Sigma = \text{Cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T].
$$

The family of Gaussian distributions is closed under the sum and product rules of probability as well as under linear transformations of the variable. The closure properties and the fact that the Gaussian family is completely determined by the mean and covariance parameters may be used to simplify Gaussian computations. Given an expression involving Gaussian distributions, the closure properties may be invoked to
C.2 Marginal and conditional distributions

determine if the result is also Gaussian. If this is the case, noting that the exponent of a Gaussian distribution is a quadratic form,

\[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) = -\frac{1}{2} x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu + \text{const}, \quad (C.4)\]

the mean and covariance of the result are found by collecting quadratic and linear terms in \(x\) and using the method of “completing the square” [Bishop, 2006, chap. 2].

C.2 Marginal and conditional distributions

Given a joint Gaussian distribution with partitioned mean and covariance

\[p(x, y) = N\left(\begin{bmatrix} x \\ y \end{bmatrix} \bigg| \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right), \quad (C.5)\]

then the marginal distribution over \(x\) is

\[p(x) = \int p(x, y) dy = N(x|\mu_x, A). \quad (C.6)\]

This is commonly referred to as the marginalisation property of multivariate Gaussian distributions. If \(x\) is observed, the conditional distribution of the random variable \(y\) is given by

\[p(y|x) = N\left(y \bigg| \mu_y + CB^{-1}(x - \mu_x), A - CB^{-1}C^T \right). \quad (C.7)\]

C.3 Linear Gaussian Models

Linear Gaussian models are common in Machine Learning and refer to models where both the likelihood as well as the prior distribution are Gaussian [Roweis and Ghahramani, 1999]:

\[p(y|x) = N(y|Ax + b, \Sigma_y) \quad (C.8)\]
\[p(x) = N(x|\mu_x, \Sigma_x). \quad (C.9)\]

Invoking the closure properties of the Gaussian family, the joint distribution

\[p(y, x) = p(y|x)p(x) \quad (C.10)\]
is easily seen to be Gaussian, and the posterior and marginal distributions are obtained in closed form (c.f. [Bishop, 2006, chap. 2]):

\[
p(y) = N(y|A\mu + b, \Sigma_y + A\Sigma_x A^T)
\]
\[
p(x|y) = N(x|\Sigma (A^T\Sigma_y^{-1}(y - b) + \Sigma_x^{-1}x), \Sigma)
\]

where
\[
\Sigma = (\Sigma_x^{-1} + A^T\Sigma_y^{-1}A)^{-1}.
\]

### C.4 Linear Transformations

As noted in the introduction, the family of Gaussian distributions is closed under linear transformations of the variables. More specifically, if

\[
y = Ax + b
\]

and

\[
p(x) = N(x|\mu, \Sigma),
\]

then

\[
p(y) = N(y|A\mu + b, A\Sigma A^T).
\]

This follows directly from the linearity of the transformation and the definition of the mean and covariance in eqs. C.2 and C.3.

### C.5 Random Sampling

In practice, it is often necessary to generate random samples from a multivariate Gaussian distribution given a particular mean and covariance. A uniform random number generator may be used to draw independent univariate normal samples [Box and Muller, 1958], which may be combined to form a sample \(\tilde{x}\) from the multivariate normal distribution

\[
p(x) = N(x|0, I).
\]

Setting \(\mu = 0\) and \(\Sigma = I\) in Section C.4, it is easily shown that a sample \(\tilde{y}\) from the multivariate Gaussian distribution

\[
p(y) = N(y|m, S)
\]
is obtained by applying the linear transformation

\[ \tilde{y} = L\tilde{x} + m, \] (C.19)

where \( S = LL^T \) is the Cholesky factorisation of \( S \).

### C.6 Gaussian Process Regression

The simplest derivation of the marginal and predictive distributions given in Eqns. 3.20 to 3.22 starts from the joint distribution

\[
p(f_*, f, y) = p(y|f)p(f, f_*)
\]

\[
= N\left( \begin{bmatrix} f_* & 0 \\ f & 0 \\ y & 0 \end{bmatrix}, \begin{bmatrix} K_{**} & K_{*N} & K_{NN} \\ K_{N*} & K_{NN} & K_{N*} \\ K_{NN} & K_{N*} & K_{NN} + \sigma^2 I \end{bmatrix} \right). \] (C.20)

Applying the marginalisation property of Eqn. C.2, the marginal likelihood of Eqn. 3.20 immediately follows. Conditioning on \( y \) using Eqn. C.7, we recover the joint posterior

\[
p(f_*, f | y) = N\left( \begin{bmatrix} f_* \\ f \\ y \end{bmatrix}, \begin{bmatrix} m_* \\ m_N \end{bmatrix}, \begin{bmatrix} C_* & C_{*N} \\ C_{N*} & C_N \end{bmatrix} \right), \] (C.21)

where

\[
m_* = K_{*N}(K_{NN} + \sigma^2 I)^{-1}y \] (C.22)

\[
m_N = K_{NN}(K_{NN} + \sigma^2 I)^{-1}y \] (C.23)

and

\[
C_* = K_{**} - K_{*N}(K_{NN} + \sigma^2 I)^{-1}K_{N*} \] (C.24)

\[
C_{*N} = K_{*N} - K_{*N}(K_{N*} + \sigma^2 I)^{-1}K_{NN} \] (C.25)

\[
C_{N*} = C_{*N}^T \] (C.26)

\[
C_N = K_{NN} - K_{NN}(K_{NN} + \sigma^2 I)^{-1}K_{NN}. \] (C.27)

Marginalising \( f \) using Eqn. C.7, the predictive distribution of Eqn. 3.21 is recovered:

\[
p(f_* | y) = N(f_* | m_*, C_*). \] (C.28)
Appendix D

Implementation

In this appendix, we give details for the implementation of some of the models presented in this thesis. The expressions obtained for the variational Bayesian models are complex, and giving all steps of the derivations is beyond the scope of this thesis. I would like to thank Michalis Titsias and Andreas Damianou for sharing (unpublished) results and derivations relating to the work in [Titsias, 2009] and [Damianou et al., 2011].

D.1 Variational Sparse GP Regression

The variational GP model by Titsias [2009] which is discussed in chapter 3.2.4 is fitted to the data by optimising the following objective function with respect to the inducing inputs $X_u$, the kernel hyperparameters $\theta_f$ and the inverse noise variance $\beta = \frac{1}{\sigma^2}$:

$$F_V(X_u, \theta_f, \beta) = - \frac{N}{2} \log 2\pi + \frac{N - M}{2} \log \beta + \frac{1}{2} \log |K_{MM}|$$

$$- \frac{1}{2} \log \left| \frac{1}{\beta} K_{MM} + K_{MN} K_{NM} \right| - \frac{\beta}{2} y^T y$$

$$+ \frac{\beta}{2} y^T K_{NM} \left( \frac{1}{\beta} K_{MM} + K_{MN} K_{NM} \right)^{-1} K_{NM} y$$

$$- \frac{\beta}{2} \text{Tr}(K_{NN}) + \frac{\beta}{2} \text{Tr}(K_{MM}^{-1} (K_{MN} K_{NM})).$$

(D.1)

Here, $N$ refers to the number of data points and $M$ is the number of inducing inputs. Noting that the matrix

$$P = \frac{1}{\beta} K_{MM} + K_{MN} K_{NM}$$

(D.2)
D.1 Variational Sparse GP Regression

is the sum of two symmetric matrices and is therefore itself symmetric, we perform Cholesky factorisations to obtain

\[ P = L_M^T L_A^T L_A L_M, \]  

(D.3)

where

\[ A = \frac{1}{\beta} I + C \]  

(D.4)

and

\[ C = (K_{NM} L_M^{-1})^T (K_{NM} L_M^{-1}). \]  

(D.5)

Following this, several terms simplify, and the objective function may be implemented more robustly and more efficiently as

\[
F_V = -\frac{N}{2} \log(2\pi) + \frac{N - M}{2} \log \beta - \frac{\beta}{2} y^T y - \sum_i \log[L_A]_{ii} + \frac{\beta}{2} \text{Tr}(D^T D) - \frac{\beta}{2} \left( \text{Tr}(K_{NN}) - \text{Tr}(C) \right),
\]  

(D.6)

where we have used

\[ D = L_A^{-T} L_M^{-T} K_{MN} y. \]  

(D.7)

D.1.1 Gradients

In this section, we present the gradient equations with respect to the kernel hyperparameters \( \theta_f \) as well as the inducing inputs \( X_u \) without giving any proofs. In order to ease the notation, we refer to the inducing inputs using \( w_{ij} = (X_u)_{ij} \).

**Gradients w.r.t. \( \theta \in \{ \sigma_f, \beta, \{ w_{ij} \}_{i,j=1}^{M,D} \} \)**

\[
\frac{\partial F_V}{\partial \theta} = -\frac{\beta}{2} \text{Tr} \left( \frac{\partial K_{NN}}{\partial \theta} \right) + \frac{1}{2} \text{Tr} \left( \frac{\partial K_{MM}}{\partial \theta} T_{MM} \right) + \beta \text{Tr} \left( \frac{\partial K_{NM}}{\partial \theta} T_{MN} \right),
\]  

(D.8)

where

\[
T_{MM} = T_x - \beta K_{MM}^{-1} K_{MN} K_{NM} K_{MM}^{-1}
\]  

(D.9)

\[
T_{MN} = T_x K_{MN} + P^{-1} K_{MN} y y^T
\]  

(D.10)
D.2 Supervised Variational GPLVM

and

\[ T_s = K_{MM}^{-1} - \frac{1}{\beta} P^{-1} - P^{-1} K_{MN} y y^T K_{NM} P^{-1} \tag{D.11} \]
\[ = L_M^T L^{-1} - \frac{1}{\beta} P^{-1} - L_M^{-1} L_A^{-1} D D^T L_A^{-T} L_M^{-T}. \tag{D.12} \]

Gradients w.r.t. \( \beta \)

\[ \frac{\partial F_V}{\partial \beta} = \frac{N - M}{2\beta} - \frac{1}{2} y^T y + \frac{1}{2\beta^2} \sum_{ik} (L_A^{-1})_{ik} (L_A^{-1})_{ik} \tag{D.13} \]
\[ + \frac{1}{2} D^T D - \frac{1}{2} E^T E - \frac{1}{2} (Tr(K_{NN}) - Tr(C)), \tag{D.14} \]

where

\[ E = L_A^{-1} L_A^{-T} L_M^{-T} K_{MN} y \tag{D.15} \]
\[ = L_A^{-1} D. \tag{D.16} \]

Note that except for \( E \), the terms in the second row are already computed when evaluating the objective function in Eqn. D.6.

D.2 Supervised Variational GPLVM

The objective function for the supervised variational GPLVM was given in Eqn. 5.20:

\[ F(q, \theta) = \tilde{F}(q, \theta) - KL(q||p) \]
\[ = \sum_{d=1}^D \log \left( \frac{e^{-\frac{1}{2} y_d^T W y_d}}{(2\pi)^{\frac{N}{2}}} \right) - \frac{\beta \psi_0}{2} + \frac{\beta}{2} Tr(K_{MM}^{-1} \Psi_2) \tag{D.17} \]
\[ - \frac{Q}{2} \log |K_Y| - \frac{1}{2} \sum_{q=1}^Q \left[ Tr(K_{Y}^{-1} S_q) + Tr(K_{Y}^{-1} \mu_q \mu_q^T) \right] + \frac{1}{2} \sum_{q=1}^Q \log |S_q| \]

with the \( \Psi \)-statistics

\[ \psi_0 = Tr \left( \langle K_{NN} \rangle_q(X) \right) \tag{D.18} \]
\[ \Psi_1 = \langle K_{NM} \rangle_q(X) \tag{D.19} \]
\[ \Psi_2 = \langle K_{MN} K_{NM} \rangle_q(X) \tag{D.20} \]

and

\[ W = \beta I_N - \beta^2 \Psi_1 (\beta \Psi_2 + K_{MM})^{-1} \Psi_1^T. \tag{D.21} \]
D.2.1 Ensuring Numerical Stability

Implementing this objective function is challenging, and attention must be paid to enforce maximum numerical stability. The techniques discussed in appendix B, including Cholesky factorisations and the inversion and determinant lemmas should be employed wherever suitable, since this often allows terms to be reorganised into simpler and more numerically stable expressions. For instance, as discussed in [Rasmussen and Williams, 2006, chap. 2.4.3], matrix inverses such as in the expression

\[(\beta \Psi^2 + K_{MM})^{-1}\]  
(D.22)

are numerically unstable due to the fact that eigenvalues of the matrix are not bounded below and may be close to zero, resulting in very large condition numbers. However, performing the Cholesky decomposition \(K_{MM} = L_M L_M^T\), the expression may be reorganised so that

\[(\beta \Psi^2 + K_{MM})^{-1} = L_M^{-T} (I + \beta L_M^{-1} \Psi^2 L_M^{-T})^{-1} L_M^{-1}.\]  
(D.23)

Now, the eigenvalues of the matrix \(A = I + \beta L_M^{-1} \Psi^2 L_M^{-T}\) are bounded within the interval \([1, 1 + \frac{M}{4} \max_{ij}(K_{MM})_{ij}\) (cf. [Rasmussen and Williams, 2006, chap. 3.4.3]), resulting in better numerical stability. Since \(\Psi^2\) is symmetric positive definite, \(A\) is also symmetric positive definite, so after computing the Cholesky decomposition of \(A\), we have

\[(\beta \Psi^2 + K_{MM})^{-1} = L_M^{-T} L_A^{-T} L_M^{-1}.\]  
(D.24)

A further example for where the identities of appendix B are useful is the computation of the reparameterised covariance matrices \(S_i\) of the variational distribution (Eqn. 5.28):

\(S_i = (K_Y^{-1} + \Lambda_i)^{-1}\).  
(D.25)

Again, this expression is not numerically stable, since the condition number is not bounded. Since \(\Lambda\) is a diagonal matrix whose inverse is trivial to compute, the matrix inversion lemma (Eqn. B.32) may be applied:

\((K_Y^{-1} + \Lambda_i)^{-1} = \Lambda_i^{-1} - \Lambda_i^{-1}(K_Y + \Lambda_i^{-1})^{-1} \Lambda_i^{-1}.\)  
(D.26)
Performing the Cholesky decomposition \( \Lambda_i = \Lambda_i^{\frac{1}{2}} \Lambda_i^{\frac{1}{2}} \), we have
\[
\Lambda_i^{-1} + K_Y = \Lambda_i^{\frac{1}{2}} (I + \Lambda_i^{\frac{1}{2}} K_Y \Lambda_i^{\frac{1}{2}}) \Lambda_i^{\frac{1}{2}}.
\]
(D.27)

Setting \( \tilde{B}_i = I + \Lambda_i^{\frac{1}{2}} K_Y \Lambda_i^{\frac{1}{2}} \), a further Cholesky factorisation results in
\[
\hat{B}_i = (\Lambda_i^{-1} + K_Y)^{-1} = \Lambda_i^{\frac{1}{2}} L_{\tilde{B}_i}^T L_{\tilde{B}_i}^{-1} \Lambda_i^{\frac{1}{2}}.
\]
(D.29)

which is implemented by first solving the equation \( L_{\tilde{B}_i} X = \Lambda_i^{\frac{1}{2}} \) using forward substitution so that \( X = L_{\tilde{B}_i}^{-1} \Lambda_i^{\frac{1}{2}} \) after which \( \hat{B}_i = X^T X \).

Furthermore, with the help of the matrix determinant lemma (Eqn. B.33), we have
\[
|S_i^{-1}| = |(K_Y^{-1} + \Lambda_i)|^{-1} = \Lambda_i^{-1} + K_Y |\Lambda_i||K_Y^{-1}|,
\]
(D.30)

so that
\[
|S_i| = |S_i^{-1}|^{-1} = \frac{1}{|\Lambda_i^{-1} + K_Y| |\Lambda_i||K_Y^{-1}|}.
\]
(D.33)

Applying these concepts to Eqn. D.17 and including the reparameterisation of the parameters \( \mu_i \) and \( S_i \) of the variational distribution (Eqn. 5.28), the implementation of the objective function in Eqn. D.17 becomes
\[
F(q, \theta) = \tilde{F}(q, \theta) - KL(q||p)
= -\frac{N D}{2} \log 2\pi - \frac{D(N - M)}{2} \log \beta - D \sum_{k=1}^{M} \log |L_k|_{kk} - \frac{\beta}{2} \sum_{m=1}^{N} \sum_{k=1}^{D} (Y \circ Y)_{mk}
+ \frac{\beta}{2} \sum_{i=1}^{M} \sum_{k=1}^{D} (P \circ P)_{ik} - \frac{\beta D \psi_0}{2} + \frac{\beta D}{2} \text{Tr}(C)
- \frac{Q N}{2} - \frac{1}{2} \sum_{i=1}^{Q} \sum_{k=1}^{N} \left[ \sum_{i=1}^{Q} \hat{B}_i \circ K_Y \right]_{mk} + \sum_{n=1}^{N} \sum_{k=1}^{D} \left( K_Y \circ \left( \sum_{i=1}^{Q} \bar{\mu}_i \bar{\mu}_i^T \right) \right)_{nk} + \sum_{i=1}^{Q} \sum_{k=1}^{N} \log |L_{\tilde{B}_i}|_{kk}.
\]
(D.35)
where

\[ C = L_M^{-1}\Psi_2L_M^{-T} \] (D.36)

\[ \hat{A} = C + \frac{1}{\beta}I \] (D.37)

\[ P = L_A^{-1}L_M^{-1}(\Psi_1^T Y). \] (D.38)

### D.2.2 Gradients

In order to apply a gradient descent optimiser to this objective function, the gradients with respect to the parameters

\[ \theta_f = \left\{ \sigma_f, \{\alpha_i\}_{i=1}^Q, \beta, X_U \right\} \] (D.39)

\[ \theta_p = \{\sigma_p, \alpha_p\} \] (D.40)

\[ \theta_{var} = \{\mu_i, S_i\}_{i=1}^Q \] (D.41)

must be computed analytically, since the computational complexity of the evaluation of the objective function (Eqn. D.35) rules out the use of a finite difference procedure. However, it is vital that the correctness of the gradients computed using the analytic gradient equations is checked by comparing against the values obtained using finite difference approximations. The derivation of the gradient equations fills several pages so that only the results are given in the following.

**Gradient w.r.t. \( \beta \)**

\[
\frac{\partial \tilde{F}}{\partial \beta} = \frac{D(N-M)}{2\beta} - \frac{1}{2}Tr(YY^T) + \frac{1}{2}Tr(PP^T) - \frac{D}{2}\Psi_0 + \frac{D}{2}Tr(C) + \frac{1}{\beta^2}Tr \left( L_A^{-T} L_A^{-1} \right) + \frac{1}{2\beta}Tr \left( \tilde{P} \tilde{P}^T \right),
\] (D.42)

where we have used

\[ \hat{A} = \Psi_2 + \frac{1}{\beta}K_{MM} \] (D.43)

\[ \tilde{P} = L_A^{-T}P. \] (D.44)
If the noise is parameterised more directly using the variance \( \eta = \frac{1}{\beta} \), the gradient with respect to \( \beta \) is easily transformed using the chain rule:

\[
\frac{\partial \tilde{F}}{\partial \eta} = \frac{\partial \tilde{F}}{\partial \beta} \frac{1}{\partial \eta} \\
= -\frac{\partial \tilde{F}}{\partial \beta} \frac{1}{\eta^2}
\]

(D.45) \hspace{1cm} \text{(D.46)}

**Gradient w.r.t. } \theta_f \in \{ \sigma_f, \{\alpha\}_i, \{(X_U)_{ij}\}_{i,j=1} \} \}

\[
\frac{\partial \tilde{F}}{\partial \theta_f} = -\frac{\beta D}{2} \frac{\partial \psi_0}{\partial \theta_f} + \beta \text{Tr} \left( \frac{\partial \Psi_1^T}{\partial \theta_f} YB^T \right) \hspace{1cm} \text{(D.47)}
\]

\[
+ \frac{1}{2} \text{Tr} \left[ \frac{\partial K_{MM}}{\partial \theta_f} \left( T - \beta D \left( L_l - T \right) M \right) \right] + \frac{\beta}{2} \text{Tr} \left( \frac{\partial \Psi_2^T}{\partial \theta_f} T \right),
\]

where

\[
\begin{align*}
P_1 &= L_M^T \hat{A}^{-T} \\
B &= (Y^T \hat{A}^{-1})^T = P_1 P \\
T &= D L_M^T L_M^{-1} - \frac{D}{\beta} P_1 P_1^T - BB^T.
\end{align*}
\]

(D.48)

Note that here the variational hyperparameters \( X_U \) are treated as kernel parameters, since they only ever appear inside the kernel.

**Gradient w.r.t. } \mu_{ik} \}

\[
\frac{\partial \tilde{F}}{\partial \mu_{ik}} = \left[ K_Y \frac{\partial \tilde{F}}{\partial \mu_i} \right]_k - \mu_{ik} \hspace{1cm} \text{(D.49)}
\]

where

\[
\frac{\partial \tilde{F}}{\partial \mu_i} = \frac{\beta D}{2} \frac{\partial \psi_0}{\partial \mu_i} + \beta \text{Tr} \left( \frac{\partial \Psi_1^T}{\partial \mu_i} YB^T \right) + \frac{\beta}{2} \text{Tr} \left( \frac{\partial \Psi_2}{\partial \mu_i} T \right). \hspace{1cm} \text{(D.50)}
\]
D.2 Supervised Variational GPLVM

Gradient w.r.t. $\lambda_{ik}$

Since the likelihood term only models variances, only the diagonal elements of the derivative w.r.t. $S_i$ are non-zero. Setting $s_{ik} = (S_i)_{kk}$, we have

$$\frac{\partial \tilde{F}}{\partial s_{ik}} = -\frac{\beta D}{2} \frac{\partial \psi_0}{\partial s_{ik}} + \beta \text{Tr} \left( \frac{\partial \Psi T}{\partial s_{ik}} YB^T \right) + \frac{\beta}{2} \text{Tr} \left( \frac{\partial \Psi_2}{\partial s_{ik}} T \right).$$  \hspace{1cm} (D.51)

Using this together with the reparameterisation equation (Eqn. 5.28) and applying the chain rule, we find

$$\frac{\partial F}{\partial \lambda_{ik}} = -\left[ (S_i \circ S_i) \frac{\partial \tilde{F}}{\partial s_i} \right]_{kk} - \frac{1}{2} \left[ (S_i \circ S_i) \lambda_i \right]_{kk}. \hspace{1cm} (D.52)$$

Gradient w.r.t. $\theta_p$

$$\frac{\partial F}{\partial \theta_p} = \sum_k \left\{ \text{Tr} \left[ \frac{\partial K_Y}{\partial \theta_p} \left( -\frac{1}{2} \left( \hat{B}_k K_Y \hat{B}_k + \bar{\mu}_k \bar{\mu}_k^T \right) \right) + \frac{\partial K_Y}{\partial \theta_p} \left( I - \hat{B}_k K_Y \right) \text{diag} \left( \frac{\partial \tilde{F}}{\partial s_k} \right) \left( I - \hat{B}_k K_Y \right)^T \right] \right\}$$  \hspace{1cm} (D.53)

where $\hat{B}_k$ is defined as in Eqn. D.28.

D.2.3 $\Psi$-Statistics

The $\Psi$-statistics are the expectations of the kernel over the variational distribution [Titsias and Lawrence, 2010]. In this section, $x_n$, $n \in \{1, \ldots, N\}$ are the latent space projections of the $N$ data points and in order to simplify notation, we use $w_m$, $m \in \{1, \ldots, M\}$ to denote the $M$ inducing inputs $X_U$. Following Titsias and Lawrence [2010], we give the statistics for the RBF automatic relevance determination (ARD) kernel

$$k(x_i, x_j) = \gamma \exp \left( -\frac{1}{2} \sum_{k=1}^{Q} \alpha_k (x_{ik} - x_{jk})^2 \right) \hspace{1cm} (D.54)$$

where we have set $\gamma = \sigma_f^2$. Because the $\Psi$-statistics only appear in the likelihood part $\tilde{F}$ of the objective function which does not model correlation between the data points, only
the diagonals $s_i$ of the covariance matrices $S_i$ appear in the following equations. For convenience of notation, we take $s_i$ to be a diagonal covariance matrix where necessary.

$\psi_0$-Statistic

$$\psi_0 = Tr(\langle K_{NN} \rangle_{q(X)})$$

$$= \sum_{n=1}^{N} \int k(x_n, x_n)N(x_n | \mu_n, s_n)dx_n$$

$$= N \gamma$$

(D.55)

Gradient w.r.t. $\gamma$

$$\frac{\partial \psi_0}{\partial \gamma} = \frac{\partial}{\partial \gamma} N \gamma$$

$$= N$$

(D.56)

$$= \psi_0 \gamma$$

(D.57)

(D.58)

$\Psi_1$-Statistic

$$\langle \Psi_1 \rangle_{nm} = \langle K_{NM} \rangle_{q(X)}$$

$$= \int k(x_n, w_m)N(x_n | \mu_n, s_n)dx_n$$

(D.59)

(D.60)

Using Eqn. D.54, we have

$$\langle \Psi_1 \rangle_{nm} = \gamma \prod_{i=1}^{Q} \frac{\exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^{\frac{1}{2}}} \right)}{(\alpha_i s_{ni} + 1)^{\frac{1}{2}}}.$$  

(D.61)

Gradient w.r.t. $\gamma$

$$\frac{\partial \langle \Psi_1 \rangle_{nm}}{\partial \gamma} = \frac{\partial}{\partial \gamma} \gamma \prod_{i=1}^{Q} \frac{\exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^{\frac{1}{2}}} \right)}{(\alpha_i s_{ni} + 1)^{\frac{1}{2}}}$$

$$= \langle \Psi_1 \rangle_{nm} \frac{\gamma}{\gamma}$$

(D.62)

(D.63)
Gradients w.r.t. \( \{\alpha_k\}_{k=1}^Q \)

\[
\frac{\partial (\Psi_1)_{nm}}{\partial \alpha_k} = \frac{\partial}{\partial \alpha_k} \gamma \prod_{i=1}^Q \exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^2} \right) \frac{1}{(\alpha_i s_{ni} + 1)^2} \tag{D.64}
\]

\[
= -\frac{1}{2} \left( \frac{s_{nk} + (\mu_{nk} - w_{mk})^2}{\alpha_k s_{nk} + 1} \right) \delta_{n'n} (\Psi_1)_{nm} \tag{D.65}
\]

Gradients w.r.t. \( \{\mu_{nk}\}_{n,k=1}^N,Q \)

\[
\frac{\partial (\Psi_1)_{nm}}{\partial \mu_{n'k}} = \frac{\partial}{\partial \mu_{n'k}} \gamma \prod_{i=1}^Q \exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^2} \right) \frac{1}{(\alpha_i s_{ni} + 1)^2} \tag{D.66}
\]

\[
= -\alpha_k \frac{\mu_{n'k} - w_{mk}}{(\alpha_k s_{n'k} + 1)} \delta_{n'n} (\Psi_1)_{nm} \tag{D.67}
\]

Gradients w.r.t. \( \{s_{nk}\}_{n,k=1}^N,Q \)

\[
\frac{\partial (\Psi_1)_{nm}}{\partial s_{n'k}} = \frac{\partial}{\partial s_{n'k}} \gamma \prod_{i=1}^Q \exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^2} \right) \frac{1}{(\alpha_i s_{ni} + 1)^2} \tag{D.68}
\]

\[
= \frac{1}{2} \left( \frac{\alpha_k^2 (\mu_{n'k} - w_{mk})^2}{(\alpha_k s_{n'k} + 1)^2} - \frac{\alpha_k}{\alpha_k s_{n'k} + 1} \right) \delta_{n'n} (\Psi_1)_{nm} \tag{D.69}
\]

Gradients w.r.t. \( \{w_{mk}\}_{m,k=1}^M,Q \)

\[
\frac{\partial (\Psi_1)_{nm}}{\partial w_{m'k}} = \frac{\partial}{\partial w_{m'k}} \gamma \prod_{i=1}^Q \exp \left( -\frac{1}{2} \frac{\alpha_i (\mu_{ni} - w_{mi})^2}{(\alpha_i s_{ni} + 1)^2} \right) \frac{1}{(\alpha_i s_{ni} + 1)^2} \tag{D.70}
\]

\[
= \alpha_k \frac{\mu_{nk} - w_{m'k}}{\alpha_k s_{nk} + 1} \delta_{m'm} (\Psi_1)_{nm} \tag{D.71}
\]

\(\Psi_2\)-Statistic

\[
\Psi_2 = \sum_{n=1}^N \Psi_2^n \tag{D.72}
\]
where $\Psi_n^2 \in \mathbb{R}^{M \times M}$ is defined as

$$
\Psi_n^2 = \langle K_{NM}K_{M'N} \rangle_q(x),
$$
(D.73)

so that

$$
(\Psi_n^2)_{mm'} = \int k(x_n, w_m) k(w_{m'}, x_n) N(x_n | \mu_n, s_n) dx_n.
$$
(D.74)

Inserting Eqn. D.54, we have

$$
(\Psi_n^2)_{mm'} = \gamma^2 \prod_{i=1}^{Q} \exp \left( -\frac{\alpha(i(w_{mi} - w_{mi}'))^2}{4} - \frac{\alpha(i(\mu_{ni} - \bar{w}_i))^2}{2\alpha_i s_{ni} + 1} \right),
$$
(D.75)

where

$$
\bar{w}_i = \frac{(w_{mi} + w_{m'i})}{2}.
$$
(D.76)

Gradient w.r.t. $\gamma$

$$
\frac{\partial (\Psi_n^2)_{mm'}}{\partial \gamma} = \frac{\partial}{\partial \gamma} \gamma^2 \prod_{i=1}^{Q} \frac{\exp \left( -\frac{\alpha_i(w_{mi} - w_{mi}'))^2}{4} - \frac{\alpha_i(\mu_{ni} - \bar{w}_i))^2}{2\alpha_i s_{ni} + 1} \right)}{(2\alpha_i s_{ni} + 1)^{\frac{3}{2}}},
$$
(D.77)

$$
= \frac{2}{\gamma} (\Psi_n^2)_{mm'}.
$$
(D.78)

Gradients w.r.t. $\{\alpha_k\}_{k=1}^{Q}$

$$
\frac{\partial (\Psi_n^2)_{mm'}}{\partial \alpha_k} = - \left[ \frac{s_{nk} + (\mu_{nk} - \bar{w}_k)^2}{2\alpha_k s_{nk} + 1} + \frac{(w_{mk} - w_{m'k})^2}{4} \right] (\Psi_n^2)_{mm'}
$$
(D.79)

From this, it follows for the derivative of the full $\Psi_2$ statistic:

$$
\frac{\partial (\Psi_2)_{mm'}}{\partial \alpha_k} = \sum_{n=1}^{N} \frac{\partial (\Psi_n^2)_{mm'}}{\partial \alpha_k}
$$

$$
= - \frac{(w_{mk} - w_{m'k})^2}{4} (\Psi_2)_{mm'} - \sum_{n=1}^{N} \frac{s_{nk} + (\mu_{nk} - \bar{w}_k)^2}{2\alpha_k s_{nk} + 1} (\Psi_n^2)_{mm'}
$$
(D.80)
Gradients w.r.t. \{\mu_{nk}\}_{n,k=1}^{N,Q}

\[
\frac{\partial (\Psi_n^2)_{mm'}}{\partial \mu_{n'k}} = \frac{\partial}{\partial \mu_{n'k}} \gamma^2 \prod_{i=1}^{Q} \exp \left( -\frac{\alpha_i (w_{mi} - w_{mi'})^2}{4} - \frac{\alpha_i (\mu_{ni} - \bar{w}_i)^2}{2\alpha_i s_{ni} + 1} \right) \left(2\alpha_i s_{ni} + 1\right)^{-\frac{1}{2}} 
= -2\alpha_k \frac{(\mu_{n'k} - \bar{w}_k)}{2\alpha_k s_{n'k} + 1} \delta_{m'n'}(\Psi_2^n)_{mm'}
\] (D.82)

Gradients w.r.t. \{s_{nk}\}_{n,k=1}^{N,Q}

\[
\frac{\partial (\Psi_n^2)_{mm'}}{\partial s_{n'k}} = \frac{\partial}{\partial s_{n'k}} \gamma^2 \prod_{i=1}^{Q} \exp \left( -\frac{\alpha_i (w_{mi} - w_{mi'})^2}{4} - \frac{\alpha_i (\mu_{ni} - \bar{w}_i)^2}{2\alpha_i s_{ni} + 1} \right) \left(2\alpha_i s_{ni} + 1\right)^{-\frac{1}{2}} 
= \left[2\alpha_k^2 \frac{(\mu_{n'k} - \bar{w}_k)^2}{(2\alpha_k s_{n'k} + 1)^2} - \frac{\alpha_k}{2\alpha_k s_{n'k} + 1}\right] \delta_{m'n'}(\Psi_2^n)_{mm'}
\] (D.84)

Gradients w.r.t. \{w_{mk}\}_{m,k=1}^{M,Q}

\[
\frac{\partial (\Psi_n^2)_{mm'}}{\partial w_{lk}} = \frac{\partial}{\partial w_{lk}} \gamma^2 \prod_{i=1}^{Q} \exp \left( -\frac{\alpha_i (w_{mi} - w_{mi'})^2}{4} - \frac{\alpha_i (\mu_{ni} - \bar{w}_i)^2}{2\alpha_i s_{ni} + 1} \right) \left(2\alpha_i s_{ni} + 1\right)^{-\frac{1}{2}} 
= \left(-\frac{1}{2} \alpha_k (w_{mk} - w_{m'k}) + \frac{\alpha_k}{2\alpha_k s_{nk} + 1} (\mu_{nk} - \bar{w}_k)\right) \delta_{lm}(\Psi_2^n)_{mm'}
\] (D.86)

\[
\frac{\partial (\Psi_n^2)_{mm'}}{\partial w_{lk}} = \frac{\partial}{\partial w_{lk}} \gamma^2 \prod_{i=1}^{Q} \exp \left( -\frac{\alpha_i (w_{mi} - w_{mi'})^2}{4} - \frac{\alpha_i (\mu_{ni} - \bar{w}_i)^2}{2\alpha_i s_{ni} + 1} \right) \left(2\alpha_i s_{ni} + 1\right)^{-\frac{1}{2}} 
= \left(-\frac{1}{2} \alpha_k (w_{mk} - w_{m'k}) + \frac{\alpha_k}{2\alpha_k s_{nk} + 1} (\mu_{nk} - \bar{w}_k)\right) \delta_{lm}(\Psi_2^n)_{mm'}
\] (D.88)
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI ’81)*, pages 674–679, April 1981. 28


REFERENCES


K. B. Petersen and M. S. Pedersen. The matrix cookbook, 2012. 111


Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In the 12th International Conference on Artificial Intelligence and Statistics (AISTATS), volume 5, 2009. 47, 48, 61, 82, 86, 116

Michalis K. Titsias and Neil D. Lawrence. Bayesian Gaussian process latent variable model. In Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2010. 64, 83, 86, 123


