Gaussian processes for pattern recognition applications

Yan, Gao

2009

Yan, G. (2009). Gaussian processes for pattern recognition applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/19301>

<https://doi.org/10.32657/10356/19301>

Gaussian Processes for Pattern Recognition Applications

Yan Gao

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

*To my husband,
who went through the toughest time with me together*

Acknowledgments

I would like to express my deepest appreciation and sincere gratitude to my supervisors, Associate Professor Dr. Kap Luk Chan and Associate Professor Dr. Wei-Yun Yau for their invaluable advice and great support in my study at Nanyang Technological University (NTU). Dr. Chan encouraged me to go through the toughest time during the program with his patience and care. He is a mentor and a friend. He also guided me to find my research interests and led me to the world of research. He is knowledgeable and always willing to discuss every research problem with his students. Dr. Yau helped me widen my research perspective through my attachment to the Institute for Inforcomm Research (I2R). He is a caring supervisor and always sheds me light on the potentials of a research problem. This work would not have been possible without Dr. Chan and Dr. Yau's excellent guidance and persistent encouragement.

I would like to thank A*STAR graduate academy (A*GA) for granting me the prestigious A*STAR Graduate Scholarship (AGS). I also appreciate the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, for providing excellent computing facilities and services for my research.

I would like to thank Dr. Lei Wang, Dr. How-Lung Eng, Dr. Peng Li, Dr. Jiayin Zhou, Dr. Sheng Fu for their valuable suggestions and discussions on various research problems. Also, I would like to give my many thanks to Mr. Wei Huang, Mr. Junyan Wang, Ms. Tiandi Duan and other research students and staff at the Media Technology Laboratory.

Finally, I would like to dedicate this thesis to my husband and parents, for their unconditioned support and love.

Contents

Acknowledgments	i
Summary	vii
List of Figures	viii
List of Tables	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Major Contribution of the thesis	5
1.4 Organization of the thesis	7
2 Bayesian Inference and Gaussian Processes	8
2.1 Bayesian inference	8
2.2 Generalizing the linear regression model to infinite dimensions	12
2.2.1 The GRBF kernel	14
2.2.2 The bias kernel	15

2.3	Gaussian process regression - the function space view	16
2.3.1	The squared exponential (SE) covariance function	18
2.3.2	Inference in the function space	19
2.3.3	Hyperparameter selection	21
2.3.4	Gaussian process and smooth functions	22
2.4	Gaussian process latent variable models	24
2.4.1	Analysis of GPLVM	26
2.4.2	Model selection for GPLVM	29
2.5	Gaussian process classification	30
2.6	Concluding remarks	32
3	Nonparametric Similarity Learning	34
3.1	Introduction	34
3.2	Related work	36
3.3	Nonparametric similarity learning	39
3.3.1	Problem formulation	39
3.3.2	Discussions on the proposed problem formulation	41
3.3.3	The inference algorithm	43
3.3.4	Properties of the learned similarity measure	44
3.3.5	Relation to other methods	46
3.3.5.1	Relation to similarity learning by GRNN	46
3.3.5.2	Relation to learning nonparametric kernel matrix	47
3.4	Graph based clustering with the learned similarity measure	47
3.5	Experiments	49

3.5.1	Clustering with the learned similarity measure	49
3.5.1.1	Toy data	49
3.5.1.2	Real data sets	50
3.5.2	Classification with the learned similarity measure	54
3.5.2.1	Toy data	56
3.5.2.2	Real data sets	57
3.5.3	Learning semantics from texture images	59
3.6	Concluding remarks	61
4	Manifold Denoising	70
4.1	Introduction	71
4.2	Gaussian process and manifolds	72
4.2.1	Gaussian prior process manifolds	72
4.2.2	Gaussian posterior process manifolds	73
4.2.3	GPLVM and manifolds	78
4.3	Manifold denoising using GPLVM	79
4.3.1	Manifold denoising by back projection	80
4.4	Comparisons to other manifold denoising methods	82
4.5	Experimental evaluation	85
4.5.1	Toy data	86
4.5.2	Swiss roll	88
4.5.3	Image manifold denoising	89
4.5.3.1	Experimental setting	91
4.5.3.2	Experimental results	92

4.5.4	Missing value recovery by manifold denoising	97
4.5.5	Manifold denoising and distance based classification	99
4.6	Concluding remarks	101
5	Shape Matching by Region Grouping	103
5.1	Introduction	103
5.2	Related work	106
5.3	A generative model for shape exemplars	108
5.3.1	Representing shapes using DT images	108
5.3.2	Applying GPLVM on the set of shape exemplars	110
5.3.3	A generative model for shape matching	111
5.3.4	The proposed similarity measure for shape matching	111
5.4	Grouping of image regions	112
5.5	Discussions	115
5.6	Experimental evaluation	117
5.6.1	Human object class	118
5.6.1.1	GPLVM training	118
5.6.1.2	Determining latent space dimensionality	119
5.6.1.3	Image region segmentation	119
5.6.1.4	Edge based shape matching	121
5.6.1.5	Detection and segmentation performance	122
5.6.2	Multiple object classes	124
5.7	Concluding remarks	127

6	Conclusions and Recommendations	129
6.1	Conclusions	129
6.2	Recommendations for further research	132
A	Drawing Manifold by Gaussian posterior process	134
B	Handling Missing Data in Manifold Denoising	136
C	Proof of Equation (3.16)	138
	Author's Publications	140
	Bibliography	142

Summary

Gaussian process (GP) is a stochastic process that has been studied for a long time and gained wide interests in the machine learning community in recent years. In this thesis, several interesting pattern analysis problems are solved using Gaussian process.

Gaussian process models can be interpreted in two views, the weight space view and the function space view. Their different interpretations can be helpful in applying GPs to solve real problems. Its capability as functional prior inspired the original work on learning nonparametric similarity measure in this thesis. The similarity between pairwise inputs is considered a smooth function described by a GP prior. Given known similarity constraints, the model can be tuned by maximum likelihood and similarity on new inputs can be inferred. The advantage is that the learned similarity measure does not assume any parametric form. It is flexible to model various data with arbitrary distributions, and handles noise and outliers in the data.

Gaussian process latent variable model (GPLVM) is a data modeling tool that is derived based on Gaussian processes from the weight space view. It was originally proposed for visualization of high dimensional data. In this thesis, it is given a new insight from the function space view, and it is deployed in two challenging tasks. One is on establishing the connections between GPs and a class of smooth manifolds, and then the connections between GPLVM and piecewise smooth embedding of manifolds, and subsequently the development of a robust method for manifold denoising. The second one is in modeling of a set of shape exemplars for region based shape matching. Under a novel region grouping framework, an alternative region based approach to shape matching is proposed to complement the edge based approaches.

List of Figures

1.1	Number of articles found by searching exact phase “Gaussian process” in every 5-year interval on Google TM Scholar (as of end of 2008). . . .	2
1.2	The standard normal distributions in 1D with a mean of zero and standard deviations of 0.5, 1 and 2. Note that since the area under the curve is always 1, none of the three models dominates in the whole domain in terms of $p(y \mu = 0, \sigma)$. This is a property that corresponds to the Occam’s razor principle. More discussions will be given in section 2.3.3.	4
2.1	A linear regression problem. The objective is to fit a straight line to the given points in the 2D plane. The standard deviation σ of the noise distribution is set to 1.	12
2.2	Regression with a GRBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$ and $\sigma = 0.2$. The solid line is the mean of the predicted target values while the shaded area represents ± 2 times the standard deviation.	15
2.3	The same linear regression problem as in figure 2.1. The kernel used is a linear kernel plus a bias kernel $k(x_i, x_j) = x_i x_j + 1$	16
2.4	Random functions sampled from a linear GP prior with covariance function $k(x_i, x_j) = x_i x_j$	19
2.5	Random functions sampled from the GP prior with a SE covariance function $k(x_i, x_j) = \exp(-\frac{1}{2}(x_i - x_j)^2)$	21

2.6	Gaussian process regression with a SE covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-\frac{1}{2}\gamma\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$. The hyperparameters are determined to be $\alpha = 0.8830, \gamma = 1.0008$ and $\sigma = 0.1607$ by maximum marginallikelihood model selection. The parameters used to generate the sample points are $\alpha = 1, \gamma = 1$ and $\sigma = 0.2$. The solid line is the mean of the predicted distribution while the shaded area represents ± 2 times the standard deviation.	23
2.7	(a) The Gaussian distributions $p(y) = \mathcal{N}(0, \sigma^2)$. (b) Given one observed input $y = 1$, maximizing the likelihood $p(y = 1 \mu = 0, \sigma)$ gives the maximum likelihood model selection of $\sigma = 1$. It does not choose a model that is too complex (larger entropy, e.g., $\sigma = 2$) or one that is too simple (smaller entropy, e.g., $\sigma = 0.5$). Entropy of Gaussian distributions is given by $\ln(\sigma\sqrt{2\pi e})$. A different illustration can be found on page 110 of [111].	23
2.8	(a) 2D input space. (b) Reconstruction to the 2D space from the latent variables obtained using GPLVM. (c) Reconstruction to the 2D space by sampling the latent space at $x^* = \min(x) : \eta : \max(x)$	27
2.9	(a) 2D input space. (b) Reconstruction to the 2D space from the latent variables obtained using GPLVM. (c) Reconstruction to the 2D space by sampling the latent space at $x^* = \min(x) : \eta : \max(x)$	28
2.10	(a) Connecting the input samples in ascending order (color coded: blue-cyan-green-yellow-red) of the corresponding 1D latent variables. (b) The Peano curve found by SOM with 20×1 neurons.	28

3.1	Clustering on a toy data consisting of 2 Gaussian clusters. (a) clustering of the test samples with the learned similarity measure. (b) the learned pairwise similarity between test samples. (c) the learned similarity to one training sample. (d) the learned similarity to one test sample. Legend: $\triangle$ and $\circ$ training samples from the 2 clusters with known similarity constraints; $+$ and $\cdot$ clustering of test points sampled at regular grids in the data space into 2 clusters.	62
3.2	Clustering on a toy data consisting of nonlinear patterns with mislabeled samples. (a) clustering result with the learned similarity measure. (b) the learned pairwise similarity.	63
3.3	Clustering on a toy data consisting of multi-modal clusters. (a) clustering result with the learned similarity measure. (b) the learned pairwise similarity.	63
3.4	Classification of the data space with nearest neighbor classifier. Legend: $\triangle$ positive training samples; $\circ$ negative training samples; $\cdot$ test samples classified as positive; $+$ test samples classified as negative.	64
3.5	Results obtained on multi-modal Gaussian distribution.	65
3.6	Classification by nearest neighbor on nonlinear pattern with mislabeled samples.	65
3.7	Classification by nearest neighbor on multi-modal data.	66
3.8	30 texture classes.	67
3.9	In Rao and Lohse's study, after constructing the similarity matrix from psychological experiments, multi-dimensional scaling (MDS) was then applied to obtain a low dimensional embedding of the 30 texture classes. It was found that embedding in a 3-dimensional space achieved a stress of only 0.045. The 3 dimensions are found to be corresponding to periodicity, directionality and structural complexity.	68

3.10	Average ranking precision.	68
3.11	The query texture image is shown on the top-left. The following three rows show the ranking of retrieved texture classes by the ground truth, by similarity learned with the proposed method, and by Euclidean distance, respectively.	69
4.1	(a) Randomly sampling the first smooth function $f_1(x)$. (b) Randomly sampling the second smooth function $f_2(x)$. (c) The two smooth functions form a smooth manifold with an intrinsic dimensionality of 1 in the 2D space $Y = [f_1(\mathbf{x}), f_2(\mathbf{x})]$	73
4.2	(a) A randomly sampled manifold in 2D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-1/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. (b) A randomly sampled manifold in 2D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-10/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. It is self-intersecting. (c) A randomly sampled manifold in 3D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-10/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. (d)-(f) Three randomly sampled manifolds in 3D generated by a GP prior with covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-1/2\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$ from a lattice of points in 2D sampled at $\{-1 : 0.04 : 1, -1 : 0.04 : 1\}$	74

4.3	(a) A manifold in 2D with an intrinsic dimensionality of 1. (b) The 1D embedding of the manifold in (a). (c) Fitting a Gaussian posterior process to the first dimension of the manifold with respect to the embedding. The black curve is the mean of the posterior process while the gray shaded area represents $\pm 100\times$ the standard deviation of the posterior process. (d) Fitting a Gaussian posterior process to the second dimension of the manifold with respect to the embedding. (e) The reconstructed 2D manifold (the black curve) from the Gaussian posterior processes.	76
4.4	(a) A parameterization used to generate the S curve. (b) The S curve. (c) - (e) The Gaussian posterior processes learned for each of the three dimensions of the S curve. (f) Sampling more points in the parameterization space. (g) The reconstructed manifold of the points sampled in the parameterization space.	77
4.5	The process of modeling of the sine curve using GPLVM. In each subplot, the three axes (from top to bottom, left to right) show the plots of $y^{(1)}$ against x , $y^{(2)}$ against x , $y^{(2)}$ against $y^{(1)}$, respectively. The blue curve is the reconstructed data by back-projection from the latent space using the latent variables and hyperparameters learned at the current iteration.	79
4.6	The process of modeling of the spiral curve using GPLVM. In each subplot, the three axes (from top to bottom, left to right) show the plots of $y^{(1)}$ against x , $y^{(2)}$ against x , $y^{(2)}$ against $y^{(1)}$, respectively. The blue curve is the reconstructed data by back-projection from the latent space using the latent variables and hyperparameters learned at the current iteration.	80

4.7	(a) The noisy data and the underlying sine curve. (b) GPLVM with $d = 1$ upon convergence. (c) KPCA with $d = 1$. (d) Graph based diffusion with $k = 30$ upon convergence. (e) Local smoothing with $d = 1, k = 30$ upon convergence.	87
4.8	(a) First 2 dimensions of the noisy data in $\mathbb{R}^{200}$. (b) Denoising by GPLVM. (c) Denoising by KPCA. (d) Denoising by Graph based diffusion. (e) Denoising by local smoothing. The numbers shown in brackets are the normalized noise on the other 198 dimensions.	88
4.9	A detailed analysis of the denoising process of the manifold denoising algorithms.	89
4.10	ISOMAP embedding of the Swiss roll data before denoising and after denoising by different methods.	90
4.11	(a) Denoising by the proposed method. (b) Denoising by KPCA. (c) Denoising by Graph based diffusion. (d) Denoising by local smoothing using WPCA. For details refer to text.	94
4.12	Denoising by various methods at different parameter settings. The numbers shown below the plots are the corresponding SNRs. First row: original and noisy images (Gaussian $\sigma = 0.4$). Second row: GPLVM at $d = 2, 5, 10, 20$. Third row: KPCA at $d = 2, 10, 50, 200$. Fourth row: graph based diffusion at iteration = 3, 10, 20, 72. Last row: Local WPCA at iteration = 1, 5, 10, 456.	96
4.13	First row: (a) original and (b) noisy images (Gaussian $\sigma = 0.8$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.	97
4.14	First row: (a) original and (b) noisy images (salt and pepper $p = 0.15$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.	98

4.15	First row: (a) original and (b) noisy images (salt and pepper $p = 0.35$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.	98
4.16	Top row: noisy images ($\sigma = 0.4$) with missing data. The red pixels indicate missing values. Bottom row: recovered images. Four columns refer to missing amount of 20%, 40%, 60% and 80%, respectively. . . .	99
5.1	(a) Images of humans in various postures. (b) Binary silhouette images showing the shapes of humans in the corresponding images in (a). (c) Edge images of the corresponding images in (b). (d) DT images of the corresponding images in (c).	104
5.2	(a) The object edge image. The pixel at (28,2) has a nearest edge pixel at (33,14). Therefore, its distance transform is $+\sqrt{(33-28)^2 + (14-2)^2} =$ 13 (+ for pixels outside object boundary). (b) The DT image. The pixel at (28,2) has a value of 13.	109
5.3	(a) 3D view of the binary silhouette image. (b) 3D view of the DT image.	110
5.4	Similarity between shapes measured by the proposed similarity measure. This figure illustrates 3 most similar and 6 most dissimilar shapes to the shape in the center.	113
5.5	(a) An image with segmented regions. There are altogether 22 distinct regions. (b) The combination of regions that is closest to a human shape, among altogether 2^{22} such combinations.	113
5.6	The proposed shape matching approach by region grouping.	117
5.7	Combination of bottom-up and top-down constraints.	118

5.8	Top: sample DT images from the training set. Middle: reconstructed images with a 1D GPLVM model. Bottom: reconstructed images with a 2D GPLVM model. The images are sorted (left to right, top to bottom) in ascending order of the 1D latent variables of the 1D GPLVM model.	120
5.9	(a) Original image; (b) Edges detected at threshold equal to the root mean square (RMS) value of the magnitude of the gradients; (c) Edges detected at threshold = 0.1; (d) Edges detected at threshold = 0.05. . .	121
5.10	False negative rate versus false positive rate. The proposed region grouping approach is evaluated with initial segmentations into 30, 40, and 50 image regions. Correspondingly, the edge based matching (Chamfer and Hausdorff) is evaluated at three different edge detection thresholds. . .	123
5.11	Sample detection and segmentation results. (a) test images. (b) initial segmentation into 40 regions. (c) matched exemplar using the proposed region grouping approach. (d) segmentation contour using the proposed method. (e) edge detected using the threshold that gives best detection results. (f) contour of matched exemplar using Chamfer matching. (g) contour of matched exemplar using Hausdorff matching.	125
5.12	6 object classes of the COIL100 data set and their corresponding silhouette images.	126
5.13	Reconstructed DT images from the 1D latent space. The images are sorted in the ascending order of the latent variables. This is the space filling curve found by GPLVM in the 1024 dimensional input space. . .	126
5.14	From left to right: the original test image, the initial segmentation by color quantization, and the final segmentation result.	127
A.1	The procedure to generate a 3D ‘yellow ribbon’ from a few hand-drawn points in 2D.	135

List of Tables

2.1	To sample a function from a GP prior distribution with a linear covariance function	18
2.2	To sample a function from a GP prior distribution with a SE covariance function	20
2.3	Latent space dimensionality selection for clean data or data that are marginally affected by noise	30
2.4	Latent space dimensionality selection for noisy data with prior knowledge	31
3.1	UCI data sets used in the experiments. n : total number of samples in the data set. d : number of features. k : number of clusters.	51
3.2	Comparison of clustering errors on 7 UCI data sets using GRNN and the proposed method. GRNN is evaluated at 4 different Gaussian widths (see text); the proposed nonparametric similarity learning (NSL) method uses the hyperparameters tuned by maximum marginal likelihood. The one with lowest clustering error is highlighted in bold typeface.	52

3.3	Clustering errors by the proposed similarity learning method and three distance metric learning methods. NSL: the proposed nonparametric similarity learning method. Xing: distance metric learning with side information. MLCC: metric learning by collapsing classes. ITML: information theoretic metric learning. Direct: direct clustering on the original data, without learning using the constraints. Two clustering methods are used. SC: spectral clustering. KM: K-mean clustering. All experiments are repeated 40 times with 40 sets of randomly sampled similarity constraints. Mean and standard deviations of the clustering errors are shown. The one with lowest clustering error is highlighted in bold typeface.	54
3.4	Clustering errors by the proposed nonparametric similarity learning method (NSL) with spectral clustering (SC) and nonparametric kernel matrix learning (NPK) with kernel k-means (kernel KM). All experiments are repeated 40 times with 40 sets of randomly sampled similarity constraints. Mean and standard deviations of the clustering errors are shown. The one with lowest clustering error is highlighted in bold typeface.	55
3.5	Classification errors by different similarity and distance metric learning methods and support vector machines on various data sets. The number of labeled training samples in each data set is 70. All experiments are repeated 40 times by randomly splitting the data sets into training and test sets. Mean and standard deviations of the classification errors are shown. The one with lowest classification error is highlighted in bold typeface.	58

3.6	Classification errors by different similarity and distance metric learning methods and support vector machines on various data sets. The number of labeled training samples in each data set is 20. All experiments are repeated 40 times by randomly splitting the data sets into training and test sets. Mean and standard deviations of the classification errors are shown. The one with lowest classification error is highlighted in bold typeface.	58
4.1	Dependent parameters of different methods: d - the subspace (or latent space) dimensionality; k/σ - the local neighborhood size in terms of k nearest neighbors or Gaussian width σ ; $iter$ - the number of iterations for methods that are solved iteratively. ‘yes’ - the effect of the parameter is studied; ‘p.d.’ - the parameter is pre-determined heuristically; ‘-’ - the parameter does not apply to the method.	92
4.2	Classification errors before and after manifold denoising by various methods. The four data sets used are the same as in the previous section, namely, USPS handwritten digits with Gaussian noise of standard deviations of 0.4 and 0.8 ($\mathcal{N} \sim 0.4$ and $\mathcal{N} \sim 0.8$), and with salt-and-pepper noise of noise levels of 0.15 and 0.35 (SnP0.15 and SnP0.35). For each data set, two labeled sample sizes are evaluated (10 and 100). The mean and standard deviations of classification errors over 12 random splits of training and test set are shown. n_{train} : number of training samples. The lowest classification errors are highlighted in bold typeface.	100
4.3	Classification errors before and after manifold denoising. For each data set, two labeled sample sizes are evaluated (10 and 100). The mean and standard deviations of classification errors over 12 random splits of training and test set are shown. n : total number of samples; q : input space dimensionality; n_{train} : number of training samples.	101

5.1	Summary of the proposed shape matching algorithm	116
5.2	Average time used for shape matching (unit: second)	124

Chapter 1

Introduction

1.1 Motivation

Gaussian process (GP) [111, 119, 149, 148, 91, 4, 33, 80] is an elegant machine learning method that provides a principled, practical and probabilistic approach to machine learning. Although it has been discovered for more than a century, it has been introduced to the machine learning community for just more than a decade [149]. It has attracted more and more research interests in recent years (see figure 1.1 for a summary of articles on GoogleTM Scholar). Nevertheless, most of its research has been mainly reported in the machine learning community, and its wide potential application to real world problems is still in its early phases. As one of the main resources in GP, the website <http://www.gaussianprocess.org/> lists annotated bibliography of major topics in GP including regression [149, 58], classification [4, 147], model selection [92, 118], sparsification and approximations [80, 89], consistency and bounds [26, 72]. However, significant contributions in bringing GP to real practice are limited. As discussed in the book *Gaussian process for machine learning* by Rasmussen and Williams [111], this may be due to the need for “the probabilistic mind-set to understand GPs, which is not very generally appreciated”. The book was published in 2006 as a long needed systematic, unified treatment of the theoretical and practical aspects of GPs

Figure 1.1: Number of articles found by searching exact phrase “Gaussian process” in every 5-year interval on GoogleTM Scholar (as of end of 2008).

in machine learning. Neil Lawrence introduced Gaussian process latent variable models (GPLVM) in 2004 [81] and later its extensions [82, 83, 79, 48], in which Gaussian process is applied in a latent variable model. It was initially proposed for visualization of high dimensional data but was quickly found to be useful to a variety of pattern analysis problems, such as fault detection [41], style based inverse kinematics [61] and prior modeling of 3D human pose for tracking [140].

Broadly speaking, machine learning [99, 14] concerns with the design of algorithms that allow computers to learn from empirical data. In *supervised learning*, a set of labeled samples is available for the purpose of fitting a model to the data and learning an input-output mapping. The labels can be categorical which corresponds to classification problems or can be continuous which leads to regression problems. In *unsupervised learning*, such labels are not available or needed. In this thesis, GPs are explored to solve both supervised and unsupervised real world problems.

So, how do we fit a suitable model to the data? The key questions include how to choose parameters and hyperparameters¹ of the model, how to incorporate prior knowledge and how to model uncertainty and noise. Bayesian inference [68, 36] provides a principled solution to the above problems. Bayesian inference is a statistical inference framework in which all forms of uncertainty are expressed in terms of probability. The fundamental task of Bayesian inference is to infer the probabilities of unknowns given some observations, whereas the auxiliary (nuisance) variables should be marginalized (by integration).

Gaussian process methods take a Bayesian approach in data modeling. Basically, Gaussian process specifies a prior for the underlying input-output mapping function f . A function can be considered an infinite dimensional vector, and Gaussian process generalizes the idea of a multivariate Gaussian distribution to infinite dimensions. A simple Gaussian (normal) distribution of a scalar variable y is given by $p(y|\mu, \sigma) = 1/\sqrt{2\pi\sigma^2} \exp(-(y - \mu)^2/2\sigma^2)$, where μ is the mean and σ is the standard deviation. Refer to figure 1.2 for a plot of a few normal distributions of a mean of zero and different standard deviations. It is an important distribution in statistical analysis due to its convenient analytical properties. Due to the central limit theorem [112], Gaussian distribution naturally approximates measurements in the physical world. As a generalization of the Gaussian distribution, Gaussian processes share the same analytical advantages, and is widely applicable to real world problems.

Inspired by the nice properties of Gaussian process above, this thesis aims to develop methods and algorithms that put Gaussian process into action for real and challenging pattern analysis problems. Visual data processing and analysis is taken as illustrating applications of the proposed algorithms. Visual data, such as images, are of important interests in the research of machine intelligence and pattern analysis. Today, computerized visual data processing and analysis is already part of our daily life. Examples include fingerprint recognition, human detection in video surveillance, image retrieval systems, etc. The scope of research comprises feature extraction, noise removal, simi-

¹In Bayesian statistics, a hyperparameter is a parameter for the prior distribution [54].

Figure 1.2: The standard normal distributions in 1D with a mean of zero and standard deviations of 0.5, 1 and 2. Note that since the area under the curve is always 1, none of the three models dominates in the whole domain in terms of $p(y|\mu = 0, \sigma)$. This is a property that corresponds to the Occam's razor principle. More discussions will be given in section 2.3.3.

larity learning, clustering and classification, etc.

1.2 Objectives

The objectives of this thesis are to solve interesting and challenging pattern analysis problems under the Gaussian process framework including their applications to visual data. In particular, three problems are targeted in this thesis. The first problem is to learn data similarity as a smooth function governed by a Gaussian process. The objective is to learn a nonparametric similarity measure from a set of similarity constraints that improves graph-based clustering, the nearest neighbor classifier, and ranking tasks in which the learned similarity is used. An application of the proposed nonparametric similarity learning method on learning semantics from texture images is discussed. The second problem concerns with the fundamental problem of manifold denoising in data analysis. Since it is inevitable that real data come with noise in the collection process, denoising is a very useful pre-processing step in pattern analysis. The properties of

Gaussian processes in relation to manifold learning is studied and a robust algorithm for manifold denoising is proposed. Its performance on image manifold denoising is also evaluated. The last problem concerns with grouping of image regions for shape based matching. The use of Gaussian process latent variable models in modeling a set of shape exemplars to define a probabilistic similarity measure for shape matching is studied. Region grouping for shape matching is proposed as an alternative approach to the traditional edge based matching such as Chamfer and Hausdorff distance matching. To achieve this, different groupings of image regions to form the foreground and background need to be properly evaluated so that the optimal grouping can be obtained efficiently.

The potential capabilities of Gaussian process will be studied in several aspects in this thesis. In nonparametric similarity learning, GP is used to describe the prior distribution of the similarity function between pairwise inputs. In this way, the similarity function is not assumed to follow a particular parametric form. In manifold denoising, Gaussian process is used in a latent variable model so that when projecting the latent variables back to the input space, noise on the manifold can be removed. The connection between GPs and smooth manifolds is also established from this study. The Gaussian process latent variable model is then explored to model a set of shape exemplars. A probabilistic similarity measure based on the trained GPLVM model is defined for efficient shape matching.

1.3 Major Contribution of the thesis

In this thesis, novel algorithms based on the idea of Gaussian processes are proposed to solve pattern analysis problems with their applications on visual data. The major contributions of this thesis are summarized as follows:

- Provide a comprehensive review and analysis of Gaussian processes, including theory on Bayesian inference and nonparametric models, Gaussian process re-

gression and classification, and in particular, a new insight on Gaussian process latent variable models. Theories are elaborated with appropriate examples. Algorithms for sampling functions from Gaussian processes are given. A practical algorithm for GPLVM model selection is developed.

- A nonparametric similarity learning method is proposed where similarity between a pair of samples is considered a function whose distribution is described by a Gaussian process. Gaussian process is employed to generalize from a set of similarity constraints a similarity measure that determines the similarity between any two points in the data space. The key advantage of the proposed method is that it assumes neither a parametric functional form of the similarity measure, nor a particular data distribution. This is very different from many existing methods. The learned similarity measure is flexible enough to model highly nonlinear, multi-modal and noisy data. Graph based clustering, the nearest neighbor classifier, and ranking of texture images benefit from the learned similarity measure as seen from the promising results on various data sets when compared with the state-of-the-arts. An application on learning semantics from texture images is also discussed.
- A manifold denoising method is developed based on Gaussian process latent variable models. The problem in question is denoising a noisy manifold M embedded in a high dimensional space. The relationship between Gaussian processes and manifolds is first established. Denoising is achieved by (nonlinearly) back-projecting the latent variables to the feature space. Through extensive comparisons with recent methods, it is shown that the proposed method causes least distortion to the manifold structure, is more stable in terms of parameter selection, and improves the performance of distance based classification such as by the nearest neighbor rule. Its application on denoising image manifolds also shows promising results, by both visual judgment and quantitative evaluation.
- A region grouping approach to exemplar based shape matching is proposed. The

distance transformed (DT) images of the training shape exemplars are modeled by a Gaussian process latent variable model (GPLVM). Each image is first segmented into coherent image regions, and possible matches are found among the regions using the proposed probabilistic similarity measure based on the learned GPLVM model. Evaluation of this new approach against edge based matching using Chamfer and Hausdorff distance shows that the proposed method outperforms the edge based approaches in terms of both discrimination capability and matching accuracy.

1.4 Organization of the thesis

This thesis is organized as follows:

- In chapter 2, Gaussian process and related theories are introduced. The advantages of GPs and GPLVMs are discussed. Algorithms for applying and analyzing GP models are given. A practical GPLVM model selection method is proposed.
- Chapter 3 presents a novel nonparametric similarity learning method. Various experiments are carried out, including an application in learning semantics from texture images.
- In chapter 4, the proposed manifold denoising algorithm based on GPLVM is presented. Experimental comparison with three different approaches is conducted to evaluate the proposed method.
- Chapter 5 deals with the challenging shape matching problem. A novel region grouping approach to exemplar based shape matching is proposed. The method is evaluated on the INRIA person data set and the COIL100 data set.
- Finally, conclusions are drawn and future research work on GPs is discussed in chapter 6.

Chapter 2

Bayesian Inference and Gaussian Processes

In this chapter, Bayesian inference is first introduced. Bayesian linear regression is taken as an example of Bayesian inference in practice, and it also serves as an entry point to the theory of Gaussian processes. Gaussian process regression is first introduced as a generalization of the Bayesian linear regression model into infinite dimensions. Then Gaussian process regression is presented from the function space view where Gaussian process serves as a prior of functions. The formal definition of Gaussian process is then given. Gaussian process latent variable models (GPLVM) is then introduced, which is a major data modeling tool used in this thesis. An elaborative analysis of GPLVM is given and a practical method for GPLVM model selection is presented.

2.1 Bayesian inference

The fundamental theory of Bayesian inference is the Bayes' theorem.

Theorem 2.1 (Bayes' theorem). *Given two events, A and B , the conditional probability*

of A given B is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.1)$$

where $P(A)$ is the prior probability of A , $P(B|A)$ is the likelihood of B given A , and $P(A|B)$ is the posterior probability of A , given B .

An example of the application of Bayesian inference is Bayesian linear regression. Let D denote a set of n observations, $D = \{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes an input vector of d dimensions and y_i denotes the corresponding target value.

The standard linear regression model is

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^T \mathbf{w}, \\ y &= f(\mathbf{x}) + \epsilon, \end{aligned} \quad (2.2)$$

where $\mathbf{w} \in \mathbb{R}^d$ is a vector of weights, $f(\mathbf{x})$ is a linear function of the input $\mathbf{x}$, and ϵ is the additive noise in the observed target values y . In other words, the target value is a linear combination of different dimensions of input $\mathbf{x}$ plus some noise. ϵ is usually assumed to follow an *independent*, identically distributed Gaussian distribution with a mean of zero and a variance of σ^2 . For the time-being it is assumed that the function $f(\mathbf{x})$ always passes through the origin, otherwise a bias term is needed ($f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b$). Referring back to theorem 2.1, the likelihood $P(B|A)$ in this case is defined as the likelihood of the target values given the unknown parameter $\mathbf{w}$:

$$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{x}^T \mathbf{w}, \sigma^2), \quad (2.3)$$

where $\mathcal{N}(\mathbf{x}^T \mathbf{w}, \sigma^2)$ represents a Gaussian distribution of a mean of $\mathbf{x}^T \mathbf{w}$ and a variance of σ^2 .

The joint distribution of the n observations is the product of the individual likelihoods, because of the independence assumption. To simplify the notations, a vector $\mathbf{y} = [y_1, \dots, y_n]^T$ is used to denote the n observed target values, and a *design matrix* $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ is used to denote the set of n observed input vectors.

Therefore,

$$\begin{aligned}
 p(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^n p(y_i|\mathbf{x}_i, \mathbf{w}) \\
 &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i\mathbf{w})^2}{2\sigma^2}\right) \\
 &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - X\mathbf{w}\|^2\right) \\
 &= \mathcal{N}(X\mathbf{w}, \sigma^2 I).
 \end{aligned} \tag{2.4}$$

Under the framework of Bayesian inference a prior is specified over the parameters $\mathbf{w}$. A unit Gaussian distribution $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, I)$ can be chosen which is a conjugate prior [54] with respect to the normal likelihood. This ensures that the posterior is also a Gaussian distribution. By Bayes' theorem, the posterior is given by

$$p(\mathbf{w}|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w}}. \tag{2.5}$$

The likelihood

$$\begin{aligned}
 p(\mathbf{y}|X) &= \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \\
 &= \int \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - X\mathbf{w}\|^2\right) \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\|\mathbf{w}\|^2\right) d\mathbf{w} \\
 &= \mathcal{N}(0, XX^T + \sigma^2 I)
 \end{aligned} \tag{2.6}$$

is a constant with respect to $\mathbf{w}$. This likelihood function is very useful in GP model selection (maximum marginal likelihood estimation). The posterior is

$$\begin{aligned}
 p(\mathbf{w}|X, \mathbf{y}) &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w})\right) \exp\left(-\frac{1}{2}\mathbf{w}^T\mathbf{w}\right) \\
 &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T\left(\frac{1}{\sigma^2}X^T X + I\right)(\mathbf{w} - \bar{\mathbf{w}})\right) \\
 &= \mathcal{N}(\bar{\mathbf{w}}, A^{-1}),
 \end{aligned} \tag{2.7}$$

where

$$\bar{\mathbf{w}} = (\sigma^2 I + X^T X)^{-1} X^T \mathbf{y}, \quad (2.8)$$

and

$$A = \frac{1}{\sigma^2} X^T X + I. \quad (2.9)$$

To predict the target value y^* given a new input vector $\mathbf{x}^*$, one way is to estimate the most likely $\mathbf{w}$ by maximizing the posterior of equation (2.7). However, point estimate of parameters may cause overfitting, especially when the estimate is obtained from a ‘spread out’ posterior [68]. Since the objective here is not to estimate $\mathbf{w}$ but rather y^* , the Bayesian way is to first define the joint distribution of y^* and $\mathbf{w}$,

$$\begin{aligned} p(y^*, \mathbf{w} | \mathbf{x}^*, X, \mathbf{y}) &= p(y^* | \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} | X, \mathbf{y}) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^* - \mathbf{x}^{*T} \mathbf{w})^2}{2\sigma^2}\right) \frac{1}{(2\pi)^{n/2} |A|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{w} - \bar{\mathbf{w}})^T A^{-1} (\mathbf{w} - \bar{\mathbf{w}})\right), \end{aligned} \quad (2.10)$$

and $\mathbf{w}$ is then integrated out to obtain $p(y^* | \mathbf{x}^*, X, \mathbf{y})$, which is again shown to be Gaussian due to the Gaussian conjugate prior used.

$$\begin{aligned} p(y^* | \mathbf{x}^*, X, \mathbf{y}) &= \int p(y^*, \mathbf{w} | \mathbf{x}^*, X, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma^2} \mathbf{x}^{*T} A^{-1} X \mathbf{y}, \mathbf{x}^{*T} A^{-1} \mathbf{x}^*\right). \end{aligned} \quad (2.11)$$

This is called marginalization of nuisance parameters. $\mathbf{w}$ is the nuisance parameter here because we are not interested in a point estimate of its most likely value. Instead of a point estimate, $\mathbf{w}$ is marginalized so that we can obtain the interested probability $p(y^* | \mathbf{x}^*, X, \mathbf{y})$. Marginalization effectively averages over all possible values of $\mathbf{w}$, weighted by the posterior $p(\mathbf{w} | X, \mathbf{y})$. It generally reduces the risk of overfitting by estimating a single value for $\mathbf{w}$. For more details on this topic, please refer to [54].

In figure 2.1, an example of Bayesian linear regression is illustrated. A linear function is used to fit the set of data $\{x_i, y_i\}, i = 1, \dots, 8$. By predicting the target values y^*

Figure 2.1: A linear regression problem. The objective is to fit a straight line to the given points in the 2D plane. The standard deviation σ of the noise distribution is set to 1.

at densely sampled x^* values, a solid line is drawn to represent the means of the predictions and a shaded area around the line is used to represent ± 2 times the standard deviations (following the convention used in [111]). Note that the prediction always passes the origin and the variance at the origin is zero due to the assumed linear model without any bias. Regression with bias will be introduced in the next section.

2.2 Generalizing the linear regression model to infinite dimensions

The Bayesian linear regression model can be extended to nonlinear regression problems with little modification to the derivation. This is done by generalizing it to infinite dimensions using the kernel trick [31, 122, 100]. The kernel trick effectively projects the inputs into a high (possibly infinite) dimensional kernel induced feature space (kernel space), and runs a linear regression in the high dimensional space. A kernel is defined as

Definition 2.1 (Kernel function [122]). A kernel is a function k that for all $\mathbf{x}, \mathbf{z} \in X$

satisfies

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \quad (2.12)$$

where ϕ is a mapping from X to a kernel induced feature space F

$$\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \in F, \quad (2.13)$$

and $\langle, \rangle$ is the dot product of two vectors.

Now suppose we map the inputs in the linear regression problem to a high dimensional space $\mathbf{x} \mapsto \phi(\mathbf{x})$, then the prediction on a new input becomes

$$p(y^* | \mathbf{x}^*, X, \mathbf{y}) = \mathcal{N} \left(\frac{1}{\sigma^2} \phi(\mathbf{x}^*)^T A^{-1} \Phi(X)^T \mathbf{y}, \phi(\mathbf{x}^*)^T A^{-1} \phi(\mathbf{x}^*) \right), \quad (2.14)$$

where $\Phi(X) = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n)]^T$, and $A = \sigma^{-2} \Phi(X)^T \Phi(X) + I$.

It can be further derived that (making use of the matrix inversion lemma [108])

$$\begin{aligned} p(y^* | \mathbf{x}^*, X, \mathbf{y}) &= \mathcal{N}(\phi(\mathbf{x}^*)^T \Phi(X)^T (\Phi(X) \Phi(X)^T + \sigma^2 I)^{-1} \mathbf{y}, \\ &\quad \phi(\mathbf{x}^*)^T \phi(\mathbf{x}^*) - \phi(\mathbf{x}^*)^T \Phi(X)^T (\Phi(X) \Phi(X)^T + \sigma^2 I)^{-1} \Phi(X) \phi(\mathbf{x}^*)) \\ &= \mathcal{N}(\mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{y}, \\ &\quad k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X)) \end{aligned} \quad (2.15)$$

where $\mathbf{k}(\mathbf{x}^*, X) = \Phi(X) \phi(\mathbf{x}^*)$ is a vector of kernel values evaluated between $\mathbf{x}^*$ and every training input vector, and $K = \Phi(X) \Phi(X)^T$ is a matrix of kernel values evaluated between all pairs of training input vectors $\mathbf{x}_i, i = 1, \dots, n$. It is clear that equation (2.15) can be written in terms of kernel values, without explicit computation of the mapping ϕ . In this way, with a properly chosen kernel [31, 122], the expressive power of a simple linear regression model is expanded, because it is now equivalent to fitting a linear regression model in an implicitly mapped space. This kernelized linear regression model is in fact Gaussian process regression in the **weight space view**. Moreover,

Gaussian process can also be represented in the **function space view**, which will be discussed in section 2.3. The definition of Gaussian process will be introduced and its role as a prior probability of functions will be discussed later.

In the following, two kernel functions of particular interests are introduced, namely, the Gaussian radial basis function (GRBF) kernel and the bias kernel.

2.2.1 The GRBF kernel

The GRBF kernel is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp \left(-\frac{1}{2} \gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right). \quad (2.16)$$

It can be shown that the GRBF kernel is a valid kernel that implicitly maps the input space to a high dimensional kernel space and it is equal to the dot product in the kernel space. A proof is given below for the 1D case (scalar inputs) $k(x, z) = \alpha \exp(-\frac{1}{2} \gamma (x-z)^2)$. Similar results can be obtained analogously for multivariate inputs.

$$\begin{aligned} k(x, z) &= \alpha \exp \left(-\frac{1}{2} \gamma (x - z)^2 \right) \\ &= \alpha \exp \left(-\frac{1}{2} \gamma (x^2 - 2xz + z^2) \right) \\ &= \alpha \exp \left(-\frac{\gamma}{2} x^2 \right) \exp \left(-\frac{\gamma}{2} z^2 \right) \exp(\gamma xz) \\ &= \alpha \exp \left(-\frac{\gamma}{2} x^2 \right) \exp \left(-\frac{\gamma}{2} z^2 \right) \left(1 + \gamma xz + \frac{(\gamma xz)^2}{2!} + \frac{(\gamma xz)^3}{3!} + \dots \right) \\ &= \left\langle \sqrt{\alpha} \exp \left(-\frac{\gamma}{2} x^2 \right) \begin{bmatrix} 1 \\ \frac{\sqrt{\gamma}x}{1} \\ \frac{\gamma x^2}{\sqrt{2!}} \\ \frac{\gamma^{3/2} x^3}{\sqrt{3!}} \\ \vdots \end{bmatrix}, \sqrt{\alpha} \exp \left(-\frac{\gamma}{2} z^2 \right) \begin{bmatrix} 1 \\ \frac{\sqrt{\gamma}z}{1} \\ \frac{\gamma z^2}{\sqrt{2!}} \\ \frac{\gamma^{3/2} z^3}{\sqrt{3!}} \\ \vdots \end{bmatrix} \right\rangle \\ &= \langle \phi(x), \phi(z) \rangle. \end{aligned} \quad (2.17)$$

Figure 2.2: Regression with a GRBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ and $\sigma = 0.2$. The solid line is the mean of the predicted target values while the shaded area represents ± 2 times the standard deviation.

This shows that the GRBF kernel effectively maps the inputs to an infinite dimensional kernel induced feature space.

Figure 2.2 illustrates a nonlinear regression with the GRBF kernel.

2.2.2 The bias kernel

In the above derivations for Bayesian linear regression model and its generalization, it is assumed that the function that fits the data has a bias or offset of zero. Ideally a linear function should be represented as $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b$, where b is the bias. With the notion of kernel function that has been presented, it is easy to include bias in the model by making use of the bias kernel.

Now consider the augmented input vector $\hat{\mathbf{x}} = [\mathbf{x}^T, 1]^T$, the linear regression model with bias becomes $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b = \hat{\mathbf{x}}^T \hat{\mathbf{w}}$, where $\hat{\mathbf{w}} = [\mathbf{w}^T, b]^T$. The kernel function becomes $k(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) = \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_j = \mathbf{x}_i^T \mathbf{x}_j + 1 = k(\mathbf{x}_i, \mathbf{x}_j) + k_b(\mathbf{x}_i, \mathbf{x}_j)$. So, including a bias kernel in the form of $k_b(\mathbf{x}_i, \mathbf{x}_j) = 1$ or any other constant is equivalent to making inference with a bias term.

Figure 2.3: The same linear regression problem as in figure 2.1. The kernel used is a linear kernel plus a bias kernel $k(x_i, x_j) = x_i x_j + 1$.

Figure 2.3 shows the same linear regression problem as in figure 2.1. A bias kernel is added to allow offsets on the y axis. Compared with that in figure 2.1, the model is more flexible and it tolerates variations as the inputs move away from the empirical data in both directions .

Note that for the GRBF kernel, a constant term is present in the expanded kernel expression in equation (2.17) ($\phi(x = 0) = \sqrt{\alpha}$). Therefore, regression with the GRBF kernel could result in fitted curves that do not pass the origin (verified with figure 2.2). In other words, a bias kernel is implicitly included in the GRBF kernel.

2.3 Gaussian process regression - the function space view

A formal definition of Gaussian process is given as follows:

Definition 2.2 (Gaussian Process [111]). *A Gaussian process is a collection of (infinite) random variables, any finite number of which have a joint Gaussian distribution.*

Consider the analogy between Gaussian distribution and Gaussian process. We know that for a multivariate Gaussian distribution, any subset of the variables has a marginal distribution that is also multivariate normal. Gaussian process extends to infinite number of variables, and therefore it is used to describe distributions over functions.

A Gaussian process is completely specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$

$$\begin{aligned} m(\mathbf{x}) &= E(f(\mathbf{x})), \\ k(\mathbf{x}_i, \mathbf{x}_j) &= E((f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))), \end{aligned} \tag{2.18}$$

where E is the expectation operator. Note that the symbol k has been reused to represent the covariance function. It will be shown later that the covariance function is equivalent to the kernel function used in the previous section.

Since Gaussian process is used as a prior probability distribution over functions, we should be able to sample random functions from this distribution. Consider a particular Gaussian process $f(\mathbf{x}) \sim \mathcal{GP}(0, \mathbf{x}_i^T \mathbf{x}_j)$ where the function f of $\mathbf{x} \in \mathbb{R}^d$ is described by a GP with a mean function of 0 and a linear covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. Suppose we want to sample a random function from this prior evaluated at a set of points $\{(\mathbf{x}_i, y_i)\}$ such that $y_i = f(\mathbf{x}_i)$, $f(\mathbf{x}) \sim \mathcal{GP}(0, \mathbf{x}_i^T \mathbf{x}_j)$. It can be shown that functions sampled from this GP prior are linear functions. The procedure to sample functions from a GP prior with linear covariance functions is described in table 2.1. Steps 3 to 5 of table 2.1 also constitute a proof of the linearity of the sampled functions. Figure 2.4 shows a few functions sampled from the linear GP prior $f(x) \sim \mathcal{GP}(0, x_i x_j)$.

Table 2.1: To sample a function from a GP prior distribution with a linear covariance function

1. First randomly sample a set of $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \dots, n$. Denote it by the matrix X . Then given the Gaussian process prior we know that the vector $\mathbf{y} = [y_1, \dots, y_n]^T \in \mathbb{R}^n$ is normally distributed with mean 0 and a covariance matrix of $K = XX^T$.
2. To generate an instance of the normally distributed multivariate vector $\mathbf{y}$, we first sample a vector from the standard multivariate normal distribution $\mathbf{z} \in \mathbb{R}^n \sim \mathcal{N}(0, I)$, and then convert it to the desired covariance.
3. Consider a linear transformation $\mathbf{y} = W\mathbf{z}$, where W is a $n \times n$ transformation matrix. The covariance of the transformed variable becomes $E(W\mathbf{z}(W\mathbf{z})^T) = E(W\mathbf{z}\mathbf{z}^T W^T) = WW^T$. To have the desired covariance of $K = XX^T$, we can design a W as $W = [X, \mathbf{0}, \dots, \mathbf{0}]$ where the first d columns of W are X and the remaining columns are all zeros. It is easy to verify that $WW^T = XX^T$.
4. Therefore, we can obtain the desired distribution of $\mathbf{y}$ by taking the transformation $\mathbf{y} = W\mathbf{z} = X\mathbf{z}_{1:d}$, where $\mathbf{z}_{1:d}$ is the first d rows of $\mathbf{z}$. Equivalently $y_i = \mathbf{x}_i^T \mathbf{z}_{1:d}$.
5. It is observed that y_i is a linear function of $\mathbf{x}_i$. The process to sample a function from a Gaussian prior with linear covariance function is equivalent to sample a random vector $\mathbf{z}_{1:d}$ from a standard normal distribution in $\mathbb{R}^d$ as the weight vector and then the linear function is give by $y_i = \mathbf{x}_i^T \mathbf{z}_{1:d}$. Functions sampled from a GP with a linear covariance function are linear functions.

2.3.1 The squared exponential (SE) covariance function

The most commonly used nonlinear covariance function in GP models is the squared exponential covariance function, which is identical to the GRBF kernel.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp \left(-\frac{1}{2} \gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right). \quad (2.19)$$

Figure 2.4: Random functions sampled from a linear GP prior with covariance function $k(x_i, x_j) = x_i x_j$.

Table 2.2 describes the procedure to sample random functions from a GP prior distribution with a SE covariance function. It is similar to the procedure used to sample from a GP prior with a linear covariance function, except that eigen-decomposition [128] is now used to find the transformation to obtain the desired covariance. Figure 2.5 shows a few functions sampled from the GP prior $f(\mathbf{x}) \sim \mathcal{GP}(0, \exp(-1/2(x_i - x_j)^2))$.

2.3.2 Inference in the function space

Given the GP prior over functions, we can solve regression problems by making inference in the function space. Given a set of n observations, $D = \{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$, let $y_i = f(\mathbf{x}_i) + \epsilon$, where $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}_i, \mathbf{x}_j))$, and $p(\epsilon) = \mathcal{N}(0, \sigma^2)$ is independent and identically distributed Gaussian noise. We have

$$p(\mathbf{y}|X) = \mathcal{N}(\mathbf{0}, K + \sigma^2 I), \quad (2.20)$$

where K is the covariance matrix with its elements $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. This equation corresponds to the likelihood in equation (2.6) in the weight space view. Now given a new test point $\mathbf{x}^*$, its function value f^* can be inferred based on the fact that the joint

Table 2.2: To sample a function from a GP prior distribution with a SE covariance function

1. Follow the first two steps in Table 2.1, except that the covariance matrix K is obtained by evaluating the SE covariance function.
2. Consider a linear transformation $W\mathbf{z}$, where W is a $n \times n$ transformation matrix. The covariance of the transformed variable becomes $E(W\mathbf{z}(W\mathbf{z})^T) = E(W\mathbf{z}\mathbf{z}^T W^T) = WW^T$. To have the desired covariance of K , we can choose W as $W = UV^{1/2}$ where U and V are the matrices for the eigenvectors and eigenvalues of K . Note that $K = UVU^T = WW^T$.
3. Therefore, we can obtain the desired distribution of $\mathbf{y}$ by taking the transformation $\mathbf{y} = W\mathbf{z}$.

distribution of f^* and $\mathbf{y}$ is again normal (Definition 2.2: any finite subset is a joint Gaussian distribution described by the mean function and the covariance function):

$$p(\mathbf{y}, f^* | X, \mathbf{x}^*) = \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} K + \sigma^2 I & \mathbf{k}(\mathbf{x}^*, X) \\ \mathbf{k}(\mathbf{x}^*, X)^T & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix} \right), \quad (2.21)$$

where $\mathbf{k}(\mathbf{x}^*, X)$ is a vector of the covariance function values evaluated from the test sample to all training samples.

We can now make inference about the posterior $p(f^* | \mathbf{x}^*, X, \mathbf{y})$ using the Bayes' theorem.

$$p(f^* | \mathbf{x}^*, X, \mathbf{y}) = \frac{p(\mathbf{y}, f^* | X, \mathbf{x}^*)}{p(\mathbf{y} | X)}. \quad (2.22)$$

Due to the conjugate prior, the posterior is again a normal distribution:

$$p(f^* | \mathbf{x}^*, X, \mathbf{y}) = \mathcal{N}(\mu(\mathbf{x}^*), \varphi(\mathbf{x}^*)), \quad (2.23)$$

Figure 2.5: Random functions sampled from the GP prior with a SE covariance function $k(x_i, x_j) = \exp(-\frac{1}{2}(x_i - x_j)^2)$

where

$$\mu(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{y}, \quad (2.24)$$

$$\varphi(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X). \quad (2.25)$$

Note that equations (2.23) to (2.25) are exactly the same as equation (2.15) which is derived from the weight space view. Therefore we can conclude that both views lead to the same solution to regression problems. Moreover, the GRBF kernel and the SE covariance function are in fact equivalent with different interpretations in the two views.

2.3.3 Hyperparameter selection

In the GP regression model, the hyperparameters include the white noise variance σ^2 and all parameters of the chosen covariance function. For example, if a SE covariance function is used, as in $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-\frac{1}{2}\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, the additional hyperparameters are α and γ . Selecting appropriate values for these hyperparameters is known as the model selection problem [40]. The hyperparameters can be empirically set if

sufficient prior knowledge is known about the problem. They can also be tuned by cross validation. Due to the Bayesian formulation of Gaussian process regression, the hyperparameters can be tuned by maximum marginal likelihood model selection.

Equation (2.20) describes the likelihood of the observed target values $\mathbf{y}$ given X and the hyperparameters (through the kernel matrix K), i.e., $p(\mathbf{y}|X) = \mathcal{N}(\mathbf{0}, K + \sigma^2 I)$. Maximizing it with respect to the hyperparameters is called the maximum marginal likelihood model selection. For differentiable covariance functions, the hyperparameters can be efficiently optimized using gradient descent.

In figure 2.6, the hyperparameters are selected by maximum marginal likelihood. The selected parameters are close to the parameters used to generate the samples, which shows that maximizing the marginal likelihood is a feasible model selection method for GP models. Moreover, it is observed that the model selected by maximum marginal likelihood does not give a perfect fit of the empirical data. Indeed, there is a natural trade-off between model fit and model complexity [111]. This is also known as the Occam's razor principle [99]. This trade-off can be illustrated by the simple example in figure 2.7.

2.3.4 Gaussian process and smooth functions

In the weight space view, the GRBF kernel can be interpreted as implicitly mapping the input space to an infinite dimensional feature space. In the function space view, the corresponding SE covariance function can be interpreted as specifying smoothness of the functions.

The functional space view of Gaussian process regression considers that a Gaussian process prior is first specified, and then the function posterior is inferred from the observed data. In figure 2.5 we have seen that functions sampled from the Gaussian process prior with the squared exponential (SE) covariance function are smooth. This is because the SE covariance function is infinitely differentiable. Functions sam-

Figure 2.6: Gaussian process regression with a SE covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-\frac{1}{2}\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. The hyperparameters are determined to be $\alpha = 0.8830$, $\gamma = 1.0008$ and $\sigma = 0.1607$ by maximum marginallikelihood model selection. The parameters used to generate the sample points are $\alpha = 1$, $\gamma = 1$ and $\sigma = 0.2$. The solid line is the mean of the predicted distribution while the shaded area represents ± 2 times the standard deviation.

Figure 2.7: (a) The Gaussian distributions $p(y) = \mathcal{N}(0, \sigma^2)$. (b) Given one observed input $y = 1$, maximizing the likelihood $p(y = 1 | \mu = 0, \sigma)$ gives the maximum likelihood model selection of $\sigma = 1$. It does not choose a model that is too complex (larger entropy, e.g., $\sigma = 2$) or one that is too simple (smaller entropy, e.g., $\sigma = 0.5$). Entropy of Gaussian distributions is given by $\ln(\sigma\sqrt{2\pi e})$. A different illustration can be found on page 110 of [111].

pled from GP with SE covariance function have mean square derivatives of all orders, and is thus very smooth [111]. The hyperparameter γ is an indicator of how smooth the functions are. It can be shown that the mean number of level-zero upcrossings is proportional to $\sqrt{\gamma}$. A larger γ means that the covariance between two inputs $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-\frac{1}{2}\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ decreases faster as the two inputs move away from each other, and therefore the function varies more as it travels along $\mathbf{x}$.

2.4 Gaussian process latent variable models

Gaussian process latent variable models (GPLVM) is now introduced. GPLVM is initially proposed for visualization of high dimensional data in [81] and it is extended for other tasks in later works [83, 79, 48, 41, 61, 139, 140]. It has a dual formulation of a probabilistic principal component analysis (PPCA) [13]. The problem is to find the latent variables [5] $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ for a set of observed variables $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times q}$. The relationship between the latent variables and the observed variables is given by

$$\mathbf{y}_i = W\mathbf{x}_i + \epsilon, \quad (2.26)$$

where W is a $q \times d$ transformation matrix, and ϵ represents the noise. Assuming a Gaussian white noise with zero mean and a diagonal covariance matrix of $\sigma^2 I$, the distribution of $\mathbf{y}_i$ is a multivariate normal distribution:

$$p(\mathbf{y}_i|\mathbf{x}_i, W) = \mathcal{N}(W\mathbf{x}_i, \sigma^2 I). \quad (2.27)$$

Similar to the Bayesian linear regression problem, we can specify a prior distribution for the transformation matrix W , $p(W) = \prod_{j=1}^q \mathcal{N}(\mathbf{w}_j|0, I)$, where $\mathbf{w}_j$ is the j -th row

of W . Integrating over W , we obtain a marginal likelihood for Y .

$$\begin{aligned}
p(Y|X) &= \prod_{j=1}^q p(\mathbf{y}_{:,j}|X) \\
&= \prod_{j=1}^q \int p(\mathbf{y}_{:,j}|X, \mathbf{w}_j) p(\mathbf{w}_j) d\mathbf{w}_j \\
&= \prod_{j=1}^q \mathcal{N}(\mathbf{y}_{:,j}|0, XX^T + \sigma^2 I) \\
&= \frac{1}{(2\pi)^{\frac{qn}{2}} |K + \sigma^2 I|^{\frac{q}{2}}} \exp\left(-\frac{1}{2} \text{tr}((K + \sigma^2 I)^{-1} YY^T)\right),
\end{aligned} \tag{2.28}$$

where $K = XX^T$, $\mathbf{y}_{:,j}$ is the j th column of Y , and $\text{tr}()$ is the trace of the matrix. Each dimension can be modeled independently due to the diagonal covariance matrix in the noise model. Again it is observed that the likelihood can be written in terms of the dot products between X . The marginal likelihood is in fact the product of q independent Gaussian processes with a linear kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. The mapping from the latent space to the original space can be easily non-linearized by using a nonlinear kernel such as the GRBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-1/2\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$. Gradient descent optimization is often used to find the latent variables X as well as the hyperparameters α, γ, σ by maximizing the likelihood $p(Y|X)$. GPLVM can be considered simultaneously performing GP regressions in multiple dimensions. In this all the GP regressions share the same set of latent variables and the same hyperparameters. Note that the difference between the GPLVM model and GP regression is that in the GPLVM model, the latent variables X are unknown and need to be found by optimizing the likelihood. In GP regression, $\mathbf{x}$ and y are the observed variables and targets, and maximizing the likelihood only finds the optimal hyperparameters.

After the GPLVM model is trained, the likelihood of a new input variable given any latent variable can also be inferred. Specifically, denote a new input variable $\mathbf{y}^* \in \mathbb{R}^q$,

its likelihood given the any point in the latent space denoted by $\mathbf{x}^* \in \mathbb{R}^d$ is given by:

$$\begin{aligned}
 & p(\mathbf{y}^* | \mathbf{x}^*, X, Y) \\
 &= \prod_{j=1}^q p(y_j^* | \mathbf{x}^*, X, Y) \\
 &= \prod_{j=1}^q \int p(y_j^* | \mathbf{x}^*, \mathbf{w}_j) p(\mathbf{w}_j | X, Y) d\mathbf{w}_j \\
 &= \prod_{j=1}^q \mathcal{N}(y_j^* | \mathbf{y}_{:,j}^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X), k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X)) \\
 &= \mathcal{N}(\mathbf{y}^* | Y^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X), (k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, X)^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*, X)) I).
 \end{aligned} \tag{2.29}$$

The intermediate derivations are based on results in Gaussian process regression (section 2.2). The mean of the prediction is $\mu^* = Y^T (K + \sigma^2 I)^{-1} \mathbf{k}(X, \mathbf{x}^*)$, while the covariance of the prediction is a diagonal matrix with equal variance on each dimension $\sigma^{*2} = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(X, \mathbf{x}^*)^T (K + \sigma^2 I)^{-1} \mathbf{k}(X, \mathbf{x}^*)$. If we make predictions on the latent variables X , the mean of the prediction can be shown to be

$$Z = K(K + \sigma^2 I)^{-1} Y, \tag{2.30}$$

where $Z \in \mathbb{R}^{n \times q}$ can be considered the reconstructed values of the original variables Y by projecting the latent variables back to the input space. In chapter 4 it is shown that this reconstruction is a useful formulation for manifold denoising.

2.4.1 Analysis of GPLVM

In N. Lawrence's papers [81, 82], various experiments on real data sets are conducted to show the advantages of GPLVM as a visualization tool. However, it is not clear what GPLVM really does because the structure of the data is unknown. In this section a fundamental analysis of the GPLVM algorithm is given on a few artificial data sets.

In figure 2.8 (a), the input data comprise 500 points sampled randomly from a uniform distribution in $\{0 < y^{(1)} < 5, 0 < y^{(2)} < 1\}$ in $\mathbb{R}^2$. A GPLVM model with a latent space dimensionality of one ($x \in \mathbb{R}^1$) is trained on the data. Figure 2.8 (b) shows the reconstructed values by projecting the latent variables X back to the input space (using equation (2.30)). The plot is color-coded by the values of X . By sampling the latent space at $x^* = \min(x) : \eta : \max(x)$, where η is a small interval, a smoother reconstruction formed by the corresponding means of the predicted y^* (equation (2.29)) in the input space is obtained (figure 2.8 (c)). Similar results are shown in figure 2.9 for a set of points sampled from a triangular region in $\mathbb{R}^2$. It is observed that the back-projection of the 1D latent variables is a smooth curve in 2D. It passes through the input region in a way reminiscent to the space-filling curve (Peano curve) [115] of the self-organizing maps (SOM) [74]. In figure 2.10 (a), the input data are connected in ascending order of the values of the latent variables x .

Figure 2.8: (a) 2D input space. (b) Reconstruction to the 2D space from the latent variables obtained using GPLVM. (c) Reconstruction to the 2D space by sampling the latent space at $x^* = \min(x) : \eta : \max(x)$.

The underlying process in the above examples is that the GPLVM algorithm optimizes for latent variables x for the inputs $\mathbf{y}$, such that the inputs $\mathbf{y}$ can be maximally represented by a hypersurface (a curve in the 1D case) reconstructed from x . The hypersurface has an intrinsic dimensionality that is equal to the latent space dimensionality. The above examples also reveal limitations of GPLVM. Firstly, as the GPLVM algorithm is an iterative algorithm that involves optimization in a high-dimensional space

Figure 2.9: (a) 2D input space. (b) Reconstruction to the 2D space from the latent variables obtained using GPLVM. (c) Reconstruction to the 2D space by sampling the latent space at $x^* = \min(x) : \eta : \max(x)$.

Figure 2.10: (a) Connecting the input samples in ascending order (color coded: blue-cyan-green-yellow-red) of the corresponding 1D latent variables. (b) The Peano curve found by SOM with 20×1 neurons.

(the dimension of the optimization space is $n \times d + \text{number of hyperparameters}$), there are many local minima. The found latent variables may not be optimal. For example, the space filling curves found in the previous examples in figures 2.8 and 2.9 do not evenly occupy the whole input space, and there are gaps between the latent variables which causes the reconstructed curve to ‘leak’ from the input region when sampling in the range of the latent variables. Secondly and more importantly, when the latent

variable model assumption does not hold, the latent variables obtained by GPLVM are meaningless from a statistical point of view. In the above examples, the 2D input data cannot be represented by the 1D latent variables. It does not make sense to use the latent variables for classification or clustering of the original input variables in this case. For real data in high dimensions, it is very difficult to know the structure of the data, and therefore it is difficult to determine whether a GPLVM of a particular latent space dimensionality is suitable for data modeling. Nevertheless, from the experimental findings in this thesis, a practical method to select the latent space dimensionality is originated. It will be detailed in the next section.

2.4.2 Model selection for GPLVM

As mentioned earlier, the latent variables and the hyperparameters of the GPLVM model are obtained by maximizing the likelihood in equation (2.28). However, the latent space dimensionality needs to be determined beforehand. In the published works by others, a latent space of 2 or 3 dimensions is commonly used, which also facilitates visualization of the data in the latent space. However, as it can be seen from the previous section, the latent space dimensionality critically determines the representation of the data. Choosing a smaller latent space dimensionality might cause severe distortion to the representation, while a larger latent space dimensionality could cause overfitting of the data.

Based on the empirical findings in this thesis, a practical latent space dimensionality selection method is presented in table 2.3 and 2.4 for clean data and for noisy data, respectively. The proposed methods have been found to be useful in the experiments carried out in this thesis, and will be presented in later chapters. Generally, it is observed that as the latent space dimensionality increases, the flexibility of the model also increases, which results in a reconstruction of the input data from the latent space to be very close to the original data, and the estimated noise variance σ^2 is small. Therefore, for clean data or data that are only marginally affected by noise,

the latent space dimensionality can be selected by slowly increasing the latent space dimensionality until the reconstructed (back-projected) data is close to the original data. For noisy data with some prior knowledge about the noise level, the latent space dimensionality can be selected by choosing the one such that the estimated noise level by maximum marginal likelihood is closest to the prior knowledge. The proposed model selection method will be further discussed in the following chapters.

Table 2.3: Latent space dimensionality selection for clean data or data that are marginally affected by noise

- Input: data Y ; output: latent space dimensionality d .
- Start from a small d (e.g., 1), train a GPLVM model by maximum marginal likelihood to determine X and the hyperparameters.
- Compute the reconstruction Z by equation (2.30).
- Check if Z is close to Y , i.e.,

$$\frac{\|Z - Y\|_F}{\|Y\|_F} < \text{th}, \quad (2.31)$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix, and th is a small threshold. If the condition meets, the latent variables X in d dimensions are adequate in modeling the input data Y .

- If equation (2.31) does not hold, moderately increase d and repeat the previous steps.

2.5 Gaussian process classification

Gaussian process classification is briefly discussed. For binary classification, a ‘squashing’ function is needed to squash the continuous $f(\mathbf{x})$ into binary outputs. For example,

Table 2.4: Latent space dimensionality selection for noisy data with prior knowledge

- Input: data Y , prior knowledge about the noise level in the data denoted by $\dot{\sigma}$; output: latent space dimensionality d .
- Start from a small d (e.g., 1), train a GPLVM model by maximum marginal likelihood to determine X and the hyperparameters.
- Check the difference between the estimated σ from maximum marginal likelihood and $\dot{\sigma}$.

$$\delta = \|\sigma - \dot{\sigma}\|, \quad (2.32)$$

- Based on the difference, moderately increase d and repeat the previous 2 steps.
- As d increases, the model complexity increases and the estimated σ by maximum marginal likelihood decreases. Thus δ becomes smaller first and then increases with d . Choose d corresponding to the smallest δ .

using the logistic function,

$$\begin{aligned} p(y = +1|f(\mathbf{x})) &= \frac{1}{1 + \exp(f(\mathbf{x}))}, \\ p(y = -1|f(\mathbf{x})) &= 1 - p(y = +1|f(\mathbf{x})) = \frac{\exp(f(\mathbf{x}))}{1 + \exp(f(\mathbf{x}))}, \end{aligned} \quad (2.33)$$

where $f(\mathbf{x})$ is a function described by the Gaussian process prior $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}_i, \mathbf{x}_j))$.

The objective is to infer the posterior

$$\begin{aligned} & p(y^* = +1|X, \mathbf{y}, \mathbf{x}^*) \\ &= \int p(y^* = +1, f^*|X, \mathbf{y}, \mathbf{x}^*) df^* \\ &= \int p(y^* = +1|f^*) p(f^*|X, \mathbf{y}, \mathbf{x}^*) df^*. \end{aligned} \quad (2.34)$$

However, in this case, $p(f^*|X, \mathbf{y}, \mathbf{x}^*)$ is no longer analytically tractable due to the non-Gaussian likelihood in equation (2.33). Approximation methods have been proposed for both binary classification and multi-class classification, such as the Laplace approximation [147] and the expectation propagation (EP) approximation [130].

2.6 Concluding remarks

This chapter is concluded with a discussion on nonparametric models. Gaussian process models belong to the class of nonparametric models. However it should not be misunderstood that ‘nonparametric’ is equivalent to ‘no parameters’. For example, k -nearest-neighbor (KNN) classifier is a nonparametric method. We still need to determine the parameter k before using it for classification. The distinction between parametric and nonparametric methods is:

Definition 2.3 (Nonparametric models [120]). *Suppose $P(x|\mathbf{w})$ is supposed to model a distribution, and we are given data $D = \{x_1, \dots, x_n\}$. The likelihood is a function $L(D, \mathbf{w})$. A statistic $\psi(D) \in \mathbb{R}^Q$ is a mapping from the data D , it is sufficient for our model if $L(D, \mathbf{w}) = \tilde{L}(\psi(D), \mathbf{w})$. A model is parametric if there exists a sufficient statistic of size Q independent of the data set size n . The model is nonparametric otherwise. For many nonparametric methods, we cannot find any sufficient statistics significantly smaller than D itself. The well-known nearest-neighbor rule and the Parzen window estimator are nonparametric because all of D has to be available upon prediction.*

In other words, nonparametric models do not intend to fit the given data with a pre-assumed model. The data speak for themselves in predicting new instances. Generally speaking, when nonparametric models are applied, there are no assumptions about the data distribution. Therefore, nonparametric models have a wider application area than the parametric counterparts. Besides KNN, other examples of nonparametric methods include histogram estimation, kernel density estimation [64], Dirichlet process [136]

and Gaussian process, which is the central topic of this thesis.

Chapter 3

Nonparametric Similarity Learning

A similarity measure is usually needed in pattern recognition tasks, such as classification, clustering, or ranking. In this chapter, a nonparametric similarity learning method is proposed where similarity between a pair of samples is considered a function whose distribution is described by a Gaussian process. Gaussian process is employed to generalize from a set of similarity constraints a similarity measure that determines the similarity between any two points in the data space. The key advantage of the proposed method is that it assumes neither a parametric functional form of the similarity measure, nor a particular data distribution. The learned similarity measure is flexible enough to model highly nonlinear, multi-modal and noisy data. Experiments on both clustering and classification on benchmark data sets with the learned similarity measure show promising results as compared with the state-of-the-arts. An application on visual data for the task of learning semantics on texture images is also reported.

3.1 Introduction

To classify, cluster, or rank patterns, a measure of how similar or dissimilar patterns are is often needed for most algorithms. For example, the nearest neighbor classifier assigns a sample to the class that its nearest neighbor belongs to. To determine the

nearest neighbor, Euclidean distance is usually used which can be considered a kind of dissimilarity measure. In graph based clustering, a pairwise similarity matrix is all that is needed to obtain a partition of the data set. In this chapter, a nonparametric similarity learning method that employs a Gaussian process to learn the similarity measure from a set of similarity constraints is proposed. The problem addressed is finding a similarity measure that is semantically agreeing with the similarity constraints (also known as side information or must-link and cannot-link constraints in some literature). The user needs not provide exact classification of the data. It is usually more convenient for the user to provide supervision in the form of similarity of object pairs in a dyadic fashion (similar/dissimilar) [95]. Moreover, label information can be converted to similarity constraints, but not vice versa. Hence the proposed method is also applicable when class labels are available. A similarity constraint determines whether a pair of samples is similar or dissimilar. The learned similarity measure should be consistent with the constraints, giving large and small values for similar and dissimilar pairs, respectively.

It is necessary to point out the difference and connection between similarity/dissimilarity learning and distance metric learning. The main difference is that distance metric learning usually requires the learned metric to follow the axioms of a metric [44]. Similarity learning, on the other hand, is not so strict. The definition of similarity can vary among authors, depending on which properties are desired. In some works, similarity/dissimilarity measures are required to have certain properties such as reflexivity and transitivity, but not limited by the triangle inequality [23]. In [75], an asymmetric dissimilarity measure is learned. The learned similarity measure of the proposed method can be shown to be symmetric. Distance can be considered a kind of dissimilarity measure; dissimilarity can be converted to similarity and vice versa in different ways [44]. In fact, distance metric and similarity/dissimilarity are usually learned to accomplish similar goals, such as to aid clustering, classification or ranking. Therefore, distance metric learning is also included in the literature discussions and experimental studies in this chapter.

Gaussian process is employed to model the similarity between pattern pairs. Gaussian process belongs to nonparametric methods. Nonparametric methods have the advantage that they do not rely on assumptions that the data are drawn from a specific probability distribution. The pairwise similarity is considered a function and inference is made in the function space. A Gaussian process prior with a chosen covariance function is specified such that nothing is said about the similarity measure before seeing the training instances except that the function is smooth. Under a Bayesian framework, a noise model is included and the hyperparameters can be learned from the data.

3.2 Related work

As mentioned earlier, similarity/dissimilarity learning are more flexible compared with distance metric learning as they do not need to follow exactly the axioms of metric. A dissimilarity learning method for nominal data is proposed in [23]. The learned metric is transitive, but it does not satisfy the triangle inequality. A neural network approach to similarity learning is proposed in [95]. The model guarantees a non-negative and symmetric measure, but not a metric otherwise. In [75], an asymmetric dissimilarity measure is learned. The measure is a weighted sum of a fixed number of dissimilarity measures, but the weights are spatially varying. Similarity learning approaches have often been used in computer vision applications that involve human perception, such as image retrieval with relevance feedback. In [42], neural networks and support vector machines are explored to predict the user's notion of similarity. In [144], optimal adaptive filters are used to build an adaptive tree similarity learning model for image retrieval. A long-term similarity learning approach in content based image retrieval is proposed in [46]. It makes use of the feedback logs collected across retrieval sessions. The learned similarity measure is a weighted combination of similarity derived from a color model and a texture model. In [62], the signed distance to cluster boundaries learned by support vector machines is used as the similarity measure for texture image retrieval.

Distance metric learning can be considered as a special case of dissimilarity learning and has attracted a lot of research attention in recent years. Most of the existing works learn a global Mahalanobis metric function parameterized by either a covariance matrix $\mathbf{A}$ or a transformation matrix $\mathbf{W}$ as in $d(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{A} (\mathbf{x}_1 - \mathbf{x}_2)$ or $d(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_1 - \mathbf{x}_2)$ [123, 150, 59, 56, 145, 151, 129, 35, 22]. The former formulation usually leads to convex optimization based solutions. The learned metric is a global metric, which is to be applied to all samples in the data space. In [59, 56], a global Mahalanobis distance is learned to collapse samples in the same class to a single point while pushing samples in different classes away from each other. In [124], a Mahalanobis distance is also learned to maximize the between-class distance and minimize the within-class distance, but with regularization added to model the prior. In [22], a Mahalanobis distance is learned by minimizing the k-means clustering error in an unsupervised fashion.

In some works that learn a Mahalanobis distance, although the learned metric is a global one, the criterion of good metrics are based on local measures. They are better suited for multi-modal data distribution [145, 151, 129, 123]. In [145], a Mahalanobis distance metric is learned for k-nearest neighbor (kNN) classification by semidefinite programming. The criterion used for optimization is the margin between different classes while penalizing large distances between each input and its target $\mathbf{k}$ nearest neighbors only. In [151], a subset of constraints requiring points to be in close proximity are used. In [129], Fisher discriminant analysis (FDA) is combined with locality-preserving projection (LPP) to formulate a local criterion to learn a transformation matrix for dimensionality reduction.

Relatively less works learn a local similarity or distance function, such that when applied to different local neighborhoods in the data space, the measure could be different. In [47], a distance function is learned for each training sample as a combination of elementary distance functions. The set of elementary functions act as a regularizer in the distance learning problem so that the final learned metric is confined to the

combinations of the elementary distances. A related work is reported in [75], where an asymmetric dissimilarity measure is learned. In [63], linear discriminant analysis is applied locally in each neighborhood to estimate an effective local metric. In [39], the decision function constructed by support vector machines is used to determine the most discriminant direction in a local neighborhood. Local methods usually work better on nonlinear and multi-modal data.

Another related research area is constrained clustering, also known as semi-supervised clustering, in which the constraints are enforced in the procedure of a particular clustering algorithm, with or without explicitly learning of a similarity or distance function [114, 7, 143, 6, 77]. In [114], similarity constraints are introduced to a density-based clustering method. In [7], the k-means clustering algorithm is adapted to both search for clusters that meet the constraints as well as to learn a Mahalanobis distance function, while an earlier work [143] only enforces the constraints. A probabilistic framework based on Hidden Markov Random Fields (HMRFs) [55] is proposed for semi-supervised clustering in [6] in which a distortion measure in terms of divergence is learned while enforcing the constraints. In [86], a boosting method is proposed that iteratively improves the accuracy of any clustering algorithm given the similarity constraints. It learns a kernel matrix transductively. Another transductive method for clustering is reported in [69], where a kernel matrix is learned from pairwise constraints. Kernel k-means is then used to obtain a partition of the input data.

The proposed method in this chapter differs from the existing methods in several fundamental aspects. First of all, the similarity learned is not limited by a pre-determined form, such as a combination of other similarity measures or based on the output of some classifier. Compared with many distance metric learning methods, it is also not limited by a parametric function, such as the Mahalanobis distance. In the proposed method, similarity is directly modeled as a smooth function governed by a Gaussian process. Secondly, as a nonparametric method, the similarity measure learned is determined by the training samples. We do not find a sufficient statistic [71]

smaller than the given training instances. Therefore, the learned similarity between unseen pattern pairs is determined by all training instances. Lastly, although a global function with a Gaussian process prior is used to model the pairwise similarity, the experimental results show that the learned similarity measure is flexible enough to model highly nonlinear and multi-modal data, which usually can only be handled by local similarity or distance metric learning methods in the past. Due to the inclusion of a noise model, the method is also able to handle noisy and overlapping data.

3.3 Nonparametric similarity learning

The proposed similarity learning method has a unique problem formulation that models the similarity between two points in $\mathbb{R}^d$ as a function of a variable in $\mathbb{R}^{2d}$ governed by a GP prior. It is argued that this function should be smooth, hence it can be described by a Gaussian process with the SE covariance function. Inference of the similarity measure follows that of a GP regression.

3.3.1 Problem formulation

Given a set of data $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \in \mathbb{R}^d$, the objective is to learn a similarity measure $s(\mathbf{x}_i, \mathbf{x}_j)$ from a set of similarity constraints. Nothing is known about the underlying function except that it is reasonable to assume that it is a smooth function with respect to both $\mathbf{x}_i$ and $\mathbf{x}_j$. That means, a slight change in $\mathbf{x}_i$ or $\mathbf{x}_j$ should not cause an abrupt change in $s(\mathbf{x}_i, \mathbf{x}_j)$. Therefore, $s(\mathbf{x}_i, \mathbf{x}_j)$ is considered a smooth function of $2d$ free variables, where d is the dimensionality of the input space. The prior probability distribution over the similarity functions is modeled by a Gaussian Process.

$$s(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{t}) + \epsilon, \quad (3.1)$$

and

$$f(\mathbf{t}) \sim \mathcal{GP}(0, k(\mathbf{t}, \mathbf{t}')), \quad (3.2)$$

where $s(\mathbf{x}_i, \mathbf{x}_j)$ is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a noise term that handles the noise in the training set, $\mathbf{t} = [\mathbf{x}_i^T, \mathbf{x}_j^T]^T$ is a vector that is of twice the dimensionality of the input data. The covariance function $k(\mathbf{t}, \mathbf{t}')$ determines properties of the Gaussian process. In particular, sampling from a Gaussian process with a linear covariance function $k(\mathbf{t}_p, \mathbf{t}_q) = \mathbf{t}_p^T \mathbf{t}_q$ produces linear functions (refer to section 2.3). Here the squared exponential (SE) covariance function is used as a smoothness constraint:

$$k(\mathbf{t}_p, \mathbf{t}_q) = \alpha \exp \left(-\frac{1}{2} \gamma \|\mathbf{t}_p - \mathbf{t}_q\|^2 \right). \quad (3.3)$$

It has been shown that the use of the SE covariance function in equations (3.1) and (3.2) corresponds to a Bayesian linear regression model with an infinite number of basis functions. Functions sampled from it are infinitely mean square differentiable [111], and hence smooth.

After the prior of the function is defined, the posterior is inferred from the observations (the set of constraints), so that predictions can be made on new data. Denote the set of M similarity constraints

$$\begin{aligned} \mathcal{C} : (i, j) \in \mathcal{C}, \\ \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are known to be similar or dissimilar.} \end{aligned} \quad (3.4)$$

The set of training vectors for the proposed similarity learning framework is obtained from the set of constraints $\mathcal{C}$. Each similarity constraint gives two training vectors, $[\mathbf{x}_i^T, \mathbf{x}_j^T]^T$ and $[\mathbf{x}_j^T, \mathbf{x}_i^T]^T$, $(i, j) \in \mathcal{C}$. This is due to symmetry that if $\mathbf{x}_i$ is similar(dissimilar) to $\mathbf{x}_j$, $\mathbf{x}_j$ is also similar(dissimilar) to $\mathbf{x}_i$. In this way, a set of training vectors $\{\mathbf{t}_m\} \in \mathbb{R}^{2d}$, $m = 1, \dots, 2M$ is obtained.

When pairwise similarity is available (for example, a person is willing to provide a

degree of similarity between two inputs), the above formulation learns from the given similarity values. However, similarity is often available in a binary form, similar or dissimilar. In such cases, the numerical similarity is defined to be 1 for similar pairs, and 0 for dissimilar pairs. i.e., $s(\mathbf{x}_i, \mathbf{x}_j), (i, j) \in \mathcal{C}$ is defined as

$$s(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are similar} \\ 0 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are dissimilar} \end{cases} \quad (3.5)$$

In this way a target vector $\mathbf{s} = [s_1, \dots, s_{2M}]^T$ corresponding to the vectors in training set $\{\mathbf{t}_m\} \in \mathbb{R}^{2d}, m = 1, \dots, 2M$ is formed.

Given the Gaussian process prior and the training set defined above, the remaining problem is to infer the similarity between any two samples of the data set $s(\mathbf{x}_i, \mathbf{x}_j), (i, j) \notin \mathcal{C}$.

3.3.2 Discussions on the proposed problem formulation

0 – 1 similarity

The use of 0 – 1 similarity values for training instances is common in the literature. In the kernel-target alignment framework proposed by Cristianini *et al.* [32], the target kernel is also defined in similar ways. In similarity learning, the 0 – 1 realization is also commonly observed (e.g., [56, 42]). Although it may be a little heuristic to set the numerical similarity values to a particular number, the 0 – 1 setting is a relative measure that signifies ‘keeping similar pairs close’ and ‘separating dissimilar pairs’. In metric learning algorithms, there is usually a mechanism to adjust the similarity relatively to a certain scale. For examples, in learning a Mahalanobis distance, the covariance matrix A is capable of scaling the learned distance to a scale between 0 and 1. In the proposed method in this chapter, the hyperparameters can be learned to adjust the similarities to the scale of 0 – 1. In addition, the proposed method is also applicable to continuous similarity values, when they are available. An example on continuous

texture similarity learning will be given towards the end of this chapter.

Mean function of the GP prior

The GP prior on the similarity measure has a mean function of 0 and a SE covariance function. It has been discussed that the SE covariance function is used to describe a smoothness constraint on the function. What about the mean function? Given the dissimilar-0, similar-1 setting of similarity values, a mean function of 0 means that when there is not enough information from the training instances, the similarity between two test samples is most likely to be 0 (dominated by prior). Empirical findings from the experiments suggest that a mean function of 0 gives better results than a mean function of, for example, 0.5. This shows that it is better to assume two samples are dissimilar, rather than assume they are somewhat similar, when little can be inferred from the training data.

Learning with weak assumptions

The proposed method has a weak learning assumption compared with other learning algorithms. Besides there is no parametric form assumed on the learned metric, the GP prior only specifies that for nearby samples (a slight change in $\mathbf{x}_i$), their similarities to other samples ($\mathbf{x}_j$) are close, due the GP prior on $[\mathbf{x}_i^T, \mathbf{x}_j^T]^T$. While one might wonder whether such weak learning assumptions could lead to good generalization, the experimental results presented later clearly indicate that the proposed method outperforms several state-of-the-art algorithms that make use of different assumptions in learning.

3.3.3 The inference algorithm

The algorithm for estimating the hyperparameters α, γ, σ and inferring the similarity measure on new samples follows that of Gaussian Process regression. A brief summary of the similarity learning algorithm is given below. For details, please refer to chapter 2.

Due to the Gaussian process prior, the joint distribution of the $2M$ training similarity values $\mathbf{s} = [s_1, \dots, s_{2M}]^T$ obtained from the constraints is

$$\mathbf{s} \sim \mathcal{N}(\mathbf{0}, K(\mathcal{T}, \mathcal{T}) + \sigma^2 I) \quad (3.6)$$

where $\mathcal{T} = [\mathbf{t}_1, \dots, \mathbf{t}_{2M}]^T$ is the set of training vectors, $K(\mathcal{T}, \mathcal{T})$ is the covariance matrix with its elements $K_{p,q} = k(\mathbf{t}_p, \mathbf{t}_q)$. Now given a new test point $\mathbf{t}^*$, the function value f^* can be inferred based on the fact that the joint distribution of the predicted function value and the training similarity is again normal:

$$\begin{bmatrix} \mathbf{s} \\ f^* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} K(\mathcal{T}, \mathcal{T}) + \sigma^2 I & \mathbf{k}(\mathbf{t}^*, \mathcal{T}) \\ \mathbf{k}(\mathbf{t}^*, \mathcal{T})^T & k(\mathbf{t}^*, \mathbf{t}^*) \end{bmatrix} \right) \quad (3.7)$$

where $\mathbf{k}(\mathbf{t}^*, \mathcal{T})$ is a column vector of the covariance function values evaluated between the test sample to all training samples.

The posterior $p(f^* | \mathbf{t}^*, \mathcal{T}, \mathbf{s})$ is also normally distributed

$$f^* | \mathbf{t}^*, \mathcal{T}, \mathbf{s} \sim \mathcal{N}(\mu(\mathbf{t}^*), \varphi(\mathbf{t}^*)) \quad (3.8)$$

where

$$\mu(\mathbf{t}^*) = \mathbf{k}(\mathbf{t}^*, \mathcal{T})^T (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)^{-1} \mathbf{s} \quad (3.9)$$

$$\varphi(\mathbf{t}^*) = k(\mathbf{t}^*, \mathbf{t}^*) - \mathbf{k}(\mathbf{t}^*, \mathcal{T})^T (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{t}^*, \mathcal{T}) \quad (3.10)$$

The predictive value of f^* is therefore given by the mean $\mu(\mathbf{t}^*)$ of the posterior.

The hyperparameters are estimated by maximizing the likelihood in equation (3.6) by gradient descent.

3.3.4 Properties of the learned similarity measure

The learned similarity measure is shown to be symmetric, given that the similarity constraints are used symmetrically, i.e., if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, both $[\mathbf{x}_i^T, \mathbf{x}_j^T]^T$ and $[\mathbf{x}_j^T, \mathbf{x}_i^T]^T$ are included in the training set $\{\mathbf{t}_m\}$. A proof is given below.

Suppose that from the set of M constraints the matrix $\mathcal{T} = [\mathbf{t}_1, \dots, \mathbf{t}_{2M}]^T$ is formed in the way such that the first two rows are from constraint 1, and next two rows are from constraint 2, and so on. Now we make prediction for the similarity between $\mathbf{a}$ and $\mathbf{b}$. We want to show that $\mu(\tau) = \mu(\tilde{\tau})$ where $\tau = [\mathbf{a}^T, \mathbf{b}^T]^T$ and $\tilde{\tau} = [\mathbf{b}^T, \mathbf{a}^T]^T$. We have

$$\mu(\tau) = \mathbf{k}(\tau, \mathcal{T})^T (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)^{-1} \mathbf{s} \quad (3.11)$$

and

$$\mu(\tilde{\tau}) = \mathbf{k}(\tilde{\tau}, \mathcal{T})^T (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)^{-1} \mathbf{s} \quad (3.12)$$

Since the SE covariance function $k(\mathbf{t}_p, \mathbf{t}_q) = \alpha \exp(-\frac{\|\mathbf{t}_p - \mathbf{t}_q\|^2}{2\beta^2})$ is isotropic, we have

$$k(\tau, [\mathbf{x}_i^T, \mathbf{x}_j^T]^T) = k(\tilde{\tau}, [\mathbf{x}_j^T, \mathbf{x}_i^T]^T) \quad (3.13)$$

Define a permutation matrix

$$P = \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_1 \\ \mathbf{e}_4 \\ \mathbf{e}_3 \\ \vdots \\ \mathbf{e}_{2M} \\ \mathbf{e}_{2M-1} \end{bmatrix} \quad (3.14)$$

where $\mathbf{e}_z$ denotes a row vector of length $2M$ with 1 in the z th position and 0 elsewhere.

Note that $P^T = P^{-1} = P$. It is easy to see that

$$\mathbf{k}(\tau, T) = \mathbf{k}(\tilde{\tau}, PT) = P\mathbf{k}(\tilde{\tau}, T) \quad (3.15)$$

Due to the way the training set is formed, we know $\mathbf{s} = P\mathbf{s}$. Therefore

$$\begin{aligned} \mu(\tilde{\tau}) &= \mathbf{k}(\tilde{\tau}, T)^T (K(T, T) + \sigma^2 I)^{-1} \mathbf{s} \\ &= \mathbf{k}(\tau, T)^T P^T (K(T, T) + \sigma^2 I)^{-1} P \mathbf{s} \\ &= \mathbf{k}(\tau, T)^T P^{-1} (K(T, T) + \sigma^2 I)^{-1} P^{-1} \mathbf{s} \\ &= \mathbf{k}(\tau, T)^T (P(K(T, T) + \sigma^2 I)P)^{-1} \mathbf{s} \\ &= \mathbf{k}(\tau, T)^T (K(T, T) + \sigma^2 I)^{-1} \mathbf{s} \\ &= \mu(\tau), \end{aligned} \quad (3.16)$$

where it can be shown that $P(K(T, T) + \sigma^2 I)P = (K(T, T) + \sigma^2 I)$ (see appendix C).

The learned similarity measure can also be constrained to be nonnegative and within the range $[0, 1]$, if a sigmoid response function such as the logistic function is appended to the standard regression model. However, doing so will make the inference no longer analytically tractable and approximations are needed (refer to Gaussian process classification in chapter 2). To avoid errors introduced by approximations nonnegativity is not enforced on the similarity learned.

3.3.5 Relation to other methods

3.3.5.1 Relation to similarity learning by GRNN

The proposed similarity learning method is different to many existing methods on similarity or distance metric learning. It has a similar problem formulation to that in [42], which also considers the pairwise similarity measure as a function of the $2d$ catenated vectors. In [42], the general regression neural network (GRNN) is used to learn the similarities. The GRNN is a radial basis neural network, with a hidden layer of as many neurons as the number of training samples. Essentially, the learned similarity is represented as (using notations defined in this chapter)

$$f(\mathbf{t}^*) = \frac{\sum_{m=1}^{2M} s_m \exp\left(-\frac{\|\mathbf{t}^* - \mathbf{t}_m\|^2}{2\beta^2}\right)}{\sum_{m=1}^{2M} \exp\left(-\frac{\|\mathbf{t}^* - \mathbf{t}_m\|^2}{2\beta^2}\right)}, \quad (3.17)$$

where β is a free parameter that needs to be set properly. In other words, the predicted similarity $f(\mathbf{t}^*)$ is a linear combination of the training similarities $s_m, m = 1, \dots, 2M$. The weights are determined by $2M$ Gaussian-shaped basis functions centered at the training samples $\mathbf{t}_m, m = 1, \dots, 2M$. On the other hand, the predicted similarity using the proposed method given by equation (3.9) can be shown to be equivalent to first expanding the input space into a feature space defined by infinite number of Gaussian-shaped basis functions centered densely in the input space, and then applying a standard linear regression model in the feature space. Moreover, the Gaussian width β in equation (3.17) needs to be specified, while all the hyperparameters in the Gaussian process model can be optimized by maximizing the likelihood of the given similarity constraints. Furthermore, as a noise model is included, the proposed nonparametric similarity learning can handle noise and outliers.

Computationally, the proposed method shares a similar complexity in the test phase

with that of GRNN. Equation (3.9) can be rewritten as

$$\mu(\mathbf{t}^*) = \mathbf{k}(\mathbf{t}^*, T)^T \nu, \quad (3.18)$$

where $\nu = (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)^{-1} \mathbf{s}$ is a vector of the same size as $\mathbf{s}$ whose computation only involves the training samples and can be computed in the training stage after the hyperparameters are tuned. Both equations (3.18) and (3.17) involve evaluation of $2M$ Gaussian-shaped basis functions and a dot product between 2 vectors of size $2M$.

3.3.5.2 Relation to learning nonparametric kernel matrix

In [69], a method for nonparametric kernel matrices learning from similarity constraints is proposed. The main difference between the proposed methods and [69] is that the proposed method learns a similarity measure that can be applied to any unseen data. On the contrary, the nonparametric kernel matrices learning method only learns a kernel matrix in a transductive manner. In [69], ‘nonparametric’ emphasizes that the learned kernel matrix is not computed from an assumed parametric kernel function. The kernel matrix is solved directly using semidefinite programming. In the proposed method, the learned similarity measure also does not have a pre-determined parametric form. Moreover, Gaussian process is used to infer the similarity measure, and Gaussian process belongs to the class of nonparametric models [120].

3.4 Graph based clustering with the learned similarity measure

Once the similarity measure is learned, it can be used in a variety of pattern recognition problems that requires a similarity or distance measure. For clustering purposes, the learned similarity is best used with graph based clustering. Graph based clustering methods such as spectral clustering are based on eigen-decomposition of a pairwise

similarity matrix between all pairs of data points. The requirement on the similarity matrix is only that it is positive¹ and symmetric [88]. Hence, it does not pose any additional constraints on the learned similarity measure. Alternatively, Laub & Müller's feature discovery method in non-metric pairwise data [78] can be explored. Below is a brief summary of the procedure of applying spectral clustering (the NJW algorithm [102]) on the learned similarity measure. For more details on spectral clustering, the readers are referred to [102] and [88].

NJW algorithm

- Compute the similarity matrix S , $S_{ij} = s(\mathbf{x}_i, \mathbf{x}_j) = \mu([\mathbf{x}_i^T, \mathbf{x}_j^T]^T)$, between all pairs of points using the learned measure. This requires $(n^2 - n)/2$ evaluations of the measure, since the measure is symmetric and self-similarity (similarity between a point and itself) is not required. S_{ii} is set to 0.
- Compute the normalized graph Laplacian $L = D^{-1/2}SD^{-1/2}$, where the degree matrix D is a diagonal matrix whose diagonal elements are defined as $d_i = \sum_{j=1}^n S_{ij}$.
- Compute the first k eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ corresponding the k largest eigenvalues, and form the matrix $V = [\mathbf{v}_1, \dots, \mathbf{v}_k] \in \mathbb{R}^{n \times k}$.
- Form the matrix $U \in \mathbb{R}^{n \times k}$ from V by normalizing the rows of V to have norm 1.
- Treating each row of U as a point in $\mathbb{R}^k$, cluster them into k clusters using k-means.
- The clustering of the rows of U is assigned as the clustering of the original n points.

¹Since non-negativeness is not enforced in the learning algorithm, the learned similarity measure between some points may be slightly negative. These negative values are adjusted by taking the absolute value.

3.5 Experiments

In this section, the proposed nonparametric similarity learning method is evaluated on both artificial and real data sets. Graph based clustering (spectral clustering), the nearest neighbor classifier, and ranking of texture images are evaluated with the learned similarity measure. For hyperparameter tuning, a maximum of 100 function evaluations is specified.

3.5.1 Clustering with the learned similarity measure

3.5.1.1 Toy data

The proposed method is firstly evaluated on a few toy data sets to help understand the similarity measure learned. The first data comprise a total of 50 samples from two clusters of equal prior, each of which is generated by a single Gaussian distribution $p(\mathbf{x}|\omega_1) = \mathcal{N}([-1/3, 0]^T, [1/9, 0; 0, 2/9])$, $p(\mathbf{x}|\omega_2) = \mathcal{N}([1/3, 0]^T, [1/9, 0; 0, 2/9])$ (figure 3.1). The test set is made up by 441 points sampled at regular grids (21×21) in the data space. A nonparametric similarity measure is learned using similarity constraints on the 50 samples. The pairwise similarity matrix is computed using the learned measure on the 441 test points. Spectral clustering is then used to cluster the test points into 2 clusters. From figure 3.1 (a), it is observed that the partitioning of the data space given by spectral clustering using the learned similarity is close to the ground truth, despite of the presence of noise due to overlapped distributions. The learned pairwise similarity matrix on the test points are shown in figure 3.1 (b). For ease of illustration, the test points have been sorted according to the clustering result such that points belonging to cluster 1 precede those in cluster 2.

In order to better understand the similarity measure learned, contour plot is used to show the learned similarity measure to one training sample and one test sample, in figure 3.1 (c) and (d), respectively. In figure 3.1 (c), the training sample is marked by

a $\times$ symbol. Different color in the plot represents the learned similarity from different places in the data space to the marked training point. The similarity values are shown in the color bar on the right. We can see that the similarity to the marked training point is a generalization of the known similarity on the training samples to the whole data space. In figure 3.1 (d), a randomly chosen test point is marked by a $\times$ symbol. The learned similarity from the data space to this test point is similar to that in (c), showing that the similarity measure learned is able to generalize to unseen samples. However, a disturbance is observed around the marked test sample. This is because self-similarity is not enforced in the learning algorithm. However, as can be seen from the clustering results, such disturbance does not affect the clustering in any significant way.

The second toy data set represents an arbitrary nonlinear pattern (figure 3.2(a)). There are some mislabeled samples. Again, it is observed that the partitioning of the data space into 2 clusters using spectral clustering with the learned similarity measure is a graceful generalization of the training samples. The effects of the mislabeled samples are suppressed. The pairwise similarity matrix shows the learned similarity measure (figure 3.2(b)).

The third toy data represents multi-modal data, which usually causes trouble to unsupervised clustering. The results are shown in figure 3.3. It shows that the proposed nonparametric method is flexible to handle such multi-modal data well.

From the above experimental results, the proposed method successfully learns a similarity measure for slightly overlapping Gaussian clusters, nonlinear patterns with mislabeled samples, as well as multi-modal clusters, without any manual tuning of the model.

3.5.1.2 Real data sets

Experiments are conducted on 7 UCI data sets. A summary of the 7 data sets is given in table 3.1.

Table 3.1: UCI data sets used in the experiments. n : total number of samples in the data set. d : number of features. k : number of clusters.

Data set	n	d	k
Iris	150	4	3
Wine	178	13	3
Ionosphere	351	34	2
Protein	116	20	6
Diabetes	768	8	2
Breast cancer	569	30	2
Housing	506	13	3

The number of constraints for the experiments is set at $M = 10d$, where d is the number of features. By keeping the ratio between the number of constraints and the dimensionality fixed, the data sets are evaluated on a comparable basis. For each data set, $10d$ pairs of samples are randomly selected and their similarity constraints (similar or dissimilar) are determined according the data labels. This is repeated 40 times to create 40 independent runs.

Comparison with similarity learning using GRNN

The proposed method is first compared to that in [42], which has a similar formulation for similarity learning but used the general regression neural network (GRNN) to learn the similarity measure. Unlike the proposed method, GRNN has parameters that need to be tuned manually. The performance of GRNN is evaluated at 4 different Gaussian widths. On the other hand, the hyperparameters of the proposed method are learned from the similarity constraints by optimizing the likelihood in equation (3.6). Spectral clustering is applied on the learned similarity matrix using both methods. Clustering performance is evaluated by clustering error defined as [150]

$$\text{clserr} = 1 - \sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{0.5n(n-1)} \quad (3.19)$$

where $1\{\cdot\}$ is the indicator function ($1\{\text{True}\} = 1, 1\{\text{False}\} = 0$), $c_i, \hat{c}_i, i = 1, \dots, n$

are the clustering by ground truth and the clustering result by spectral clustering, respectively. This criterion measures how well the clustering result matches the ground truth in terms of whether they agree on two points belonging to the same cluster.

The results on the 7 UCI data sets are shown in table 3.2. For GRNN, the Gaussian width is set to 4 different values of $0.05\delta, 0.1\delta, 0.5\delta, \delta$, where δ is the mean pairwise Euclidean distance between the points. From the results, we can see that the clustering errors of GRNN varies with the choice of parameters. The clustering errors given by the proposed method, with the hyperparameters optimized by maximizing likelihood, is below or close the best one given by GRNN in most cases.

Table 3.2: Comparison of clustering errors on 7 UCI data sets using GRNN and the proposed method. GRNN is evaluated at 4 different Gaussian widths (see text); the proposed nonparametric similarity learning (NSL) method uses the hyperparameters tuned by maximum marginal likelihood. The one with lowest clustering error is highlighted in bold typeface.

Data set	NSL	GRNN(0.05δ)	GRNN(0.1δ)	GRNN(0.5δ)	GRNN(δ)
Iris	6.78	6.99	6.62	10.6	13.1
Wine	1.88	3.22	2.66	3.0	3.9
Ionosphere	17.6	20.38	19.8	31.97	34.73
Protein	14.57	18.56	16.73	16.77	18.53
Diabetes	44.02	42.93	42.84	42.06	42.23
Breast cancer	4.75	6.2	6.34	11.81	11.15
Housing	36.82	36.85	36.72	38.75	38.66

Comparison with distance metric learning

The proposed method is also compared with a few state-of-the-art distance metric learning methods. The advantage of distance metric learning is that since the learned measure is a valid metric, the clustering algorithm thus is not limited by spectral clustering. Both spectral clustering and k-means clustering with the learned distance metric are evaluated. The first distance metric learning method compared with is

Xing’s distance metric learning with side information (Xing) [150]. The second one is metric learning by collapsing classes (MLCC) [56]. The third one is information theoretic metric learning (ITML) [35]. All these three methods learn a Mahalanobis distance in terms of a covariance matrix A as in $\|\mathbf{x}_i - \mathbf{x}_j\|_A = (\mathbf{x}_i - \mathbf{x}_j)^T A (\mathbf{x}_i - \mathbf{x}_j)$. For spectral clustering, the pairwise similarity is obtained by converting the learned distance into $S_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_A}{\sigma})$, where σ is set to the mean of all pairwise distances.

With the same amount of similarity constraints as used in the previous experiment, the clustering results by different methods are shown in table 3.3. The proposed method outperforms the rest on 4 data sets. ITML with k-means gives best clustering errors on 2 data sets, while MLCC leads on 1. Nonetheless, compare with unsupervised clustering on the original data without learning from the similarity constraints, clustering result generally improves after learning.

Comparison with nonparametric kernel matrix learning

Lastly, a quick comparison with the nonparametric kernel matrix learning in [69] is also performed. The method in [69] learns a kernel matrix for a set of data based on the similarity constraints in a transductive manner. The code provided by the authors² is used and the default parameters are set. Since the output of the learning algorithm is a kernel matrix, it is suitable for kernel k-means clustering [37]³. The clustering results are shown in table 3.4. The proposed method gives better results on 5 out of 7 data sets. It is also observed that the performance of nonparametric kernel matrix learning is not stable over the 40 sets of randomly selected constraints on the iris and wine data sets, as indicated by the large variances.

²<http://www3.ntu.edu.sg/home/chhoi/publication.htm>

³In [69], the authors used constrained kernel k-means wherein the known constraints are enforced in the clustering procedure. However, in the experiments, it is found that constrained clustering is very unstable. It frequently outputs ‘no solution’. As discussed in [8], this is due to the greedy nature of constrained clustering. Therefore, constrained clustering is not used here.

Table 3.3: Clustering errors by the proposed similarity learning method and three distance metric learning methods. NSL: the proposed nonparametric similarity learning method. Xing: distance metric learning with side information. MLCC: metric learning by collapsing classes. ITML: information theoretic metric learning. Direct: direct clustering on the original data, without learning using the constraints. Two clustering methods are used. SC: spectral clustering. KM: K-mean clustering. All experiments are repeated 40 times with 40 sets of randomly sampled similarity constraints. Mean and standard deviations of the clustering errors are shown. The one with lowest clustering error is highlighted in bold typeface.

Data set		NSL	Xing	MLCC	ITML	Direct
Iris	sc	6.78±2.73	6.81±1.27	6.93±2.89	4.88±1.86	13.21
	km		5.38±1.2	6.22±2.52	4.47±1.64	12.63
Wine	sc	1.88±0.95	5.89±1.64	2.86±1.43	2.67±1.56	3.8
	km		7.41±1.84	2.89±1.45	2.76±1.38	5.85
Ionosphere	sc	17.6±7.37	19.81±1.51	26.52±4.02	22.19±2.95	41.82
	km		20.87±1.14	23.65±2.68	22.93±2.24	41.11
Protein	sc	14.57±2.2	18.73±1.51	16.88±2.34	16.83±2.44	20.75
	km		18.93±1.62	16.03±1.55	16.63±1.91	19.31
Diabetes	sc	44.02±2.77	43.01±2.68	44.28±3.62	44.28±3.62	44.67
	km		42.46±2.77	42.31±2.93	43.36±3.91	44.42
Breast cancer	sc	4.75±0.65	8.71±0.72	6.52±1.28	6.52±1.04	12.79
	km		10.12±0.84	7.88±1.3	7.46±1.14	13.4
Housing	sc	36.82±2.33	37.35±1.11	35.39±2.68	37.55±3.86	42.29
	km		37.39±1.07	33.51±2.95	31.55±3.53	40.09

3.5.2 Classification with the learned similarity measure

Since class labels can be converted to similarity constraints, the proposed method is also applicable in a classification setting. For a set of n_t training samples, n_t^2 similarity constraints can be obtained. Various methods such as active set selection [121] can be applied to select a subset of the available constraints to reduce the computational demand. However, the focus here is on the small training sample size problem which is a typical problem in pattern classification tasks. Data labeling process can be expensive or inconvenient, and in such cases the small sample problem is likely

Table 3.4: Clustering errors by the proposed nonparametric similarity learning method (NSL) with spectral clustering (SC) and nonparametric kernel matrix learning (NPK) with kernel k-means (kernel KM). All experiments are repeated 40 times with 40 sets of randomly sampled similarity constraints. Mean and standard deviations of the clustering errors are shown. The one with lowest clustering error is highlighted in bold typeface.

Data set	NSL with SC	NPK with kernel KM
Iris	6.78±2.73	21.64± 10.15
Wine	1.88±0.95	4.23±8.05
Ionosphere	17.6±7.37	15.69±9.67
Protein	14.57±2.2	17.31±2.54
Diabetes	44.02±2.77	46.1±2.02
Breast cancer	4.75±0.65	2.52±0.98
Housing	36.82±2.33	38.97±5.22

to cause overfitting. The experimental results show that the proposed method performs favorably even compared with state-of-the-art classifiers when the number of available training samples is small. The learned similarity measure is evaluated in the classification setting using the nearest neighbor classifier. A set of labeled samples is given, and the similarity measure is learned from the similarity constraints derived from the labeled set. Test samples are classified to the classes of the training samples that have the largest similarity to them. Since labeled samples are available, another two recent metric learning methods, namely, distance metric learning for large margin nearest neighbor classification (LMNN) [145] and local fisher discriminant analysis for supervised dimensionality reduction (LFDA) [129] are used for comparison. These two methods are ‘discrimination’ oriented to learn a distance metric from the training samples, which are better suited for classification tasks. In LMNN, a Mahalanobis distance metric for nearest neighbor classification is learned by semidefinite programming. The metric is trained with the goal that the nearest neighbors always belong to the same class while samples from different classes are separated by a large margin. LFDA is a local variant of Fisher discriminant analysis that finds the discriminant components lo-

cally so that it is better suited for multi-modal data. Further, classification by support vector machines (SVMs) [141, 19] is also included in the comparison, to gain insights into how similarity learning methods compare with the state-of-the-art classifier.

3.5.2.1 Toy data

Firstly artificially generated data are used to examine the learned similarity measure for classification. The first data is the same as in the clustering experiment: a total of 50 samples from two classes of equal prior, each of which is generated by a single Gaussian distribution $P(\mathbf{x}|\omega_1) = \mathcal{N}([-1/3, 0]^T, [1/9, 0; 0, 2/9])$, $P(\mathbf{x}|\omega_2) = \mathcal{N}([1/3, 0]^T, [1/9, 0; 0, 2/9])$ (figure 3.4). The test set is made up by 441 points sampled at regular grids (21×21) in the input space. A similarity measure is learned using the 50 labeled samples and the nearest neighbor classifier is applied on the test samples with the learned similarity measure. The classification of test samples shows a decision function that is close to an optimal Bayes decision function ($x_1 = 0$) if the underlying process to generate the samples is known. In contrast, both LMNN and LFDA produce overfitted decisions due to the slightly overlapping distributions of the two classes. The classification obtained by SVM⁴ is also satisfactory due to the use of a regularization parameter C (C-SVM) to control the tradeoff between function complexity and training error. It is necessary to highlight that the result obtained by the proposed method is achieved without any manual setting of parameters and with no prior knowledge about the decision function (e.g., linear or nonlinear).

In the next experiment, multi-modal Gaussian distributions are used. $P(\omega_1) = 2/3$, $p(\mathbf{x}|\omega_1) = 0.5\mathcal{N}([-1/2, 1/2]^T, [1/9, 0; 0, 1/9]) + 0.5\mathcal{N}([-1/2, -1/2]^T, [1/9, 0; 0, 1/9])$; $P(\omega_2) = 1/3$, $p(\mathbf{x}|\omega_2) = \mathcal{N}([1/2, 0]^T, [1/9, 0; 0, 1/9])$. Again, the decisions made by the proposed method is close to that of an optimal Bayes, as shown in figure 3.5. If the underlying distribution (mixture of Gaussians) is known, a Gaussian mixture model

⁴For fair comparison, it is assumed that no prior knowledge about whether the decision function is linear is given. Therefore, a general GRBF kernel is used in C-SVM. The kernel width is set to 0.2 and C is set to 1.

(GMM)[94] would probably give similar results, but requiring the knowledge of the number of modes in each class.

Classification on nonlinear and multi-modal data such as the two used in the clustering experiments also shows good results. The toy data in figure 3.6 represents nonlinear data with mislabeled samples. The toy data in figure 3.7 represents multi-modal data. Both results are promising. x

3.5.2.2 Real data sets

Experiments are performed on 7 UCI data sets (table 3.1) as well as 2 handwritten digit data sets (USPS and MNIST). A nearest neighbor classifier based on Euclidean distance in the original input space is also evaluated.

Two sets of training samples are used, $n_t = 70$ and $n_t = 20$, to simulate a small sample size scenario. The training samples are randomly selected, with the condition that at least two training samples from the same class are included. This is due to the fact that both LMNN and LFDA require at least one nearest neighbor from the same class for each training sample. The sets of data exhibit different levels of difficulties depending on the feature distribution, dimensionality and number of classes. The two digit data sets represent a harder classification problem, with 10 classes each, and the number of training samples is small. The gray scale pixel values are used as the features. Therefore, they are very high-dimensional data ($16 \times 16 = 256$) with limited training samples.

The average classification errors over 40 runs of random splitting of training set and test sets are shown in table 3.5 and 3.6, for $n_t = 70$ and $n_t = 20$, respectively. The proposed method generally performs better than the two metric learning methods. For classification by support vector machines, a Gaussian radius basis function (GRBF) kernel is used and 10 folds cross validation is used to choose the model parameters (kernel width and C). The proposed method also outperforms support vector machines on some data sets. Moreover, when the number of training samples is small, a nearest

neighbor classification in the original space may actually outperform some of the metric learning methods. It shows that overfitting is likely to occur with small sample size.

Table 3.5: Classification errors by different similarity and distance metric learning methods and support vector machines on various data sets. The number of labeled training samples in each data set is 70. All experiments are repeated 40 times by randomly splitting the data sets into training and test sets. Mean and standard deviations of the classification errors are shown. The one with lowest classification error is highlighted in bold typeface.

$n_t = 70$					
Data set	NSL	NN	LMNN	LFDA	SVM
Iris	6±2.16	5.66±2.47	5.22±2.28	5.22±2.45	5.41±2.54
Wine	4.17±2.05	5.6±1.82	3.1±1.71	2.25±1.37	3.13±1.22
Ionosphere	14.93±3.35	17.48±3.34	15.05±2.52	23.41±4.62	7.95±2.02
Protein	23.57±6.72	32.28±6.2	29.84±7.43	30.54±6.73	25.22±6.89
Diabetes	29.33±2.97	32.5±2.57	31.04±2.88	31.43±2.93	27.31±2.44
Breast cancer	4.8±1.21	6.32±1.07	5.59±1.39	5.5±1.37	5.11±1.47
Housing	23.64±3.17	26.07±3.1	24.33±2.95	25.98±3.32	21.15±3.18
USPS	27.34±2.82	33.34±2.56	34.48±3.25	37.28±4.11	29.12±3.23
MNIST	27.54±2.56	31.08±2.05	36.13±3.04	39.41±3.49	28.01±2.86

Table 3.6: Classification errors by different similarity and distance metric learning methods and support vector machines on various data sets. The number of labeled training samples in each data set is 20. All experiments are repeated 40 times by randomly splitting the data sets into training and test sets. Mean and standard deviations of the classification errors are shown. The one with lowest classification error is highlighted in bold typeface.

$n_t = 20$					
Data set	NSL	NN	LMNN	LFDA	SVM
Iris	8.83±4.52	8.63±4.04	7±3.21	6.12±4.72	9.4±5.57
Wine	6.61±2.78	8.1±2.82	9.18±3.56	14.35±5.8	6.57±4.78
Ionosphere	23.13±6.02	24.75±6.94	24.4±4.86	24.20±4.03	19.76±9.62
Protein	43.44±5.77	47.63±5.92	52.53±6.26	60.99±7.06	44.71±7.13
Diabetes	32.59±4.04	34.93±4.03	36.02±4.97	34.73±4.76	32.27±3.48
Breast cancer	7.63±2.78	8.88±2.87	9.86±5.39	14.89±6.08	8.93±4.37
Housing	32.18±5.82	32.99±5.78	32.08±7.49	34.56±7.25	31.2±6.31
USPS	41.44±4.27	43.82±3.61	42.07±4.66	44.63±4.87	40.53±14.41
MNIST	48.33±3.8	50.58±4.01	50.6±3.67	53.47±4.9	50.63±5.08

3.5.3 Learning semantics from texture images

In this section, an application of the proposed similarity learning method on learning semantics from texture images is discussed. Texture is one of the important visual properties that has received intense studies [90, 93, 110]. Texture descriptors describing textural properties both in the spatial image domain and frequency domain have been reported in literature. In the study of human perception, psychological experiments on textures were also reported [110]. There have been some exciting findings about texture perception. It has been identified that periodicity, directionality and structural complexity [110] are the most dominant factors in discriminating different texture classes. However, it is believed that none of the computational image features we have at our disposal directly measure or correlate with the factors that affect human perception, even though there are image features extracted from multichannel Gabor filters being thought of as a close modeling of the early human vision. It is of imminent importance and of great interests to bridge this so-called perceptual gap.

Various psychological experiments have been done to study human perception of textures. In [110], a subset of 30 images from Brodatz album were used and this subset was chosen to capture the variations in the texture across the album and was a representative subset of the original 112 pictures. The 30 texture images are shown in figure 3.8. Twenty subjects participated in the study. A bottom-up sorting procedure was adopted because it enabled the sorter to create hierarchically structured groups of similar items. Each subject was asked to group the 30 pictures into as many classes as desired. The subjects then repeatedly regrouped the pictures into smaller numbers of classes. For each of the initial groupings of the 30 textures by the 20 subjects, a lower triangular similarity matrix was constructed in which the matrix elements were filled with ones for two textures within the same group and they were filled with zeros for textures in different groups. These individual subject matrices were summed across all 20 subjects to give a pooled similarity matrix S . Multi-dimensional scaling (MDS)[28] was used to produce an embedding of the 30 texture classes in a 3D space. It is shown

in figure 3.9.

Given the known similarity matrix, the experimental setting in this section is as follows. Each texture class (a 512×512 texture image) is subdivided into 4 non-overlapping texture images. Gabor features [93] are used to describe the texture classes. In [87], it has been shown that Gabor feature is the most perceptual consistent feature among the several types of texture feature tested. Following [93], Gabor feature at 4 scales and 6 orientations are extracted and mean and variance are used to represent each band. Therefore a 48 dimensional feature vector representation of the texture image is obtained. One image from each texture class is taken out as the training set. The known similarity on the 30 texture classes is used to learn the similarity measure. Note that the proposed method is capable of learning from continuous similarity values. The remaining three images from each class form the test set. For each test image, the 30 texture classes are retrieved in an order according to the inferred similarity. The ranking precision is evaluated by

$$\text{prec}(x) = \# \{ \{ \text{top } x \text{ retrieved} \} \cap \{ \text{ground truth} \} \} \quad (3.20)$$

which is the number of images in the top x retrieved that are also in the top x images ranked according to the true similarity between texture classes. The average ranking precisions of the images in the test set are shown in figure 3.10. For comparison, 2 reference precisions are also plotted. One is the ranking produced by the ground truth, and the other one is ranking by Euclidean distance. It is observed that although Gabor features have been shown to be a good representation of texture, it is still not very consistent with human perception in terms of ranking. The proposed similarity learning method improves the consistency with human perception.

The retrieval result of one query image is shown in figure 3.11. The top 6 retrieved texture classes by the ground truth, by similarity learned with the proposed method, and by Euclidean distance are shown. In terms of ranking precision defined earlier, ranking with the learned similarity measure gives 4/6 while that of Euclidean distance

gives $3/6$.

3.6 Concluding remarks

In this chapter a novel nonparametric approach for similarity learning is proposed. Pairwise similarity is considered a function whose distribution is described by a Gaussian process. The nonparametric formulation using Gaussian process offers richer and more flexible representations of the similarity measure to be learned. The experimental results on both artificial and real data sets indicate that the learned similarity measure works successfully on nonlinear and multi-modal data. Other advantages of the proposed method include automatic hyperparameter selection and noise and outlier handling. Its capability is also demonstrated on learning continuous similarity on texture images.

Figure 3.1: Clustering on a toy data consisting of 2 Gaussian clusters. (a) clustering of the test samples with the learned similarity measure. (b) the learned pairwise similarity between test samples. (c) the learned similarity to one training sample. (d) the learned similarity to one test sample. Legend: Δ and $\circ$ training samples from the 2 clusters with known similarity constraints; $+$ and $\cdot$ clustering of test points sampled at regular grids in the data space into 2 clusters.

Figure 3.2: Clustering on a toy data consisting of nonlinear patterns with mislabeled samples. (a) clustering result with the learned similarity measure. (b) the learned pairwise similarity.

Figure 3.3: Clustering on a toy data consisting of multi-modal clusters. (a) clustering result with the learned similarity measure. (b) the learned pairwise similarity.

Figure 3.4: Classification of the data space with nearest neighbor classifier. Legend: $\triangle$ positive training samples; $\circ$ negative training samples; $\cdot$ test samples classified as positive; $+$ test samples classified as negative.

Figure 3.5: Results obtained on multi-modal Gaussian distribution.

Figure 3.6: Classification by nearest neighbor on nonlinear pattern with mislabeled samples.

Figure 3.7: Classification by nearest neighbor on multi-modal data.

Figure 3.8: 30 texture classes.

Figure 3.9: In Rao and Lohse's study, after constructing the similarity matrix from psychological experiments, multi-dimensional scaling (MDS) was then applied to obtain a low dimensional embedding of the 30 texture classes. It was found that embedding in a 3-dimensional space achieved a stress of only 0.045. The 3 dimensions are found to be corresponding to periodicity, directionality and structural complexity.

Figure 3.10: Average ranking precision.

Figure 3.11: The query texture image is shown on the top-left. The following three rows show the ranking of retrieved texture classes by the ground truth, by similarity learned with the proposed method, and by Euclidean distance, respectively.

Chapter 4

Manifold Denoising

The focus of this chapter is on denoising a noisy manifold M embedded in a high dimensional space. Here the manifold noise is considered data points that exhibits extreme variation (outliers) and possible additive noise induced in the data sampling process. These outliers and noise will cause trouble to manifold learning methods due to the ‘short circuit’ problem. The purpose of manifold denoising is to remove the outliers and noise, and to preserve the intrinsic structure of the manifold. The manifold M is a priori unknown and we are only given noisy samples. This is an unsupervised learning problem. In this chapter, a connection between Gaussian process models and a class of smooth manifolds is first established. It is observed that Gaussian prior process generates smooth manifolds while Gaussian posterior process models smooth manifolds. Further, it is observed that although fitting a Gaussian process latent variable model to manifolds does not guarantee the recovery of the correct low-dimensional embedding (as other manifold learning algorithms such as ISOMAP does), it provides a promising solution to noise removal on the high dimensional manifold. Through extensive comparisons with alternative approaches, the performance of the proposed method is shown to be promising.

4.1 Introduction

A manifold is a topological object which has an intrinsic dimensionality smaller than the input space dimensionality. It can usually be ‘unfolded’ to obtain a low dimensional embedding. Many real data sets form a manifold structure in the input space. For example, a set of images of hands may only have two intrinsic parameters characterizing the gesture, but when represented by their gray scale, the dimensionality of the input space is as high as the number of pixels (see e.g., [133] for some illustrations, or figure 4.10 for the typical Swiss roll data used in manifold learning).

Dealing with manifolds is an important research area in machine learning. Relevant studies comprise nonlinear dimension reduction [133, 113, 9], clustering [125], semi-supervised learning [10, 20], etc. Their applications encompass face tracking [85], image manifold analysis [38, 98], human activity recognition [24], motion segmentation [57] and image retrieval [65]. Manifold learning techniques usually come with strong restrictions and assumptions on the data. For example, ISOMAP [133], LLE [113] and Laplacian Eigenmap [9] are based on preserving local neighborhoods. They are prone to noise and outliers which could cause ‘short-circuit’ in the construction of the local neighborhood graph. Moreover, noise could cause trouble to distance based classifiers, especially when the manifold is residing in a high-dimensional space. Hence, manifold denoising is usually an essential preprocessing step to facilitate other manifold learning or classification tasks [67, 105, 152].

In this chapter, the connection between Gaussian processes and a class of smooth manifolds is first elaborated by a few examples. Two concepts are defined, namely, the Gaussian prior process manifolds and the Gaussian posterior process manifolds. It is observed that Gaussian prior process is a generating process of smooth manifolds, while a Gaussian posterior process efficiently models a given manifold and its intrinsic parameterization. Based on the findings, a manifold denoising algorithm using Gaussian process latent variable models (GPLVM) is proposed. Recall that a noise model is explicitly defined in GPLVM (section 2.4), hence making it natural to use it for

denoising. In the method proposed in this chapter, noise is removed in the process of mapping the latent variables back to the feature space. From the weight space view, it can be shown that this reconstruction depends only on the dot products between latent variables. This makes nonlinear modeling and reconstruction possible through Gaussian processes. From the function space view, it is proposed in this thesis to use the squared exponential (SE) covariance function in the Gaussian process so that the manifold structure can be recovered by fitting smooth functions from the latent space to the input space, even in the presence of high dimensional noise in the input space.

Extensive experiments are conducted to study the performance of denoising by GPLVM compared with other state-of-the-art methods. The focuses of the study include evaluating the distortion to the manifold after denoising, evaluating the dependency on parameters, and evaluating denoising as preprocessing for classification. The performance of the proposed method is shown to be promising. It causes least distortion to the manifold topology. It is also more stable in terms of model selection. Lastly, it improves the performance of distance based classification such as by the nearest neighbor rule.

4.2 Gaussian process and manifolds

4.2.1 Gaussian prior process manifolds

Firstly a class of smooth manifolds called the Gaussian prior process manifolds is defined.

Definition 4.1. *A Gaussian prior process manifold is a manifold that is generated by mapping points in a low dimensional space to each dimension of a higher dimensional space by random sampling from a smooth GP prior (e.g., a SE covariance function).*

An example is shown in figure 4.1. In figure 4.1 (a) and (b), a line segment defined by points at $x = \{-1 : 0.02 : 1\}$ is mapped to each dimension of a 2D space by

randomly sampling two smooth functions $y_1 = f_1(x)$ and $y_2 = f_2(x)$ from a GP prior $f(x) \sim \mathcal{GP}(0, k(x_i, x_j))$, where $k(x_i, x_j) = \exp(-1/2(x_i - x_j)^2)$. These two smooth functions form a smooth manifold with an intrinsic dimensionality of 1 in the 2D space $Y = [f_1(\mathbf{x}), f_2(\mathbf{x})]$, where $\mathbf{x} = [-1 : 0.02 : 1]^T$ (figure 4.1 (c)). The manifold can be parameterized by a single parameter x .

Figure 4.1: (a) Randomly sampling the first smooth function $f_1(x)$. (b) Randomly sampling the second smooth function $f_2(x)$. (c) The two smooth functions form a smooth manifold with an intrinsic dimensionality of 1 in the 2D space $Y = [f_1(\mathbf{x}), f_2(\mathbf{x})]$.

Figure 4.2 shows more examples of Gaussian prior process manifolds generated by different covariance functions. It is observed that the Gaussian prior process manifolds are smooth and can be self-intersecting. This could potentially allow efficient parameterizations of complex manifolds such as intersecting paths which are common in natural data sets [126].

4.2.2 Gaussian posterior process manifolds

Given a smooth manifold and its intrinsic parameterization, the manifold can be efficiently modeled as a Gaussian posterior process.

Definition 4.2. *A Gaussian posterior process manifold is a manifold that is obtained by the posterior inferred from a given smooth manifold and its intrinsic parameterization.*

Given the manifold $Y \in \mathbb{R}^{n \times q}$ and the corresponding intrinsic parameterization $X \in \mathbb{R}^{n \times d}$, Gaussian process regression can be applied to model the mapping from X

Figure 4.2: (a) A randomly sampled manifold in 2D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-1/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. (b) A randomly sampled manifold in 2D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-10/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. It is self-intersecting. (c) A randomly sampled manifold in 3D generated by a GP prior with covariance function $k(x_i, x_j) = \exp(-10/2(x_i - x_j)^2)$ from 1D line segment $\mathbf{x} = [-1 : 0.02 : 1]^T$. (d)-(f) Three randomly sampled manifolds in 3D generated by a GP prior with covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-1/2\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ from a lattice of points in 2D sampled at $\{-1 : 0.04 : 1, -1 : 0.04 : 1\}$.

to each dimension of Y . The optimal hyperparameters can be obtained by maximizing the overall likelihood

$$\begin{aligned}
 p(Y|X) &= \prod_{j=1}^q p(\mathbf{y}_{:,j}|X) \\
 &= \prod_{j=1}^q \mathcal{N}(\mathbf{y}_{:,j}|0, K + \sigma^2 I),
 \end{aligned} \tag{4.1}$$

where K is the covariance matrix obtained from evaluating a proper covariance function such as the SE covariance function. Equation (4.1) is the same as the likelihood function

of GPLVM (equation (2.28)), except that now the latent variables X are known and only the hyperparameters are to be optimized. Inference on a set of new points in the latent space X^* gives

$$\begin{aligned}
 p(Y^*|X^*) &= \prod_{j=1}^q p(\mathbf{y}_{:,j}^*|X^*) \\
 &= \prod_{j=1}^q \mathcal{N}(\mathbf{y}_{:,j}^*|K(X^*, X)(K + \sigma^2 I)^{-1}\mathbf{y}_{:,j}, \\
 &\quad K(X^*, X^*) - K(X^*, X)(K + \sigma^2 I)^{-1}K(X^*, X)^T),
 \end{aligned} \tag{4.2}$$

where Y^* is the corresponding set of variables in the input space, $K(X^*, X)$ is a matrix of kernel values evaluated from every point in X^* to every training point in X , and $K(X^*, X^*)$ is a matrix of kernel values evaluated between every pair of points in X^* . Equation (4.2) is a product of q Gaussian posterior processes with their respective mean functions and a common covariance function. Functions can be sampled from the posterior processes. If we are only concerned about the mean of the posterior processes, it is given by

$$\Gamma = K(X^*, X)(K + \sigma^2 I)^{-1}Y. \tag{4.3}$$

An illustration is given in figure 4.3. A Gaussian posterior process is inferred for each of the 2 dimensions of the manifold with respect to the 1D unfolded embedding of the manifold, which acts as the intrinsic parameterization of the manifold. The SE covariance function is used and the hyperparameters are optimized via maximum marginal likelihood. Since the manifold itself is smooth, the mapping from the parameterization to each dimension of the manifold can be approximated by a smooth function closely. The predicted variances are very small (figure 4.3 (c) and (d)). By sampling densely in the parameterization space, a smooth Gaussian posterior process manifold can be interpolated in the input space (figure 4.3 (e)).

In figure 4.4 an illustration of the Gaussian posterior process of a manifold in 3D

Figure 4.3: (a) A manifold in 2D with an intrinsic dimensionality of 1. (b) The 1D embedding of the manifold in (a). (c) Fitting a Gaussian posterior process to the first dimension of the manifold with respect to the embedding. The black curve is the mean of the posterior process while the gray shaded area represents $\pm 100\times$ the standard deviation of the posterior process. (d) Fitting a Gaussian posterior process to the second dimension of the manifold with respect to the embedding. (e) The reconstructed 2D manifold (the black curve) from the Gaussian posterior processes.

with an intrinsic dimensionality of 2 is shown. Note that the intrinsic parameterization used to infer the Gaussian posterior process in this case is not the low dimensional embedding, but simply a parameterization that is consistent with the manifold. A smooth surface is inferred from the 2D parameterization to each dimension of the 3D manifold. By sampling more points in the parameterization space, a smooth Gaussian posterior process manifold can be generated.

In appendix A, an application of Gaussian posterior process on drawing manifolds is given. From a few hand-drawn points, a smooth manifold can be automatically drawn by the Gaussian posterior process.

Remarks In [43], it was shown that the generalized radial basis function interpola-

Figure 4.4: (a) A parameterization used to generate the S curve. (b) The S curve. (c) - (e) The Gaussian posterior processes learned for each of the three dimensions of the S curve. (f) Sampling more points in the parameterization space. (g) The reconstructed manifold of the points sampled in the parameterization space.

tion is able to model the mapping between the embedding space and the input space of a manifold. Comparatively, using the Gaussian posterior process to model the mapping has the advantages of efficient model selection (by maximum marginal likelihood) and the capability to output a probabilistic estimate.

4.2.3 GPLVM and manifolds

In chapter 2 it has been shown that GPLVM can be used to find a space-filling curve of a 2D input space by using a latent space dimensionality of one. What happens if the data in the input space indeed has a smaller intrinsic dimensionality? In the previous section it has also been shown that Gaussian posterior process is capable of modeling smooth manifolds if the intrinsic parameterization or low dimensional embedding is known. What if it is not known and is instead obtained from the latent variables found by GPLVM? In figure 4.5 the process of fitting a GPLVM with latent space dimensionality of one to a sine curve in 2D input space is shown. The latent variables are randomly initialized. The plot of each dimension of Y against the latent variables X upon initialization is scattered and fitting a smooth function to it will lead to erroneous reconstruction (figure 4.5 (a)). As the optimization process iterates, the latent variables continuously organize themselves such that the mapping from X to Y becomes smoother.

Another example is shown in figure 4.6. It is observed that in both cases GPLVM models the manifold in $\mathbb{R}^2$ well with a latent space of $\mathbb{R}^1$. However, because GPLVM suffers from local minima problem due to gradient descent, there is no guarantee that the latent variables found correspond to the smoothest fitting of the input data. Comparing figure 4.6 and figure 4.3, it is clear that the embedding of the manifold in figure 4.3 produces a much smoother fitting to the input data, which indicates that the set of latent variables found in figure 4.6 by GPLVM is only a local minimum of the gradient descent. Therefore, GPLVM is not capable of finding a set of latent variables that is consistent with the topology of the manifold as an embedding of the manifold in a lower dimensional space. GPLVM usually only finds a piecewise smooth embedding of the manifold. Nonetheless, this does not stop GPLVM from performing other manifold learning tasks, such as the problem of manifold denoising addressed in this chapter, in which the piecewise smooth embedding is exploited. Moreover, since GPLVM finds a piecewise smooth embedding of the manifold, it means that it could handle multi-modal

manifold with multiple connected components.

Figure 4.5: The process of modeling of the sine curve using GPLVM. In each subplot, the three axes (from top to bottom, left to right) show the plots of $y^{(1)}$ against x , $y^{(2)}$ against x , $y^{(2)}$ against $y^{(1)}$, respectively. The blue curve is the reconstructed data by back-projection from the latent space using the latent variables and hyperparameters learned at the current iteration.

4.3 Manifold denoising using GPLVM

Inspired by the above findings about manifolds and GPLVM modeling of manifolds, it is proposed in this thesis to use GPLVM for manifold denoising.

Figure 4.6: The process of modeling of the spiral curve using GPLVM. In each subplot, the three axes (from top to bottom, left to right) show the plots of $y^{(1)}$ against x , $y^{(2)}$ against x , $y^{(2)}$ against $y^{(1)}$, respectively. The blue curve is the reconstructed data by back-projection from the latent space using the latent variables and hyperparameters learned at the current iteration.

4.3.1 Manifold denoising by back projection

It is proposed to denoise Y by mapping the latent variables X back to the original space. Given the noise model $\mathbf{y}_i = W\mathbf{x}_i + \epsilon$, the denoised data $Z = [\mathbf{z}_1, \dots, \mathbf{z}_n]^T \in \mathbb{R}^{n \times q}$ is obtained by removing the noise ϵ .

$$Z = XW^T. \quad (4.4)$$

In chapter 2, it has been shown that one way to reconstruct the latent variables is to take the mean of the posterior distribution. Here another equivalent way to reconstruct the data in input space from the latent variables based on the maximum a posterior (MAP) estimate of W is derived. Both are shown to be the same in this case.

Referring back to section 2.4, the maximum a posterior (MAP) estimate of W is

derived from the posterior distribution of W

$$\begin{aligned} P(\mathbf{w}_j | \mathbf{y}_{:,j}, X, \beta) \\ = \mathcal{N}(\mathbf{w}_j | \sigma^2 I + X^T X)^{-1} X^T \mathbf{y}_{:,j}, (I + \sigma^{-2} X^T X)^{-1}). \end{aligned} \quad (4.5)$$

Hence, the MAP estimate is simply the mean in the above normal posterior,

$$W_{\text{MAP}}^T = (\sigma^2 I + X^T X)^{-1} X^T Y, \quad (4.6)$$

and the reconstructed data is given by (making use of the Sherman-Morrison formula [108, 18] and identities for inversion sums [18])

$$\begin{aligned} Z &= X W_{\text{MAP}}^T \\ &= X (\sigma^2 I + X^T X)^{-1} X^T Y \\ &= (I - (I + \sigma^{-2} X X^T)^{-1}) Y \\ &= (I - (I + \sigma^{-2} K)^{-1}) Y \\ &= K (K + \sigma^2 I)^{-1} Y \end{aligned} \quad (4.7)$$

A nice property about this reconstruction is that it only depends on K , the covariance matrix. By taking $K = \Phi(X)\Phi(X)^T$ using a valid kernel, the denoising algorithm can model nonlinear mappings from the latent space to the original space. For the purpose of manifold denoising it is proposed to use the SE covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \alpha \exp(-1/2\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. This is based on the earlier findings that the mapping from the low dimensional embedding of the manifold to each dimension of the manifold is a smooth function.

4.4 Comparisons to other manifold denoising methods

In the following, the proposed manifold denoising method is compared with related methods in a unified framework. First of all, the general manifold noise model is described by (adopted from [67])

$$Y = i(M) + \epsilon, \quad (4.8)$$

where $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times q}$ is the set of observed variables, M is the underlying manifold in $\mathbb{R}^{n \times d}$, i is a smooth mapping $i : \mathbb{R}^d \rightarrow \mathbb{R}^q$, and ϵ represents the additive noise in $\mathbb{R}^q$. A natural approach to manifold denoising is to estimate M and i and find the noise-free variables by reconstructing $i(M)$. However, this is not easy to do, since i , ϵ and M are all unknown. Below is a comparison of the proposed method with some recent and state-of-the-art techniques for manifold denoising that make different assumptions in order to approximate i , ϵ and M .

Proposed method: The proposed manifold denoising method has a noise model that is closest to the general manifold noise model in equation (4.8). In the weight space view, the noise model is the GPLVM model $Y = \Phi(X)W + \epsilon$, where $\Phi(X)$ is an implicit mapping into the kernel induced feature space. In the function space view, the noise model is

$$Y = \prod_j^q \mathcal{GP}_j(X) + \epsilon, \quad (4.9)$$

where the smooth mapping $i(M)$ in equation (4.8) is approximated by the product of q independent Gaussian processes, which have been shown to be able to generate arbitrary smooth manifolds in section 4.2.1. ϵ is assumed to be isotropic Gaussian in the high dimensional input space but other noise models are also possible [82]. The optimization of the model is by maximizing the likelihood of $P(Y|X)$. Noise is removed by back projection from the latent space. From the experimental evaluation,

the proposed method is less sensitive to the choice of model parameters, and achieves better denoising results compared with others.

Kernel principal component analysis (KPCA): KPCA [117] has been used for image manifolds denoising [96, 76] and human motion denoising [132]. Denoising by KPCA is a natural nonlinear extension to denoising by principal component analysis (PCA). PCA solves for a lower dimensional representation of the feature space that minimizes the difference between the original and new representations in a least sum-of-squares sense. The noise model is given by

$$Y = XU^T + \epsilon^1, \quad (4.10)$$

where $U \in \mathbb{R}^{q \times d}$ is obtained from the first d eigenvectors with largest eigenvalues of the covariance matrix $\mathbf{C} = \frac{1}{n-1}Y^TY$, $X \in \mathbb{R}^{n \times d}$ is the projected vectors in the subspace given by $X = YU$, and ϵ is the residue between Y and the back projected XU^T which is considered to be the noise. KPCA is to perform PCA in a kernel induced space implicitly using the kernel trick (in a way similar to how the kernel trick was applied to linear regression in chapter 2), which enables nonlinear patterns to be discovered. PCA and KPCA are variance preserving transformations. The components kept by PCA and KPCA projections are those with largest variances. To do denoising, the residue between the original variables and the back projection from the lower dimensional space is considered as the noise, which also corresponds to the components with small variances. We can see that this assumption may not be true in real cases. Noises could also occur in components with large variances and useful components with small variances may be removed, causing distortion to the data. The assumption contradicts the general manifold noise model in equation (4.8), which states that noise is present in every component in the input space. Moreover, unlike PCA, KPCA does not have exact pre-image² mapping from the low dimensional subspace to the original space, and

¹For simplicity of representation, it is assumed that Y has been centered by subtracting off the mean.

²Give a function $y = f(x)$, the image of x is $f(x)$. The pre-image of y is $f^{-1}(y) = \{x | f(x) = y\}$.

approximations need to be made to obtain the reconstructed variables for denoising.

Graph based diffusion: In [67], a manifold denoising algorithm using graph based diffusion is proposed. The denoising process takes effects directly in the original space, without involving any subspace. The input space is considered as a result of a diffusion of the density function P_M of the manifold M .

$$Y = \vartheta(M) + \epsilon \quad (4.11)$$

where $\vartheta(M)$ represents a diffusion process on $\mathbb{R}^q$. The principle of denoising is to reverse the diffusion process. An iterative algorithm is proposed to reverse the diffusion process. A Gaussian noise is assumed $\epsilon \sim N(0, \sigma^2)$, but it is claimed that other reasonably concentrated isotropic noise should also work. Since this method does not involve any subspace embedding, there is one less parameter (subspace dimensionality) to consider. However, the iterative procedure turns out to be not optimal on convergence. The time to stop the iterations becomes crucial to the denoising result of this method.

Local smoothing: While KPCA and PCA project the data globally into a lower dimensional space, a local weighted linear smoothing technique for outlier handling and noise reduction is proposed in [105]. The manifold is locally projected to an affine subspace using weighted principal component analysis (WPCA), while the local neighborhood is defined using k nearest neighbors. Locally, the noise model is similar to that of PCA

$$Y = XU_w^T + \epsilon, \quad (4.12)$$

except that U_w is found in a weighted manner. The weights are determined so as to weigh down possible outliers. A bias correction method is used to prevent the manifold from shrinking. The local WPCA projection and bias correction are used iteratively until convergence. A similar approach is proposed in [152], where a local surface estimator is used to in the place of WPCA. From the experimental results,

In the context of subspace models, the pre-image mapping refers to the mapping from the subspace to the input space [2].

the WPCA approach to denoising also suffers from distortion to the manifold upon convergence of the algorithm, despite the use of bias correction.

Robust methods for manifold learning: There are robust manifold learning methods proposed to enhance the capabilities of the respective manifold learning algorithms in the presence of noise and outliers. In [21], a preprocessing procedure is proposed for manifold learning algorithms such as LLE and ISOMAP. The purpose is to make the low dimensional embedding more robust to noise and outliers. The proposed procedure consists of outlier detection by histogram analysis and local smoothing using the error-in-variables (EIV) model. In [25], in order to enhance the topological stability of ISOMAP, critical outliers are first detected using network flow. This step is integrated with the computation of geodesic distances in the ISOMAP algorithm. These robust methods are only designed to enhance the manifold embedding algorithms in the presence of noise. They usually do not have a reconstruction method to project the embedding back to the feature space. Therefore, they are not suitable for general purpose manifold denoising.

4.5 Experimental evaluation

The proposed method is compared with three other representative manifold denoising methods, namely KPCA [76], graph based diffusion [67] and local smoothing by WPCA [105].

Besides KPCA, all other methods include the proposed are solved in an iterative manner. The convergence criterion for the proposed method using GPLVM is that the change in the objective function (marginal likelihood) and the change in the variables (latent variables and hyperparameters) is less than a threshold THRES. For graph based diffusion, the convergence criterion is that the relative change after one iteration in the feature space is less than THRES. For local WPCA, the convergence criterion is also the relative change in the feature space after one round of local projection and

back projection being less than THRES. THRES is set to $1e - 6$ in all experiments. The maximum number of iterations is set to 2000 for all methods. All the reported experiments converge before reaching the limit.

4.5.1 Toy data

The first set of experiments are conducted on a toy data set similar to that used in [67]. The manifold M is given as $t \rightarrow [\sin(2\pi t), 2\pi t]$, where t is sampled uniformly on $[0, 1]$. Isotropic Gaussian noise with a standard deviation $\sigma = 0.2$ is added to each data point. The intrinsic dimensionality of the manifold is 1. For fair comparison, it is assumed that this intrinsic dimensionality is available to all methods as the prior knowledge.

Figure 4.7 plots the data before and after denoising by the four methods. Figure 4.7 (a) shows the original data corrupted by noise and the underlying sine curve without noise. Figure 4.7 (b) shows the denoising effect by the proposed method using a latent space of $d = 1$ upon convergence of the algorithm. It recovers a sine curve without much distortion. Figure 4.7 (c) shows denoising by KPCA by keeping the first principal component. The reconstruction is done by Kwok & Tsang's [76] method. We can see that the distortion to the manifold is severe. This is due to the fact that the reconstruction method needs to construct a local neighborhood to determine the pre-image in the input space, but in a subspace of 1D, the algorithm could not construct meaningful local neighborhoods. In figure 4.7 (d) the denoising result using graph based diffusion with a neighborhood size $k = 30$ upon convergence is shown. It can be seen that the manifold has been distorted with a fragmentation effect. For denoising by local WPCA, the subspace dimensionality used in the WPCA projection is set to 1. The local neighborhood size k is set to 30. The result on convergence (subplot (e)) also shows some distortion along the manifold.

In the next experiment, the manifold M is further embedded into $\mathbb{R}^{200}$ by placing it into the first two dimensions of $\mathbb{R}^{200}$. The values on the other 198 dimensions are set to,

Figure 4.7: (a) The noisy data and the underlying sine curve. (b) GPLVM with $d = 1$ upon convergence. (c) KPCA with $d = 1$. (d) Graph based diffusion with $k = 30$ upon convergence. (e) Local smoothing with $d = 1$, $k = 30$ upon convergence.

without loss of generality, a constant value of zero. Full isotropic Gaussian noise in $\mathbb{R}^{200}$ with a standard deviation $\sigma = 0.2$ is added to each data point. This helps to evaluate the robustness of the manifold denoising algorithms when the manifold is embedded in a much higher dimensional space and is corrupted by noise in the high dimensional space. The results are shown in figure 4.8. The first two dimensions are plotted while the noise on the other 198 dimensions are indicated by the numbers shown in brackets below the respective plots. The measure is normalized noise in percentage defined by $\|Z(:, 3 : 200)\|_F / \|Y(:, 3 : 200)\|_F \times 100\%$, where Z is the reconstructed data matrix, Y is the noisy data matrix, and $\|\cdot\|_F$ is the Frobenius norm. Again, the proposed method gives better denoising results in terms of both distortion to the manifold and noise removal on the irrelevant dimensions.

In figure 4.9, a detailed analysis of each manifold denoising algorithm is given. For denoising by GPLVM, the reconstructions at different iterations of the optimization process are shown. Since a latent space of 1 dimension is used, the reconstruction is always a curve that has an intrinsic dimensionality of one. It iteratively converges to align itself with the manifold to maximize the likelihood of fitting smooth functions to both dimensions of the manifold. For KPCA, since the reconstruction is based on matching local neighborhoods in the subspace and the input space, preserving more principal components leads to better reconstruction, but the capability of noise re-

Figure 4.8: (a) First 2 dimensions of the noisy data in $\mathbb{R}^{200}$. (b) Denoising by GPLVM. (c) Denoising by KPCA. (d) Denoising by Graph based diffusion. (e) Denoising by local smoothing. The numbers shown in brackets are the normalized noise on the other 198 dimensions.

moval drops. Manifold denoising by graph based diffusion works directly in the input space, by reversing the assumed diffusion process. Upon convergence, the reconstructed manifold is more distorted than intermediate results. This has been mentioned in the original paper [67]. To prevent severer distortion, the number of iterations needs to be considered as a parameter of the algorithm. For local smoothing by WPCA with bias correction, the reconstructed manifold has more distortion towards both ends of sine curve, although the bias correction has been used to prevent over-shrinking of the manifold. In the original paper of [105], the number of iterations is heuristically set to 10.

In conclusion, the proposed manifold denoising method based on GPLVM preserves the manifold structure while removing noise on the manifold as well as on the irrelevant dimensions when it is embedded in a high dimensional space. Moreover, the denoising performance is optimal on the convergence of the the algorithm.

4.5.2 Swiss roll

The performance of the proposed manifold denoising method is also illustrated on the typical Swiss roll data commonly used in manifold learning. The ISOMAP [133]

Figure 4.9: A detailed analysis of the denoising process of the manifold denoising algorithms.

algorithm is applied to the Swiss roll data to unfold the manifold. The results obtained on the original noisy data and the denoised data by various manifold denoising methods are shown in figure 4.10. A neighborhood size of $k = 7$ is used in the construction of neighborhood graph in ISOMAP. It is observed that the embedding with the proposed method is closest to the ground truth. The manifold topology is severely distorted by the KPCA method. The ‘Swiss roll’ is not completely unfolded after denoising by graph based diffusion and local smoothing by WPCA, which also happens when ISOMAP is directly applied on the noisy data. This is due to the presence of noise that causes ‘short circuits’ in the construction of neighborhood graph.

4.5.3 Image manifold denoising

An image manifold is a collection of related images [98], such as a set of handwritten digit images. The set of images can be modeled by a manifold embedded in an $M \times N$

Figure 4.10: ISOMAP embedding of the Swiss roll data before denoising and after denoising by different methods.

dimensional space, where the images have a size of $M \times N$ pixels, and gray scale at each pixel is used as the feature to represent the images. Image manifolds have been used extensively to evaluate manifold learning algorithms [133, 113, 9, 38, 146].

The experiments are carried out on the USPS handwritten digit data set. Handwritten digit data sets have been used for manifold learning and denoising in [133, 96, 67, 76]. The image manifold formed by images of different digits and different writing styles makes it a good choice for study because of the high-dimensional, multiple connected component manifold structure. Using it for denoising has the additional advantage of easy visualization of the denoising effects. Note that in the experiment, images of all digits are used together, instead of treating each digit class as a separate data set. This will make the feature space consisting of multiple connected components. Handling multiple connected components simultaneously is important as it allows direct

clustering or classification on the whole data set.

4.5.3.1 Experimental setting

A set of 500 images, 50 from each of the 10 digit classes is used. Each sample is represented by its gray scale image of size 16×16 with gray values mapped to $[-1, +1]$ (so dimensionality of feature space = 256). In the first experiment, full isotropic Gaussian noise of a standard deviation of 0.4 is added to each sample. The noise-free images (as ground truth for evaluation) and the noisy images are shown in figure 4.12 (a) and (b). It is difficult to know the intrinsic dimensionality of the manifold beforehand. The estimator in [66] gives the estimated intrinsic dimensionality of the noise-free data to be 3, while the estimated intrinsic dimensionality of the noisy data is 9. Since three of the methods studied here depend on a subspace dimensionality d that may be correlated to the intrinsic dimensionality, one of the objectives here is to study how sensitive these methods are with respect to the choice of d .

Other parameters are also considered. For the proposed method, the hyper-parameters α, γ, σ of the GPLVM model can all be optimized together with the latent variables X by maximum likelihood. The free parameter is the dimensionality of the latent space. The effects of the number of iterations in the optimization process is also studied. For KPCA, the algorithm in [76] is used, which has been verified to be a much better approximation method for KPCA back-projection than the algorithms in [96]. The free parameters include the Gaussian width in the Gaussian radial basis function (GRBF) kernel and the number of principal components to keep (the subspace dimensionality). Since there is no principled way to select the width, it is set empirically to the mean distance between the data points, to indicate the local neighborhood size. For both graph based diffusion and local smoothing, it is found that the number of iterations needs to be considered as a free parameter because the denoising effects may deteriorate as the iterative algorithms converge. For both methods a local neighborhood size is also needed. The local neighborhood is defined by k nearest neighbors and k

is set heuristically. For denoising by local smoothing, it also depends on the number of principal components to keep (local subspace dimensionality). A summary of the dependency of the methods on parameters is given in table 4.1.

Table 4.1: Dependent parameters of different methods: d - the subspace (or latent space) dimensionality; k/σ - the local neighborhood size in terms of k nearest neighbors or Gaussian width σ ; $iter$ - the number of iterations for methods that are solved iteratively. ‘yes’ - the effect of the parameter is studied; ‘p.d.’ - the parameter is pre-determined heuristically; ‘-’ - the parameter does not apply to the method.

	d	k/σ	$iter$
GPLVM	yes	-	yes
KPCA	yes	p.d.	-
Graph	-	p.d.	yes
WPCA	yes	p.d.	yes

4.5.3.2 Experimental results

In the first experiment, full isotropic Gaussian noise of a standard deviation of 0.4 is added to each sample. Manifold denoising is applied using the four methods under study with different combinations of parameters. The quality of denoising is measured by the average signal to noise ratio (SNR) defined as $-10 \log_{10}(\|(Z - Y_o)\|_F^2/nq)$, where $Y_o \in \mathbb{R}^{n \times q}$ is the matrix representing noise-free images, $Z \in \mathbb{R}^{n \times q}$ is the reconstructed matrix after denoising, $\|(Z - Y_o)\|_F$ is the Frobenius norm of the difference matrix, and nq is equal to the number of elements of the matrices. The SNR curves are plotted in figure 4.11. All the plots have the same y-axis range of SNR values for ease of comparison. In (a), the proposed method is evaluated at latent dimensionality of 2, 5, 10 and 20, and also at both convergence and a maximum number of 100 iterations. It is observed that the curves have a flat bell shape. With a latent dimensionality of 2, the SNR is satisfactory (as shown by the reconstructed images in figure 4.12 (c)). It further improves with ‘larger’ latent spaces at $d = 5$ and $d = 10$. As the latent dimensionality goes up further, the performance drops. Recall that the latent variables are solved via optimization, and as a result, the number of variables to optimize increases linearly

with d . Although it is difficult to estimate the optimal latent space dimensionality, it is observed that the proposed method is not very sensitive to it. It can be concluded that a latent space dimensionality of 5 to 10 offers good denoising quality, while a latent space dimensionality of 2 or 3 is ideal for visualizing the latent space. The SNRs by fixing a maximum number of 100 iterations are slightly worse than that on the convergence of the algorithm. This shows that the proposed method can be stopped at an earlier stage before convergence to save computation.

The above findings are also used to verify the GPLVM model selection algorithm proposed in section 2.4.2. The noise levels (standard deviation of the white noise) estimated by maximum marginal likelihood at $d = 2, 5, 10, 20$ are 0.58, 0.51, 0.45, 0.33, respectively. The estimated noise level reduces with increasing d and it is closest to the true noise level of 0.4 at $d = 10$. Therefore, if there is some prior knowledge about the noise level in the data, it can be used to select the latent space dimensionality.

Figure 4.11 (b) shows the performance of KPCA on the USPS handwritten digit data set at $d = 2, 10, 50, 100, 200, 500$. GRBF kernel is used and the Gaussian width is set to the mean of pairwise distances between the points. As being observed in the toy data in the earlier section, KPCA does not perform well with a small number of principal components. At $d = 2$, the reconstructed images (in figure 4.12 (g)) do not resemble the original digits at all. We can see that the optimal d (50 in this case) is far away from the estimated intrinsic dimensionality of 9. As more principal components are used, its performance goes down. This is due to the fact that when too many components are used, large portions of noise may be included in the reconstructed images as well.

Figure 4.11 (c) shows the performance of graph based diffusion. It requires a local neighborhood to be defined to construct the graph, which is fixed at $k = 30$ nearest neighbors. This is the only method compared here that performs denoising directly in the feature space and hence there is no subspace dimensionality involved. However, it is observed that this method is very sensitive to the number of iterations. For the

Figure 4.11: (a) Denoising by the proposed method. (b) Denoising by KPCA. (c) Denoising by Graph based diffusion. (d) Denoising by local smoothing using WPCA. For details refer to text.

USPS data set, its denoising performance in terms of SNR is best at iterations 5 and 10, while it continuously deteriorates with more iterations till convergence. As mentioned in [67], this method moves data points towards high-density regions. This could cause the denoised data to lie on a few cluster prototypes. Due to this, convergence should not be the criterion to stop the algorithm for denoising purpose and the number of iterations has to be treated as a free parameter.

Figure 4.11 (d) shows the performance of local WPCA. This method is dependent on both the subspace dimensionality d and the number of iterations. It also depends on a local neighborhood size k , which is fixed at 30. It is observed that the performance of

the method on the USPS data set deteriorates after the first iteration. In other words, it is better to use this method ‘one-shot’ rather than use it iteratively. Moreover, it is also sensitive to the number of principal components used in the WPCA projection. This is because in WPCA, noise removal is by removing a number of principal components with least variances. The results show that the SNR is much better at $d = 10$ than those at $d = 2$ and $d = 20$, and this is when d is close to the estimated intrinsic dimensionality of 9.

Overall, the proposed method achieves the best SNRs among the four methods evaluated with different combinations of parameters, and it is also relatively less sensitive to the choice of parameters. In figure 4.12 the reconstructed images at selected settings of parameters are shown for all four methods. The number shown below the plot is the SNR. Figure 4.12 (c)-(f) show denoising by the proposed method upon convergence with $d = 2, 5, 10, 20$. Figure 4.12 (g) - (j) show denoising by KPCA at $d = 2, 10, 50, 200$. Figure 4.12 (k) - (n) show denoising by graph based diffusion at iterations 2, 10, 20 and 72 at which it converges. Figure 4.12 (o) to (r) show denoising by local WPCA at $d = 10$ and iterations 1, 5, 10 and 456 at which the algorithm converges.

In the next experiment the Gaussian noise level in the images is increased to a standard deviation of 0.8. In figure 4.13 the reconstructed images using the four methods are shown. For each method the parameter settings that has been observed to be the best in the previous experiment are used. Again, the proposed method achieves best results in terms of both SNR and visual quality of the reconstructed images.

All the four methods under study have assumptions about the noise present in the manifold. The GPLVM uses a Gaussian noise model. KPCA has the assumption that the noise lie on the principal components of the kernel space with least variances. Graph based diffusion also assumes a Gaussian noise model. Local WPCA has the assumption that the noise can be removed by removing the principal components with least variances in the local WPCA model. Due to these assumptions, it is interesting to find out how these methods compare with each other with noise other than Gaussian.

Figure 4.12: Denoising by various methods at different parameter settings. The numbers shown below the plots are the corresponding SNRs. First row: original and noisy images (Gaussian $\sigma = 0.4$). Second row: GPLVM at $d = 2, 5, 10, 20$. Third row: KPCA at $d = 2, 10, 50, 200$. Fourth row: graph based diffusion at iteration = 3, 10, 20, 72. Last row: Local WPCA at iteration = 1, 5, 10, 456.

Figure 4.13: First row: (a) original and (b) noisy images (Gaussian $\sigma = 0.8$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.

Experiments with salt and pepper noise in images are carried out. Salt and pepper noise are randomly occurring white and black pixels across an image. Two noise levels of $p = 0.15$ and 0.35 are examined. A noise level of p means $100p\%$ of pixels are affected by noise. The reconstructed images are shown in figures 4.14 and 4.15. Again, in terms of SNRs of the reconstructed images, the proposed method still outperforms the others, especially when noise level is high.

4.5.4 Missing value recovery by manifold denoising

As a probabilistic model, the proposed method naturally handles missing values in the data. The detailed derivations of the model when there are missing values in the data is given in appendix B. In the following experiment it is assumed that a batch of noisy digit images are received with missing values. The proposed method can be used to reduce the noise as well as to recover the missing values. Again the experiment is conducted on the USPS digit data set with a Gaussian noise of $\sigma = 0.4$, and with the amount of missing values ranging from 20% to 80%. Satisfactory recovery is obtained,

Figure 4.14: First row: (a) original and (b) noisy images (salt and pepper $p = 0.15$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.

Figure 4.15: First row: (a) original and (b) noisy images (salt and pepper $p = 0.35$). Second row: (c) GPLVM at $d = 10$, (d) KPCA at $d = 50$, (e) Graph based diffusion at iteration 10, (f) local WPCA at iteration 1.

even at high missing rates of 60% or 80%. Results are shown in figure 4.16.

Figure 4.16: Top row: noisy images ($\sigma = 0.4$) with missing data. The red pixels indicate missing values. Bottom row: recovered images. Four columns refer to missing amount of 20%, 40%, 60% and 80%, respectively.

4.5.5 Manifold denoising and distance based classification

In the previous section the manifold denoising effect on the USPS handwritten digit images has been examined both visually from the denoised images and quantitatively based on SNR. In this section the objective is to examine how manifold denoising helps improve classification. It is well-known that distance based classifiers are sensitive to noise and outliers in the data. The nearest neighbor classifier is used on the USPS data set as well as the benchmark data sets from [20] to show the advantage of manifold denoising as a pre-processing step before classification.

There are two settings of the experiments. The first one is with 10 labeled samples and the second one is using 100 labeled samples. The labeled samples are randomly chosen and the process is repeated 12 times to create 12 training/test sets for each case. There are no bias in the selection process except that at least one training sample has to be selected from each class. The experiments are conducted in a semi-supervised setting. Both labeled and unlabeled samples are used for unsupervised

denoising. Classification of the unlabeled samples is performed on the denoised data using a nearest neighbor classifier. For the USPS data set, the parameters for various methods are set according to the empirical evaluation in the previous section. The average classification errors and the standard deviations are shown in table 4.2. It is observed that all classifications improve after performing manifold denoising using the proposed method compared with that on the noisy data. Graph based diffusion also gives good classification results among the four methods; the proposed method outperforms graph based diffusion when the noise level is relatively high.

Table 4.2: Classification errors before and after manifold denoising by various methods. The four data sets used are the same as in the previous section, namely, USPS hand-written digits with Gaussian noise of standard deviations of 0.4 and 0.8 ($\mathcal{N} \sim 0.4$ and $\mathcal{N} \sim 0.8$), and with salt-and-pepper noise of noise levels of 0.15 and 0.35 (SnP0.15 and SnP0.35). For each data set, two labeled sample sizes are evaluated (10 and 100). The mean and standard deviations of classification errors over 12 random splits of training and test set are shown. n_{train} : number of training samples. The lowest classification errors are highlighted in bold typeface.

data set	n_{train}	Noisy	GPLVM	KPCA	Graph	WPCA
$\mathcal{N} \sim 0.4$	10	53.8 $\pm$ 5.3	49.3 $\pm$ 6.0	51.45 $\pm$ 5.63	47.59$\pm$6.99	54.97 $\pm$ 6.53
	100	31.1 $\pm$ 3.0	27.0 $\pm$ 2.4	27.88 $\pm$ 2.77	25.42$\pm$2.3	31.06 $\pm$ 2.62
$\mathcal{N} \sim 0.8$	10	57.6 $\pm$ 5.1	47.6$\pm$6.7	52.26 $\pm$ 5.63	49.37 $\pm$ 6.32	58.83 $\pm$ 3.62
	100	39.3 $\pm$ 2.9	31.3$\pm$1.8	34.29 $\pm$ 2.25	32.71 $\pm$ 2.19	40.27 $\pm$ 3.35
SnP0.15	10	62.6 $\pm$ 5.0	55.2 $\pm$ 5.2	56.36 $\pm$ 5.12	53.15$\pm$5.07	61.68 $\pm$ 5.18
	100	48.6 $\pm$ 2.5	39.0 $\pm$ 3.1	38.0$\pm$3.52	38.67 $\pm$ 2.42	43.96 $\pm$ 3.32
SnP0.35	10	73.6 $\pm$ 3.4	64.9$\pm$5.6	67.72 $\pm$ 4.8	67.09 $\pm$ 4.87	78.55 $\pm$ 6.14
	100	65.5 $\pm$ 2.3	55.5$\pm$2.4	56.83 $\pm$ 2.86	58.5 $\pm$ 1.99	68.67 $\pm$ 2.35

On the benchmark data from [20], the latent space dimensionality in GPLVM is set to 5 and the maximum number of iterations in GPLVM maximum marginal likelihood optimization is fixed to 100 to save computation. The 7 data sets used for evaluation consists of three artificially created data sets (g241c, g241d, Digit1) and four data sets derived from real data (USPS, Coil, BCI, Text). These data are embedded in high dimensional space with noise. Results are shown in table 4.3. It is observed that for

data sets g241c, g241d and text, significantly better performance can be obtained by using GPLVM based manifold denoising as a preprocessing step, whereas the results are roughly indifferent on the other data sets. This is consistent with the findings in [67].

Table 4.3: Classification errors before and after manifold denoising. For each data set, two labeled sample sizes are evaluated (10 and 100). The mean and standard deviations of classification errors over 12 random splits of training and test set are shown. n : total number of samples; q : input space dimensionality; n_{train} : number of training samples.

data set	n	q	n_{train}	MD	Noisy
g241c	1500	241	10	32.91 $\pm$ 6.92	44.05 $\pm$ 3.30
			100	26.58 $\pm$ 1.57	40.28 $\pm$ 2.25
g241d	1500	241	10	27.77 $\pm$ 10.77	43.2 $\pm$ 2.9
			100	13.04 $\pm$ 1.87	37.49 $\pm$ 1.61
Digit1	1500	241	10	22.52 $\pm$ 5.64	23.47 $\pm$ 5.46
			100	5.34 $\pm$ 1.51	6.12 $\pm$ 1.32
USPS	1500	241	10	20.16 $\pm$ 4.53	19.82 $\pm$ 3.96
			100	7.15 $\pm$ 1.07	7.64 $\pm$ 0.86
Coil	1500	241	10	65.90 $\pm$ 4.22	65.91 $\pm$ 4.14
			100	22.78 $\pm$ 1.92	23.27 $\pm$ 2.03
BCI	1500	117	10	48.33 $\pm$ 2.92	48.74 $\pm$ 2.83
			100	47.53 $\pm$ 2.38	44.83 $\pm$ 1.77
Text	1500	11960	10	36.47 $\pm$ 11.59	39.28 $\pm$ 5.1
			100	27.98 $\pm$ 1.97	30.75 $\pm$ 1.08

4.6 Concluding remarks

A method for manifold denoising based on Gaussian process latent variable models is proposed. Noise is removed by inferring Gaussian processes on each dimension of the manifold and back-projecting the latent variables to the feature space. Through the use of a squared exponential (SE) covariance function in GPLVM, noise on the manifold is smoothed and reduced. The proposed method is compared with kernel principal component analysis, graph based diffusion and a local smoothing method based on

weighted principal component analysis. Experimental results on various data sets indicate that the proposed method is less sensitive to model parameters, achieves better denoising performance in terms of SNR, and improves distance based classification. A detailed analysis on image manifold denoising on the USPS handwritten digit data set is also conducted.

Chapter 5

Shape Matching by Region Grouping

This chapter proposes a novel region grouping approach to shape matching. It is proposed as an alternative region based approach to the traditional edge based shape matching. It has the advantage of obtaining a higher detection rate and obtaining meaningful object segmentation simultaneously.

5.1 Introduction

Shape is a rich representation for many objects. For example, figure 5.1 shows humans in various postures. By just looking at the binary silhouette images in figure 5.1 (b), it is possible to ascertain with some confidence that they are the shapes of human subjects. In the proposed framework, human and other objects are identified by their shape. A set of training shape exemplars is acquired. When a new instance is close enough to a training shape, a match is found.

Although exemplar based shape matching approaches for object detection and recognition are usually computationally intensive, they do not disappear from recent

Figure 5.1: (a) Images of humans in various postures. (b) Binary silhouette images showing the shapes of humans in the corresponding images in (a). (c) Edge images of the corresponding images in (b). (d) DT images of the corresponding images in (c).

literature [51, 52, 135]. The advantage of exemplar based approaches is that they make much weaker assumptions. All that is needed is a training set that ideally covers as many as possible instances of the target object class. No feature correspondence or

construction of a parameterized shape model is needed. Only a similarity measure between shape exemplars is needed. In addition, various methods have been proposed to alleviate heavy computation, such as the use of a hierarchical shape representation and a coarse-to-fine matching of exemplars [51, 52, 1, 84].

A crucial problem in matching is the definition of similarity. Image edges have been popularly used for shape matching due to the efficiency of computing pairwise similarities using distance transforms [70, 52, 50, 51]. Examples of such similarity measures include Chamfer distance and Hausdorff distance. The deficiency commonly observed is that there are usually many false positives when they are applied in shape based object detection [51]. In images which are relatively textured, the edge based similarity measure is not robust. Moreover, the segmentation of the matched object is not readily available. The contour of the closest exemplar is used as the contour of the matched object. In this chapter, a novel region grouping approach for exemplar based shape matching is proposed. Instead of edge detection, the image is first segmented into regions. The combinations of regions that best match the exemplars are determined based on a novel probabilistic similarity measure. It is shown that under a direct comparison with the edge based approaches, the proposed method is more robust in terms of discriminating objects and non-objects. Moreover, segmentation of the object is obtained simultaneously.

The shape exemplars are represented by the distance transformed (DT) images and modeled by a Gaussian process latent variable model (GPLVM). In this way, the probability of an image region residing inside the object contour can be efficiently defined. The grouping of image regions that best conforms to a valid object class is then decided. Moreover, the similarity between the obtained grouping and the training exemplars can also be efficiently computed as a probabilistic measure.

5.2 Related work

The related literature to this work includes object detection, object segmentation, and shape matching. Object detection refers to those algorithms that only detect the (approximate) locations of a target object. Usually a bounding box around the object is the output. Advances have been made in recent years, particularly in human and face detection [156, 50, 34, 142, 107, 106]. The advantages are that these algorithms are usually fast (required for real-time applications), and have reached a satisfactory performance in terms of detection accuracy (e.g., face detection softwares have been widely used in commercial cameras). A multiple object class detection method is proposed in [97]. Object segmentation or figure-ground segregation, on the other hand, is a more difficult problem. It requires the segregation of the objects from the image in terms of its boundary or a segmentation mask. In recent years, object segmentation is usually done by a bottom-up or combination of top-down and bottom-up approach [109, 137, 138, 154, 155, 53, 116]. In these approaches, statistical analysis and reasoning are usually applied to a set of training samples to obtain global or local detectors. Another approach is to incorporate high level prior knowledge in variational methods for image segmentation such as the level set method [30, 29].

Parts detection is commonly used in object segmentation [137, 154]. The problem with these approaches is that a collection of best parts may not form a meaningful object. For example, in an image, there may be image patches that fit (are visually similar to) the head, the arms, and the legs, but their combination do not reconstruct the human body. Top-down approach can be used as a remedy to help identify whether the parts form a meaningful object. For example, in [154], parts of the human body are first detected and then cues from patches and edges are combined to segment out the final meaningful object. It is similar to the proposed approach in that it also groups smaller parts to form the object. The proposed method differs from [154] in that all combinatorial groupings of image regions are considered implicitly to obtain the globally optimal foreground figure conforming to the target object classes. Obtaining

an initial segmentation of coherent regions is far less demanding than accurate parts detection. In [45], a shape guided image segmentation and classification framework is proposed, where the classification subsystem acts as the high level knowledge to guide segmentation.

Top-down approaches look for predefined target objects in the image. Research works in this area are mostly model-based. Purely top-down object segmentation has been less popular because it suffers from high computational cost when handling large and complex groupings of image regions [109]. In [109], an object detection method by top-down perceptual grouping is proposed. The method makes use of a set of global criteria obtained from subjective human observations. The criteria are based on some hypotheses such as: the objects are multi-part and low-textured, and the background needs to be relatively uncluttered. As mentioned in their paper, the extraction of a perceptually significant grouping is a difficult problem, but the perceptual evaluation of this grouping is even more challenging as it entails the visual assessment of its proper shape.

The proposed method is more related to [15] and [109]. In [15], the authors also mainly make use of shape to characterize objects. The main difference is that the proposed method uses the shape of the whole object to determine the overall similarity, while the method in [15] uses shape locally for fragments detection. Compared with [109], the probabilistic shape matching criterion in the proposed method is automatically acquired through a set of training exemplars, while the criteria in [109] are obtained subjectively. The difficulty in perceptual evaluation is also addressed as the proposed solution is based on evaluations on all groupings in an image, and global optimality is sought, while the computation is yet efficient.

For edge based shape matching, Chamfer distance [16, 153, 134, 73] and Hausdorff distance [70, 103, 131] are two typically used criteria to compare images. The advantages lie in efficient computations made by the use of distance transforms (DT). In [50], a real-time object detection method for smart vehicles is proposed. The matching

method used is exemplar based shape matching using Chamfer distance. Real-time efficiency is obtained by using a template hierarchy and a coarse-to-fine searching over the shape hierarchy. In [51], the idea is further improved by using a Bayesian model to estimate a matching criterion. In [52], shape matching by distance transforms forms part of a multi-cue pedestrian detection and tracking system.

Besides direct matching with each exemplar, another approach is to acquire a shape model from a set of training shapes [1, 3, 84, 11, 12, 49]. This expands to a much wider research area, including shape registration, shape deformation models, and various feature extraction methods for shape description. The purpose is to find a compact and parameterized shape representation. The computational workload in the online matching phase could be reduced due to the compact representation. However, such modeling approach may not work well for non-rigid object classes that have a lot of intra-class variations, such as human objects.

5.3 A generative model for shape exemplars

In this section the distance transformed representation of shape is first introduced. A GPLVM model is then trained on the training shape exemplars to obtain a generative model for further shape matching.

5.3.1 Representing shapes using DT images

The distance transformed (DT) image is computed as the Euclidean distance transform of the binary edge image. For each pixel in the edge image, the distance transform assigns a number that is the distance between that pixel and the nearest edge pixel. True Euclidean distance is computed using the linear-time algorithm as described in [17]. Moreover, distance from pixels inside the object boundary is set to be negative while the distance from pixels outside the object boundary is positive. In this way, it is possible to identify a pixel as a foreground or background pixel by its polarity. This

is also known as the signed distance function in the level set methods [104]. Figure 5.2 illustrates the computation of DT image from the edge image.

Figure 5.2: (a) The object edge image. The pixel at (28,2) has a nearest edge pixel at (33,14). Therefore, its distance transform is $+\sqrt{(33-28)^2 + (14-2)^2} = 13$ (+ for pixels outside object boundary). (b) The DT image. The pixel at (28,2) has a value of 13.

An alternative representation of shape is to directly use the binary silhouette images which is also a possible choice for the shape matching framework proposed in this chapter. The reason for representing the training images as signed DT images instead of the binary silhouette images lies in two folds. Firstly, DT images are continuous while the binary images are discrete. DT images provides a more detailed information about how close a pixel is to the object boundary. The probability of a pixel being a foreground pixel or a background pixel can be computed by $\Pr(dt < 0)$ and $\Pr(dt > 0)$, which is a useful representation in the proposed method. Secondly, for GPLVM modeling, it is much easier to handle continuous features. For discrete features, additional modelings are needed and exact inference is no longer possible. Figure 5.3 shows the discrete versus continuous representation of the same object shape.

Figure 5.3: (a) 3D view of the binary silhouette image. (b) 3D view of the DT image.

5.3.2 Applying GPLVM on the set of shape exemplars

The object shape is represented by the DT images described in the previous section. Each pixel in the DT image contributes to one dimension in the feature space, so the feature space dimensionality is equal to the number of pixels. A set of n such DT training images is denoted as $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times q}$, where q is the feature space dimensionality.

Apply the GPLVM model with the squared exponential covariance function on the set of DT images to obtain the latent variables $X \in \mathbb{R}^{n \times d}$ and the hyperparameters α, γ and σ . The choice of the SE covariance function follows the findings in chapter 4. It is reasonable to assume that the set of exemplar shapes of one or multiple object classes form manifold structure in the high dimensional input space. The latent space dimensionality d is 2, which is determined by the proposed GPLVM model selection algorithm in section 2.4.2. More details on selecting the latent space dimensionality will be given in section 5.6.1.2.

5.3.3 A generative model for shape matching

The GPLVM model trained on the DT images is a generative model which will be used for shape matching in later sections. Each exemplar in the training set generates new DT images according to a Gaussian distribution. Equation (2.29) is rewritten as:

$$\begin{aligned}
 & p(\mathbf{m}|\mathbf{x}_i, \mathbf{X}, \mathbf{Y}) \\
 &= \prod_{j=1}^q p(m_j|\mathbf{x}_i, \mathbf{X}, \mathbf{Y}) \\
 &= \prod_{j=1}^q \mathcal{N}(m_j|\mathbf{y}_{:,j}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}, k_{i,i} - \mathbf{k}_{:,i}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}) \\
 &= \mathcal{N}(\mathbf{m}|Y^T (K + \sigma^2 I)^{-1} \mathbf{k}_{:,i}, (k_{i,i} - \mathbf{k}_{:,i}^T (K + \sigma^2 I)^{-1} \mathbf{k}_{:,i}) I),
 \end{aligned} \tag{5.1}$$

where $\mathbf{m}$ represents DT images generated by the training exemplar $\mathbf{x}_i$ and m_j is the j th pixel, and $k_{i,i} = k(\mathbf{x}_i, \mathbf{x}_i)$. Note that the probability of each dimension $p(m_j|\mathbf{x}_i, \mathbf{X}, \mathbf{Y})$ can be computed independently. The distribution of new DT images generated by a particular training sample $\mathbf{x}_i$ is characterized by a mean image $\boldsymbol{\mu}_i$ and a corresponding variance φ_i .

$$\begin{aligned}
 \boldsymbol{\mu}_i &= Y^T (K + \sigma^2 I)^{-1} \mathbf{k}_{:,i}, \\
 \varphi_i &= k_{i,i} - \mathbf{k}_{:,i}^T (K + \sigma^2 I)^{-1} \mathbf{k}_{:,i}.
 \end{aligned} \tag{5.2}$$

5.3.4 The proposed similarity measure for shape matching

Finally, the similarity measure between a new shape and the training exemplars is defined. A new shape $\mathcal{S}$ is represented as a partition of the image pixels $j = 1, \dots, q$ into foreground $\mathcal{F}$ and background $\mathcal{B}$.

$$\Pr(\mathcal{S}|\mathbf{x}_i) = \prod_{j \in \mathcal{F}} \Pr(m_j < 0|\mathbf{x}_i, X, Y) \prod_{j \in \mathcal{B}} \Pr(m_j > 0|\mathbf{x}_i, X, Y), \tag{5.3}$$

and

$$\begin{aligned}
\Pr(j \in \mathcal{F}|\mathbf{x}_i) &= \Pr(m_j < 0|\mathbf{x}_i, X, Y) \\
&= \int_{-\inf}^0 p(m_j|\mathbf{x}_i, X, Y) dm_j \\
&= \int_{-\inf}^0 \mathcal{N}(m_j|\mathbf{y}_{:,j}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}, k_{i,i} - \mathbf{k}_{:,i}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}) \\
&= \Psi\left(\frac{0 - \mu_{j,i}}{\varphi_i}\right), \\
\Pr(j \in \mathcal{B}|\mathbf{x}_i) &= \Pr(m_j > 0|\mathbf{x}_i, X, Y) \\
&= 1 - \Pr(j \in \mathcal{F}|\mathbf{x}_i),
\end{aligned} \tag{5.4}$$

where $\Psi(u) = 1/2[1 + \text{erf}(u/\sqrt{2})]$ is the cumulative distribution of a standard normal distribution, $\mu_{j,i} = \mathbf{y}_{:,j}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}$, and $\varphi_i = k_{i,i} - \mathbf{k}_{:,i}^T \mathbf{K}^{-1} \mathbf{k}_{:,i}$.

This proposed similarity measure brings significant advantages in efficient and robust exemplar based shape matching which will be discussed in later sections. To show this is a meaningful similarity measure, an illustration is given in figure 5.4. A randomly chosen human shape exemplar is taken and similarities from all training human shapes to the exemplar is calculated using equation (5.3). It shows that the proposed similarity measure is perceptually sound for shape matching.

5.4 Grouping of image regions

Given a new image $\mathcal{I}$ with segmented regions $R_c, c = 1, \dots, C$, the proposed region grouping approach originates from an intuitive idea: is there an object residing in the image by grouping one or more regions as the foreground $\mathcal{F}$ and the remaining as background $\mathcal{B}$? There are altogether 2^C candidates of binary silhouette images representing the shape of the partitioned foreground and background, where C is the number of distinct image regions. One possible approach is to compare each candidate with the training shapes by equation (5.3). The one with the maximum similarity to

Figure 5.4: Similarity between shapes measured by the proposed similarity measure. This figure illustrates 3 most similar and 6 most dissimilar shapes to the shape in the center.

the training exemplars is considered as the most likely candidate in the image, and it can be compared against a threshold to determine whether it belongs to the target object class. See figure 5.5 for an illustration.

Figure 5.5: (a) An image with segmented regions. There are altogether 22 distinct regions. (b) The combination of regions that is closest to a human shape, among altogether 2^{22} such combinations.

The number of regions C could be large, making the matching process extremely slow. It is proposed to take the advantage of the nice properties of the proposed similarity measure to tremendously speed up the matching process. The probabilities of each pixel being inside or outside the object boundary defined in equation (5.4) are pre-computed. Take the negative logarithm of the the probabilities in equation (5.4),

$$\begin{aligned} L(j \in \mathcal{F}|\mathbf{x}_i) &= -\log(\Pr(j \in \mathcal{F}|\mathbf{x}_i)), \\ L(j \in \mathcal{B}|\mathbf{x}_i) &= -\log(\Pr(j \in \mathcal{B}|\mathbf{x}_i)). \end{aligned} \quad (5.5)$$

$L(j \in \mathcal{F}|\mathbf{x}_i)$ and $L(j \in \mathcal{B}|\mathbf{x}_i)$ for all pixels $j = 1, \dots, q$ against all training exemplars $\mathbf{x}_i, i = 1, \dots, n$ can be pre-computed.

Given a partition of regions $\{R_c : c = 1, \dots, C\} = \mathcal{F} \cap \mathcal{B}$, we can then compute the likelihood of each region being residing in foreground ($\mathcal{F}$) or background ($\mathcal{B}$) conveniently.

$$L(R_c \subseteq \mathcal{F}|\mathbf{x}_i) = \sum_{j \in R_c} L(j \in \mathcal{F}|\mathbf{x}_i) \quad (5.6)$$

$$L(R_c \subseteq \mathcal{B}|\mathbf{x}_i) = \sum_{j \in R_c} L(j \in \mathcal{B}|\mathbf{x}_i) \quad (5.7)$$

Note that since it is assumed that the image has been segmented into coherent image regions, the object boundary cannot reside inside an image region. Therefore, the pixels in a region must be either all foreground pixels or all background pixels. The grouping of regions that forms the foreground and background with maximum likelihood is given by:

$$\mathcal{S}_i : \begin{cases} \mathcal{F} = \{R_c : L(R_c \subseteq \mathcal{F}|\mathbf{x}_i) < L(R_c \subseteq \mathcal{B}|\mathbf{x}_i)\} \\ \mathcal{B} = \mathcal{F}^c \end{cases} \quad (5.8)$$

where $\mathcal{S}_i$ represents the shape obtained by the grouping of image regions into foreground

and background. The likelihood of the obtained shape is

$$L(\mathcal{S}_i|\mathbf{x}_i) = \sum_{c=1}^C L(R_c \subseteq *|\mathbf{x}_i), \quad (5.9)$$

where $*$ is chosen based on equation (5.8).

In this way, the best grouping of image regions given a training exemplar can be found, and a probabilistic similarity measure is also obtained. The matched training exemplar is given by the minimum of the negative log likelihoods

$$\tilde{i} = \arg \min_{i=1,\dots,n} L(\mathcal{S}_i|\mathbf{x}_i). \quad (5.10)$$

In object detection, if $L(\mathcal{S}_{\tilde{i}}|\mathbf{x}_{\tilde{i}})$ is below a certain threshold, the object is considered present in the image.

A summary of the algorithm is given in table 5.1.

From the summary, we can see that the computation in the matching phase mainly involves $2nC$ summations. If the individual likelihoods of the pixels are not pre-computed, it is necessary to compute the likelihoods of all 2^C combinations of regions for n times using equation (5.1), which involves multiplication of large matrices. The result in equation (5.10) is precisely equivalent to finding the minimum of the $2^C n$ values. This is how a huge speed-up is obtained, without resorting to any approximations. The likelihood $L(\mathcal{S}_{\tilde{i}}|\mathbf{x}_{\tilde{i}})$ is guaranteed to be the global minimum.

5.5 Discussions

The overall shape matching process is illustrated in figure 5.6. It consists of an offline phase and an online phase. The offline phase includes GPLVM training and pre-computation of probability maps. The online phase includes initial image segmentation, grouping of image regions, and the final matching to the exemplars and segmentation

Table 5.1: Summary of the proposed shape matching algorithm

- Train a GPLVM model of the DT images of training exemplars of the target object classes $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times q}$, $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, α, γ, σ .
- Pre-compute the (negative log) likelihoods of each image pixel being foreground $L(j \in \mathcal{F}|\mathbf{x}_i)$ and background $L(j \in \mathcal{B}|\mathbf{x}_i)$ (equation (5.1)).
- Given an image or a detection window, obtain the segmented image regions $R_c, c = 1, \dots, C$, using any suitable image segmentation algorithm.
- for $i = 1 : n$
 - for $c = 1 : C$
 - * Compute the likelihoods of region R_c being foreground $L(R_c \in \mathcal{F}|\mathbf{x}_i)$ and background $L(R_c \in \mathcal{B}|\mathbf{x}_i)$ against training shape $\mathbf{x}_i$ (equations (5.6) and (5.7)).
 - * Take the smaller one of the two computed above, and R_c is determined to be foreground or background accordingly (equation (5.8)).
 - The similarity between the found grouping of image regions and training exemplar $\mathbf{x}_i$ is defined as the likelihood in equation (5.9).
- The likelihood of an object of the target object classes being present in the image is $L(\mathcal{S}_{\bar{i}}|\mathbf{x}_{\bar{i}})$, where $\bar{i}$ is the matched exemplar determined by equation (5.10).
- If $L(\mathcal{S}_{\bar{i}}|\mathbf{x}_{\bar{i}})$ is below a certain threshold, the image contains an object that belongs to the target object classes, and the precise segmentation of the object is also obtained by the shape $\mathcal{S}_{\bar{i}}$.

of the object.

The proposed region grouping based shape matching can be considered a combination of bottom-up and top-down approaches. The bottom-up part is in the form of

Figure 5.6: The proposed shape matching approach by region grouping.

image region constraints obtained from initial image segmentation. The figure-ground segregation is confined to the combinatorial space of the image regions. The top-down part is the modeling of the training shape exemplars by a GPLVM, and the computation of high level probability maps given the trained object classes. This is summarized in figure 5.7.

5.6 Experimental evaluation

Comparisons are made with the traditional edge based matching. Experiments are conducted on the non-rigid human shapes and a set of multiple object classes with varying view angles.

Figure 5.7: Combination of bottom-up and top-down constraints.

5.6.1 Human object class

The INRIA person data set [34] is used. The training set consists of 2416 humans captured in aligned image windows of size 112×48 pixels. Their binary silhouette images are manually delineated (see figure 5.1 for examples of training images and the corresponding binary silhouette images, edge images, and DT images). The test set consists of 1132 positive human image patches and 3000 negative image patches. The 3000 negative image patches are obtained by random sampling from the 300 negative images in INRIA person data set. The human objects in the INRIA person data set are mostly bystanders with no bias on the pose or view point. Some person-riding-a-bike objects are also included.

5.6.1.1 GPLVM training

A GPLVM model is applied on the 2416 training exemplars. The data space dimensionality is $112 \times 48 = 5376$. The latent variables are initialized using principal component analysis (PCA).

5.6.1.2 Determining latent space dimensionality

The latent space dimensionality is chosen to be 2 in the GPLVM model. Recall the GPLVM model selection algorithms proposed in section 2.4.2. Since the binary silhouette images are manually obtained, it can be assumed to be clean data without much noise. For clean data, the model can be selected by trying from a small latent space dimensionality and gradually increasing it. The flexibility of the model increases with it and the reconstructed images will become closer to the original images. On the other hand, smaller latent space dimensionality is always preferred, to prevent the model from overfitting. Therefore, a latent space dimensionality of 1 is tried first, and the reconstructed DT images are shown in figure 5.8. The 1D GPLVM model produces the ‘space filling curve’ (section 2.4.1) in the 5376-dimensional DT image space. It is observed that natural non-rigid variations of the human contour have been largely removed, keeping only 1 dominant variation, which is observed to correspond to the ‘width’ or ‘fatness’ of the human shape. It indicates that 1D latent space is not rich enough to model human shapes. A latent space dimensionality of 2 is then tried, which is observed to be flexible enough to model the natural variations of human shapes. The reconstructed images are smoothed versions of the original DT images, but still preserve enough variations to discriminate different shape exemplars (see figure 5.8).

5.6.1.3 Image region segmentation

For image region segmentation, the linear time multi-scale normalized cut (Ncut) image segmentation [27] is used. It is an efficient linear time algorithm that produces a fixed number of disjoint image regions. The features used in segmentation include color (RGB) and intervening contours. Large image graphs are compressed into multiple scales capturing image structure at increasingly large neighborhood. Graphs at multiple scales are partitioned simultaneously and consistently to obtain image segmentation. The number of image regions is set to three values, 30, 40 and 50. This generally results in slightly ‘over-segmented’ regions (see figure 5.11). Since the pro-

Figure 5.8: Top: sample DT images from the training set. Middle: reconstructed images with a 1D GPLVM model. Bottom: reconstructed images with a 2D GPLVM model. The images are sorted (left to right, top to bottom) in ascending order of the 1D latent variables of the 1D GPLVM model.

posed matching method works by re-grouping of image regions, it is preferred that the initial segmentation is ‘over-segmentation’ rather than ‘under-segmentation’.

5.6.1.4 Edge based shape matching

For shape matching by Chamfer distance and Hausdorff distance, the edge image is first obtained using the Sobel operator [60]. The threshold in determining whether an edge exists is also examined at three different values: the root mean square (RMS) value of the magnitude of the gradients, 0.1 and 0.05. The edges detected at the three thresholds are shown in figure 5.9.

The chamfer distance is given by

$$D_{\text{Chamfer}}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t), \quad (5.11)$$

where $|T|$ denotes the number of edge pixels in the template T and $d_I(t)$ denotes the distance between the edge point t in T and the closest edge pixel of the image I . The Hausdorff distance, on the other hand, is taking the maximum instead of the average.

$$D_{\text{Hausdorff}}(T, I) = \max_{t \in T} d_I(t), \quad (5.12)$$

Figure 5.9: (a) Original image; (b) Edges detected at threshold equal to the root mean square (RMS) value of the magnitude of the gradients; (c) Edges detected at threshold = 0.1; (d) Edges detected at threshold = 0.05.

5.6.1.5 Detection and segmentation performance

Both the edge based approaches and the proposed region grouping approach can be used to detect the presence of objects by thresholding the similarity to the matched exemplar. Figure 5.10 plots the false negative rate (miss detections) vs. false positive rate (false detections) by varying the thresholds at uniform intervals for both methods. It is seen that the proposed method outperforms both Chamfer distance matching and Hausdorff distance matching. Moreover, while both edge based matching methods are sensitive to the thresholds used in the edge detector, the proposed region based matching is more robust to the clutter of initial segmentation. The edge based approaches perform best with the edge detection threshold that produces least edges, which shows that they are prone to scatter in the images.

In figure 5.11, some detection and segmentation results are shown. The proposed method is shown with an initial segmentation of 40 regions. Both Chamfer distance matching and Hausdorff distance matching are shown with the edge detection threshold that gives best detection results. It is also observed that the proposed method gives a more precise segmentation. The edge matching approaches could only use the contour of the matched exemplar as the segmentation of the test image. Its accuracy is dependent on whether there is a training exemplar that matches the image closely. The segmentation contour may not follow the edges in the image. In contrast, the proposed method finds an optimal grouping of image regions that can be generalized from the training exemplars. The final segmentation could be different from the matched exemplar.

The average time needed for preprocessing and matching on a PC (Matlab® implementation) is shown in table 5.2. The preprocessing for the proposed method is image segmentation by multi-scale Ncut. The preprocessing for matching by Chamfer and Hausdorff distance is edge detection by the Sobel operator. It is observed that both preprocessing and matching stages of the proposed method take longer than the edge based counterparts. Hierarchical matching can be used to further speed up the

Figure 5.10: False negative rate versus false positive rate. The proposed region grouping approach is evaluated with initial segmentations into 30, 40, and 50 image regions. Correspondingly, the edge based matching (Chamfer and Hausdorff) is evaluated at three different edge detection thresholds.

matching process. The higher computational demand of the proposed region-based shape matching is mainly due to the new approach taken to group image regions into meaningful objects. Although GPLVM itself is a computationally intensive method, it does not slow down the proposed approach as the high computation is only involved in the training phase. The probability maps are pre-computed once the model is trained. From table 5.2, the preprocessing step for the proposed method (initial region segmentation) takes much longer than the pre-processing step (edge detection) of edge based approaches. Therefore, the higher computational demand is shared by all region-based approaches but it is dependent on which segmentation method is used.

Table 5.2: Average time used for shape matching (unit: second)

	proposed method	Chamfer	Hausdorff
preprocessing	1.09	0.0048	
matching	0.9058	0.4896	0.2094

5.6.2 Multiple object classes

The proposed method is readily extendable to multiple object classes, since GPLVM is capable of handling multiple connected components. An illustrative experiment is conducted on 6 object classes from the COIL100 data set [101]. For each object, there are 72 samples showing the object at uniform 5 degrees viewpoint changes. The binary silhouette images are obtained by thresholding the RGB values, since the images are taken under controlled environment with relatively uniform background. There are some mislabeled pixels due to color variations. These are not manually removed but considered as noise. 300 samples are randomly selected out of the total of $72 \times 6 = 432$ images as the training set (about 70%). Although this is a simple data set with uniform backgrounds, without prior knowledge on the background, precisely segmenting out the object is still a difficult problem. Examples of training images and the corresponding silhouette images are shown in figure 5.12

The images are resized to 32×32 . All 300 training samples are modeled with GPLVM in a 1D latent space. In other words, the space filling curve is capable of representing the original training exemplars in 1024 dimensional input space. The reconstructed images sorted in the ascending order of the latent variables are shown in figure 5.13. Note that the intrinsic degree of freedom of the manifold formed by images of the same object at different viewing angles is indeed 1. The space filling curve found by GPLVM connects different objects sequentially.

Since the images are mainly characterized by color, color quantization is used to obtain an initial segmentation of the images, which is much faster than the multi-scale Ncut used in the previous section. The proposed region grouping method is then used

Figure 5.11: Sample detection and segmentation results. (a) test images. (b) initial segmentation into 40 regions. (c) matched exemplar using the proposed region grouping approach. (d) segmentation contour using the proposed method. (e) edge detected using the threshold that gives best detection results. (f) contour of matched exemplar using Chamfer matching. (g) contour of matched exemplar using Hausdorff matching.

Figure 5.12: 6 object classes of the COIL100 data set and their corresponding silhouette images.

Figure 5.13: Reconstructed DT images from the 1D latent space. The images are sorted in the ascending order of the latent variables. This is the space filling curve found by GPLVM in the 1024 dimensional input space.

to find the best grouping of image regions to form any of the learned object classes. The final segmentation is obtained by choosing the grouping of regions with the maximum

likelihood of being generated by the training exemplars. Sample results are shown in figure 5.14. Segmentation with consistent quality is obtained on all test images. Compared against the ground truth, the average percentage of correctly labeled pixels in each image reaches $96.81 \pm 1.71\%$ in the 132 test images.

Figure 5.14: From left to right: the original test image, the initial segmentation by color quantization, and the final segmentation result.

5.7 Concluding remarks

In this chapter the proposed region grouping approach to shape matching is evaluated. It is observed the proposed method is more robust than edge based matching using Chamfer and Hausdorff distances in terms of detection of target object classes and segmentation of the objects, at a cost of slower computation. Aspects that could make the method more practical and efficient will be the focus of future work.

Speeding up the method for real-time applications The common practice to speed up exemplar based matching is hierarchical shape description by clustering the

exemplars and a coarse-to-fine matching scheme.

Scaling problem In the experiments conducted here, the objects haven been properly scaled to about the same size. To deal with object scaling problem in real applications, there are two approaches. One is to perform a sliding window detection at various scales by resizing the image. The second approach is to include training exemplars at difference scales, similar to that in [51]. This will further enlarge the training set. Hierarchical shape description by clustering and coarse-to-fine matching schemes can be adopted here to help reduce the computation.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

In this thesis, Gaussian process is explored to solve interesting pattern analysis problems, ranging from the basic problems of data processing (manifold denoising) and similarity learning, to the more practical application on shape matching. The contributions to the respective domain of research have been discussed. These different applications are connected by the GP philosophy of problem solving. In the following, a summary of the main contributions in this thesis is given, followed by a discussion on the advantages of using GPs to solve the problems.

Nonparametric similarity learning Similarity learning is usually more flexible than distance metric learning, which is required to follow the axioms of a metric. However, previous similarity learning methods also assume certain forms of the similarity measure, such as a parametric form or a combination of elementary similarities. In this thesis, a novel nonparametric similarity learning method is proposed. It models pairwise similarity with a Gaussian process. The key advantage of the proposed method is that it assumes neither a parametric functional form of the similarity measure, nor a particular data distribution. The learned similarity measure is flexible enough to model

highly nonlinear, multi-modal and noisy data. Extensive experimental comparisons are made with state-of-the-art approaches, and the proposed method compares favorably in terms of both clustering and classification performance.

Manifold denoising A new manifold denoising method is developed. It is based on the findings between GP models and smooth manifolds. A Gaussian prior process generates smooth manifolds, while a Gaussian posterior process can be used to model and make inference on a given manifold. Gaussian process latent variable models, on the other hand, can be used to find a piecewise smooth embedding of a manifold, or manifolds of connected components. Thus, GPLVM can be applied to solve the manifold denoising problem, and it shows promising results from the comparisons with recent methods that either adopt a local or global subspace model, or take a diffusion approach to remove noise directly in the input space.

Shape based matching by region grouping This work presents a novel region grouping approach to shape matching. The image is initially segmented into coherent image regions. Groupings of regions that form valid target object classes are then obtained. This is accomplished by the training of a GPLVM on a set of object shape exemplars and pre-computation of the pixel probability maps. The groupings of image regions can be efficiently computed with linear complexity with respect to the number of image regions. Comparisons are made with the edge based Chamfer and Hausdorff distance matching. The results indicate that the region based approach outperforms the edge based approaches in terms of both discrimination of objects and non-objects and segmentation accuracy of the objects.

In the above works based on GPs, it is important to appreciate the advantages brought by the GP framework. The first advantage is the use of maximum marginal likelihood in model (hyperparameter) selection. As discussed in section 2.3.3, maximum marginal likelihood estimation is a viable solution to model selection. There is a natural trade-off between model fit and model complexity due to the Occam's razor principle. In chapter 3, the hyperparameters control the smoothness of the similarity

measure and are optimized by maximizing the likelihood of the given similarity constraints. In chapter 4, both the latent variables and the hyperparameters are tuned by maximum marginal likelihood to find a piecewise smooth embedding of the manifold for the purpose of manifold denoising. In chapter 5, the GPLVM model is optimized by maximum marginal likelihood to obtain a generative model of target object shapes. Furthermore, maximum marginal likelihood estimation can also be used to help select the latent space dimensionality in GPLVM. A practical algorithm is proposed in section 2.4.2. It has been validated by the experiments in both chapters 4 and 5.

The second advantage is in noise handling. Under the Bayesian inference framework, noise is modeled in terms of probability distribution. In chapter 3, it has been demonstrated on various data sets that noise and outliers can be well handled by the proposed nonparametric similarity learning method. Chapter 4 is a straightforward adoption of the GP noise model as the manifold noise model, due to the connections between GPs and manifolds established in this chapter. In chapter 5, it has been shown that imperfections in obtaining training silhouette images on the COIL100 data set can be tolerated in the modeling of training shape exemplars by GPLVM.

The last common advantage shared by the works in this thesis is the flexibility of data modeling by GPs in the form of smoothness constraints. GPs are nonparametric models which generally do not assume a particular data distribution. Nonetheless, GP models constrain the learning problem with the choices of the mean function and the covariance function. The mean function is conventionally set to zero. The covariance function used in this thesis is the squared exponential (SE) covariance function. Functions sampled from a GP prior with the SE covariance function are smooth functions. In chapter 3, the proposed nonparametric similarity learning method does not assume any parametric form of the similarity measure. It only assumes that pairwise similarity should be a smooth function with respect to both inputs. In chapter 4, it is shown that a Gaussian prior process with SE covariance function generates smooth manifolds while a Gaussian posterior process with SE covariance function models smooth mani-

folds well. In chapter 5, the set of training shape exemplars are smoothed to construct a generative model of target object classes.

Although the proposed methods enjoy advantages brought by the GP framework, limitations are also introduced in the process. First, the squared exponential covariance function is broadly used in this thesis. The squared exponential covariance function suggests smoothness which is generally preferred in many learning tasks. However, it has been argued that strong smoothness assumptions can be unrealistic in modeling physical processes [127]. Other covariance functions can be considered for specific tasks. Second, the GP models are relatively more computationally demanding. For larger datasets, approximated models may need to be explored.

6.2 Recommendations for further research

For manifold denoising using the proposed method, other noise models can be considered, although it has been shown here that the Gaussian noise model works well for other types of noises in images. With other noise models, the solution may no longer be analytically tractable, and approximation models may be needed. Effects of such approximations need to be considered carefully. The findings between GPs and manifolds such as the Gaussian prior process manifolds, Gaussian posterior process manifolds, and GPLVM as a piecewise smooth embedding of manifolds, may have much more far-reaching potential applications than just manifold denoising considered in this thesis. Appendix A gives a preliminary illustration of its potential application.

For nonparametric similarity learning, the focus of further research can be on the full use of the probabilistic similarity measure learned. For example, the variance of the learned pairwise similarity can be incorporated into the clustering or classification process. Those with small variance can be given higher weights when determining the cluster or class assignments.

For the proposed method of shape matching by region grouping, it has been eval-

uated in its basic form. Further work may be explored to make it suitable for object detection and segmentation tasks. A system incorporating hierarchical matching, detection and segmentation can be developed. Besides the exemplar based matching approach considered in this thesis, optimization based matching by iteratively searching for the best match in the latent space is also a desirable direction to work on.

Appendix A

Drawing Manifold by Gaussian posterior process

Following the discussion on Gaussian prior process manifold and Gaussian posterior process manifold, a preliminary application is discussed below. Suppose that we want to generate a lattice of points to form a smooth manifold that we have in mind, for example, a 3D ‘yellow ribbon’. Figure A.1 illustrates the procedure to do this by making use of the Gaussian posterior process. In (a), a few points are drawn in 2D to form a rough structure of the intended manifold (for example, using Matlab® *impixel* function). The requirement is that the points are drawn in sequential order, but they do not need to be very smooth or evenly spaced. In (b) and (c), a regression function is learned for the mapping from $x = 0 : \eta : 1$, $\eta = 1/(n - 1)$ to each dimension of the drawn points $y^{(1)}$ and $y^{(2)}$, where n is the number of points drawn. Consequently, a smooth reconstruction can be found in the 2D space (figure A.1 (d)). To sample a 3D manifold, a lattice of points are first sampled at regular grids $\{0 : 0.01 : 1, 0 : 0.01 : 0.1\}$ in a 2D space (each point is denoted by $[x_t, z] \in \mathbb{R}^2$, where N is the number of points sampled) (figure A.1 (e)), and the corresponding $y_t^{(1)}$ and $y_t^{(2)}$ are found by mapping x_t using the learned regression functions. Finally, the 3D ribbon is formed by the three components $[y_t^{(1)}, y_t^{(2)}, z]$ (figure A.1 (f)).

Figure A.1: The procedure to generate a 3D 'yellow ribbon' from a few hand-drawn points in 2D.

Appendix B

Handling Missing Data in Manifold Denoising

In [82], the author has shown that GPLVM could fill in the missing values in test data after modeling on a complete training set. Here it is shown that the proposed denoising algorithm is able to handle large percentages of missing values present in the given data in an unsupervised fashion through the denoising process, and again the reconstruction can be efficiently kernelized to handle nonlinear mappings. Since the missing entries from each dimension in the data could be different, the likelihood of each dimension of the observed variables becomes

$$p(\mathbf{y}_{\mathbf{sub}_j,j} | X_{sub_j,:}) = \mathcal{N}(0, K_{\mathbf{sub}_j}^{-1}), \quad (\text{B.1})$$

where $\mathbf{sub}_j$ is the index vector of the non-missing entries in the j th dimension, so that $\mathbf{y}_{\mathbf{sub}_j,j}$ is a sub-vector of the j th column of Y , $X_{sub_j,:}$ is a sub-matrix of X which is obtained by deleting the rows corresponding to the missing entries in the j th dimension, and $K_{\mathbf{sub}_j}$ is a sub-matrix of K obtained by deleting the rows and columns corresponding to the missing entries in the j th dimension.

Now the likelihood function becomes

$$\begin{aligned}
& P(Y|X) \\
&= \prod_{j=1}^q p(\mathbf{y}_{\text{sub}_j,j} | X_{\text{sub}_j,:}) \\
&= \frac{1}{(2\pi)^{\frac{qn}{2}} \prod_{j=1}^q |K_{\text{sub}_j} + \sigma^2 I|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \sum_{j=1}^q (\mathbf{y}_{\text{sub}_j,j}^T (K_{\text{sub}_j} + \sigma^2 I)^{-1} \mathbf{y}_{\text{sub}_j,j})\right)
\end{aligned} \tag{B.2}$$

and the MAP estimate of $\mathbf{w}_j$ is

$$\mathbf{w}_{j\text{MAP}}^T = (\sigma^2 I + X_{\text{sub}_j,:}^T X_{\text{sub}_j,:})^{-1} X_{\text{sub}_j,:}^T \mathbf{y}_{\text{sub}_j,j} \tag{B.3}$$

and the reconstruction back to the data space is

$$\begin{aligned}
& \mathbf{z}_{:,j} \\
&= X \mathbf{w}_{j\text{MAP}}^T \\
&= X (\sigma^2 I + X_{\text{sub}_j,:}^T X_{\text{sub}_j,:})^{-1} X_{\text{sub}_j,:}^T \mathbf{y}_{\text{sub}_j,j} \\
&= X X_{\text{sub}_j,:}^T (X_{\text{sub}_j,:} X_{\text{sub}_j,:}^T + \sigma^2 I)^{-1} \mathbf{y}_{\text{sub}_j,j} \\
&= K_{:,\text{sub}_j} (K_{\text{sub}_j}^{-1} + \sigma^2 I) \mathbf{y}_{\text{sub}_j,j}
\end{aligned} \tag{B.4}$$

where $K_{:,\text{sub}_j}$ is obtained by taking the columns of K corresponding to the non-missing entries, and the derivation is again by the Sherman-Morrison formula. Both the marginal likelihood function (equation (B.2)) and the reconstruction formulae (equation (B.4)) are dependent only on K , which again allows the use of nonlinear covariance functions. The marginal likelihood function can be used for maximum marginal likelihood estimation of hyperparameters.

Appendix C

Proof of Equation (3.16)

We need to show that $P(K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)P = (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)$. Given the definition of P in equation (3.14), $P(K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)P$ is equivalent to first swap every two rows of the matrix in the middle, followed by the columns. Taking a 4×4 matrix for example, let

$$A = \begin{bmatrix} a1 & a2 & a3 & a4 \\ a5 & a6 & a7 & a8 \\ a9 & a10 & a11 & a12 \\ a13 & a14 & a15 & a16 \end{bmatrix}, \quad (\text{C.1})$$

then

$$PAP = \begin{bmatrix} a6 & a5 & a8 & a7 \\ a2 & a1 & a4 & a3 \\ a14 & a13 & a16 & a15 \\ a10 & a9 & a12 & a11 \end{bmatrix}. \quad (\text{C.2})$$

It can be seen that the permutation does a diagonal swap in both directions on every 2×2 sub-matrix. Therefore it is equivalent to show that $PK(\mathcal{T}, \mathcal{T})P = K(\mathcal{T}, \mathcal{T})$, since after such permutation the diagonal matrix $\sigma^2 I$ remains the same. Consider a 2×2

sub-matrix of $K(\mathcal{T}, \mathcal{T})$ and its permuted version

$$\begin{bmatrix} k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n-1}) & k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n}) \\ k(\mathbf{t}_{2m}, \mathbf{t}_{2n-1}) & k(\mathbf{t}_{2m}, \mathbf{t}_{2n}) \end{bmatrix} \rightarrow \begin{bmatrix} k(\mathbf{t}_{2m}, \mathbf{t}_{2n}) & k(\mathbf{t}_{2m}, \mathbf{t}_{2n-1}) \\ k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n}) & k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n-1}) \end{bmatrix}. \quad (\text{C.3})$$

Due to the way the training set $\mathbf{t}$ is arranged and the isotropic nature of the SE covariance matrix, $k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n-1}) = k(\mathbf{t}_{2m}, \mathbf{t}_{2n})$, and $k(\mathbf{t}_{2m-1}, \mathbf{t}_{2n}) = k(\mathbf{t}_{2m}, \mathbf{t}_{2n-1})$. Hence $P(K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)P = (K(\mathcal{T}, \mathcal{T}) + \sigma^2 I)$.

Author's Publications

1. **Y. Gao**, L. Wang, K.L. Chan, W.Y. Yau, "Learning Texture Similarity with Labels and Pairwise Distances", Proceedings of the 4th International Workshop on Texture Analysis and Synthesis, Oct 2005, China.
2. L. Wang, **Y. Gao**, K.L. Chan, P. Xue, W.Y. Yau, "Retrieval with Knowledge-driven Kernel Design: An Approach to Improving SVM-based CBIR with Relevance Feedback", 10th IEEE International Conference on Computer Vision (ICCV), Oct 2005, China.
3. **Y. Gao**, L. Wang, K.L. Chan, W.Y. Yau, P. Xue, "Improving SVM-based Content Based Image Retrieval through Dynamically Refreshed Kernel Learned with a Fast and Robust Model Selection Criterion", Asia-Pacific Workshop on Visual Information Processing (VIP), 11-13 Dec 2005, Hong Kong.
4. K.L. Chan, **Y. Gao**, W. Huang, J. Zhou, V. Chong, "Nasopharyngeal Carcinoma Lesion Extraction from MR Images using Spectral Clustering with Learned Metric", 3rd Asian Conference on Computer Aided Surgery (ACCAS), 2007.
5. **Y. Gao**, K.L. Chan, W.Y. Yau, "Learning in Content Based Image Retrieval - A Brief Review", 6th International Conference on Information, Communications and Signal Processing (ICICS), Dec 2007, Singapore.
6. W. Huang, **Y. Gao**, K.L. Chan, "A Review of Region-Based Image Retrieval", Journal of Signal Processing Systems, Springer, 2008.
7. W. Huang, K.L. Chan, **Y. Gao**, V. Chong, "Nasopharyngeal Carcinoma Lesion Extraction using Clustering via Semi-supervised Metric Learning with Side-information", 5th IET Visual Information Engineering Conference (VIE), 2008, China.
8. W. Huang, K.L. Chan, **Y. Gao**, J. Zhou, V. Chong, "Semi-supervised Nasopharyngeal Carcinoma Lesion Extraction from Magnetic Resonance Images using

- Online Spectral Clustering with a Learned Metric”, 11th International Conference on Medical Image Computing and Computer Assisted Intervention, MICCAI 2008, New York USA.
9. **Y. Gao**, K.L. Chan, W.Y. Yau, “Manifold Denoising with Gaussian Process Latent Variable Models”, 19th International Conference on Pattern Recognition, 2008 , Tampa Florida, USA.

Bibliography

- [1] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1606–1621, 2004.
- [2] G. H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.
- [3] E. Bala and A. E. Cetin. Computationally efficient wavelet affine invariant functions for shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1095–1099, 2004.
- [4] D. Barber and C. K. I. Williams. Gaussian processes for Bayesian classification via hybrid Monte Carlo. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS) 9*, pages 133–165. The MIT Press, 1997.
- [5] D. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis*. London: Arnold, 1999.
- [6] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, New York, NY, USA, 2004. ACM.
- [7] S. Basu, M. Bilenko, and R. J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of*

- the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 42–49. Springer, August 2003.
- [8] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC Press, 2008.
 - [9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
 - [10] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
 - [11] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. *Eighth IEEE International Conference on Computer Vision (ICCV)*, 1:454–461 vol.1, 2001.
 - [12] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr 2002.
 - [13] C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.
 - [14] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006.
 - [15] E. Borenstein and J. Malik. Shape guided object segmentation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:969–976, 17-22 June 2006.
 - [16] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, Nov 1988.
 - [17] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman. Linear time Euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:529–533, 1995.

- [18] M. Brookes. The matrix reference manual. [online] <http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html>, 2005.
- [19] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [20] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- [21] H. Chen, G. Jiang, and K. Yoshihira. Robust nonlinear dimensionality reduction for manifold learning. *International Conference on Pattern Recognition (ICPR)*, 2:447–450, 2006.
- [22] J. Chen, Z. Zhao, J. Ye, and H. Liu. Nonlinear adaptive distance metric learning for clustering. In *the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 123–132, 2007.
- [23] V. Cheng, C.-H. Li, J. T. Kwok, and C.-K. Li. Dissimilarity learning for nominal data. *Pattern Recognition*, 37(7):1471–1477, July 2004.
- [24] T.-J. Chin, L. Wang, K. Schindler, and D. Suter. Extrapolating learned manifolds for human activity recognition. *IEEE International Conference on Image Processing (ICIP)*, 1:I –381–I –384, Oct 2007.
- [25] H. Choi and S. Choi. Kernel ISOMAP on noisy manifold. *International Conference on Development and Learning*, 0:208–213, 2005.
- [26] T. Choi and M. J. Schervish. Posterior consistency in nonparametric regression problems under Gaussian process priors. Technical Report 809, Department of Statistics, CMU, 2004.
- [27] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:1124–1131 vol. 2, June 2005.
- [28] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall, 2001.

- [29] D. Cremers. Nonlinear dynamical shape priors for level set segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, June 2007.
- [30] D. Cremers, T. Kohlberger, and C. Schnörr. Shape statistics in kernel space for variational image segmentation. *Pattern Recognition*, 36(9):1929–1943, 2003.
- [31] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
- [32] N. Cristianini, J. Shawe-Taylor, and J. Kandola. On kernel target alignment. *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 367–373, 2002.
- [33] L. Csat and M. Opper. Sparse representation for Gaussian process models. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems (NIPS) 13*. The MIT Press, 2001.
- [34] N. Dalai and B. Triggs. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:886–893 vol. 1, June 2005.
- [35] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning (ICML)*, pages 209–216, 2007.
- [36] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, New York, 1996.
- [37] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556, New York, NY, USA, 2004. ACM.
- [38] P. Dollár, V. Rabaud, and S. Belongie. Learning to traverse image manifolds. In *Advances in Neural Information Processing Systems (NIPS)*, Dec. 2006.

- [39] C. Domeniconi, D. Gunopulos, and J. Peng. Large margin nearest neighbor classifiers. *IEEE Transactions on Neural Networks*, 16(4):899–909, 2005.
- [40] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2nd edition, 2001.
- [41] L. Eciolaza, M. Alkarouri, N. D. Lawrence, V. Kadiramanathan, and P. J. Fleming. Gaussian process latent variable models for fault detection. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 287–292. IEEE, 2007.
- [42] I. El-Naqa, Y. Yang, N. Galatsanos, R. Nishikawa, and M. Wernick. A similarity learning approach to content-based image retrieval: application to digital mammography. *IEEE Transactions on Medical Imaging*, 23(10):1233–1244, Oct 2004.
- [43] A. Elgammal and C.-S. Lee. Inferring 3D body pose from silhouettes using activity manifold learning. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:II–681–II–688 Vol.2, June-2 July 2004.
- [44] F. Esposito, D. Malerba, V. Tamma, and H. Bock. Classical resemblance measures. In H. Bock and E. Diday, editors, *Analysis of Symbolic Data*, pages 139–152. Springer, 2000.
- [45] M. Farmer and A. Jain. A wrapper-based approach to image segmentation and classification. *IEEE Transactions on Image Processing*, 14(12):2060–2072, Dec. 2005.
- [46] J. Fournier and M. Cord. Long-term similarity learning in content-based image retrieval. *Proceedings. 2002 International Conference on Image Processing*, 1:I–441–I–444 vol.1, 2002.
- [47] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems (NIPS) 19*, pages 417–424. 2007.

- [48] P. Gao, A. Honkela, M. Rattray, and N. D. Lawrence. Gaussian process modelling of latent chemical species. *Bioinformatics*, 24(16):i70–i75, 2008.
- [49] D. Gavrilă, J. Giebel, and H. Neumann. Learning shape models from examples. *Pattern Recognition*, page 369, 2001.
- [50] D. Gavrilă and V. Philomin. Real-time object detection for smart vehicles. *The Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*, 1:87–93 vol.1, 1999.
- [51] D. M. Gavrilă. A Bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1408–1421, 2007.
- [52] D. M. Gavrilă and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59, 2007.
- [53] Y. Gdalyahu, D. Weinshall, and M. Werman. Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.
- [54] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 1995.
- [55] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, 1998.
- [56] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS) 18*, pages 451–458. 2006.
- [57] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, June 2007.
- [58] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: a Gaussian process treatment. In *Advances in Neural Informa-*

- tion Processing Systems (NIPS) 10*, pages 493–499, Cambridge, MA, USA, 1998. MIT Press.
- [59] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS) 17*, pages 513–520. 2005.
 - [60] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Wellesley-Cambridge Press, 2nd edition, 2002.
 - [61] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Special Interest Group on Graphics and Interactive Techniques Conference (SIGGRAPH)*, pages 522–531, 2004.
 - [62] G. Guo, S. Z. Li, and K. L. Chan. Learning similarity for texture image retrieval. In *Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 178–190, London, UK, 2000. Springer-Verlag.
 - [63] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.
 - [64] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
 - [65] X. He, W.-Y. Ma, and H.-J. Zhang. Learning an image manifold for retrieval. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 17–23, New York, NY, USA, 2004. ACM.
 - [66] M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in Euclidean space. *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 289–296, 2005.
 - [67] M. Hein and M. Maier. Manifold denoising. *Advances in Neural Information Processing Systems (NIPS)*, pages 561–568, 2006.
 - [68] A. Hertzmann. Introduction to Bayesian learning. In *ACM SIGGRAPH 2004 Course Notes*, page 22, New York, NY, USA, 2004. ACM.

- [69] S. C. H. Hoi, R. Jin, and M. R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the 24th International Conference on Machine Learning*, pages 361–368, New York, NY, USA, 2007. ACM.
- [70] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.
- [71] T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer Academic Publishers, 2004.
- [72] S. Kakade, M. Seeger, and P. Foster. Worst-case bounds for Gaussian process models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS) 18*. The MIT Press, 2006.
- [73] I. Katz and H. Aghajan. Multiple camera-based chamfer matching for pedestrian detection. In *Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–5, Sep 2008.
- [74] T. Kohonen. *Self-organizing maps*. Springer, 2001.
- [75] K. Kummamuru, R. Krishnapuram, and R. Agrawal. On learning asymmetric dissimilarity measures. *Fifth IEEE International Conference on Data Mining*, 27–30 Nov. 2005.
- [76] J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. *International Conference on Machine Learning (ICML)*, pages 408–415, 2003.
- [77] T. Lange, M. H. C. Law, A. K. Jain, and J. M. Buhmann. Learning with constrained and unlabelled data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 731–738, Washington, DC, USA, 2005. IEEE Computer Society.
- [78] J. Laub and K.-R. Müller. Feature discovery in non-metric pairwise data. *J. Mach. Learn. Res.*, 5:801–818, 2004.

- [79] N. Lawrence. Learning for larger datasets with the Gaussian process latent variable model. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*. Omnipress, 2007.
- [80] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, pages 609–616. The MIT Press, 2003.
- [81] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems 16*, 2004.
- [82] N. D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, pages 1783–1816, 2005.
- [83] N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 481–488, New York, NY, USA, 2007. ACM.
- [84] A. Levinshstein, C. Sminchisescu, and S. Dickinson. Learning hierarchical shape models from examples. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 251–267, 2005.
- [85] W.-K. Liao and G. Medioni. 3D face tracking and expression inference from a 2D sequence using manifold learning. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.
- [86] Y. Liu, R. Jin, and A. K. Jain. Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 450–459, New York, NY, USA, 2007. ACM.
- [87] H. Long and W. K. Leow. Perceptual texture space improves perceptual consistency of computational features. *Proceedings of Internatioanl Joint Conference on Artificial Intelligence*, pages 1391–1396, 2001.

- [88] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.
- [89] M. Seeger. Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations. PhD thesis, Institute of Adaptive and Neural Computation, University of Edinburgh, 2003.
- [90] W. Y. Ma and B. S. Manjunath. Texture features and learning similarity. *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 425–430, 1996.
- [91] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI, pages 133–165. Springer, 1998.
- [92] D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.
- [93] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):837–842, August 1996.
- [94] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.
- [95] S. Melacci, L. Sarti, M. Maggini, and M. Bianchini. A neural network approach to similarity learning. In *International Workshop on Artificial Neural Networks in Pattern Recognition*, pages 133–136, 2008.
- [96] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. *Advances in Neural Information Processing Systems (NIPS)*, pages 536–542, 1998.
- [97] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. volume 1, pages 26–36, June 2006.
- [98] H. minn Lu, Y. Fainman, and R. Hecht-Nielsen. Image manifolds. In *Proceedings of the SPIE Symposium on Electronic Imaging: Science and Technology; Conference on Artificial Neural Networks in Image Processing III*, 1998.

- [99] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [100] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, Mar 2001.
- [101] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-100). *Technical Report CUCS-006-96*, February 1996.
- [102] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.
- [103] C. Olson. A probabilistic formulation for Hausdorff matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 150–156, Jun 1998.
- [104] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.
- [105] J. Park, Z. Zhang, H. Zha, and R. Kasturi. Local smoothing for manifold learning. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:II-452–II-459 Vol.2, June-2 July 2004.
- [106] M.-T. Pham and T.-J. Cham. Fast training and selection of Haar features using statistics in boosting-based face detection. *IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–7, Oct 2007.
- [107] M.-T. Pham, V.-D. Hoang, and T.-J. Cham. Detection with multi-exit asymmetric boosting. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.
- [108] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.
- [109] V. Randrianarisoa, J.-F. Bernier, and R. Bergevin. Detection of multi-part objects by top-down perceptual grouping. *Proceedings of the 2nd Canadian conference on Computer and Robot Vision*, pages 536–543, 2005.

- [110] A. R. Rao and G. L. Lohse. Identifying high level features of texture perception. *CVGIP: Graphical Models and Image Processing*, 55:218–233, May 1993.
- [111] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [112] J. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 2nd edition, 1995.
- [113] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.
- [114] C. Ruiz, M. Spiliopoulou, and E. M. Ruiz. C-dbscan: Density-based clustering with constraints. In A. An, J. Stefanowski, S. Ramanna, C. J. Butz, W. Pedrycz, and G. Wang, editors, *International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 216–223. Springer, 2007.
- [115] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.
- [116] D. Schlüter and S. Posch. Combining contour and region information for perceptual grouping. *German Association for Pattern Recognition (DAGM) Symposium*, pages 393–401, 1998.
- [117] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
- [118] M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems (NIPS) 12*, pages 603–609. The MIT Press, 2000.
- [119] M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14:69–106, 2004.
- [120] M. Seeger. Bayesian modelling for data analysis and learning from data. Invited Lecture IK 2006 Neuroscience Summer School, 2006.
- [121] M. Seeger and C. K. I. Williams. Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on AI and Statistics 9*, 2003.

- [122] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.
- [123] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 776–792, 2002.
- [124] L. Si, R. Jin, S. C. H. Hoi, and M. R. Lyu. Collaborative image retrieval via regularized metric learning. *ACM Multimedia Systems Journal (MMSJ), Special Issue on Machine Learning Approaches to Multimedia Information Retrieval*, 12(1):34–44, 2006.
- [125] R. Souvenir and R. Pless. Manifold clustering. *Tenth IEEE International Conference on Computer Vision (ICCV)*, 1:648–653 Vol. 1, Oct 2005.
- [126] R. Souvenir and R. Pless. Manifold clustering. *Tenth IEEE International Conference on Computer Vision (ICCV)*, 1:648–653 Vol. 1, Oct 2005.
- [127] M. L. Stein. *Interpolation of Spatial Data*. Springer-Verlag, 1999.
- [128] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 3rd edition, 2003.
- [129] M. Sugiyama. Local Fisher discriminant analysis for supervised dimensionality reduction. In *International Conference on Machine Learning (ICML)*, pages 905–912, New York, NY, USA, 2006.
- [130] T. P. Minka. A family of algorithms for approximate Bayesian inference. PhD thesis, Massachusetts Institute of Technology, 2001.
- [131] H. Tan and Y.-J. Zhang. Computing eigenface from edge images for face recognition based on Hausdorff distance. In *Fourth International Conference on Image and Graphics*, pages 639–644, Aug. 2007.
- [132] T. Tangkuampien and D. Suter. Human motion de-noising via greedy kernel principal component analysis filtering. volume 3, pages 457–460, 0-0 2006.

- [133] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.
- [134] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–127–I–133 vol.1, June 2003.
- [135] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *International Journal of Computer Vision*, 48(1):9–19, 2002.
- [136] V. Tresp. Dirichlet processes and nonparametric Bayesian modelling. *Tutorial at the Machine Learning Summer School 2006 in Canberra*, 2006.
- [137] Z. Tu. An integrated framework for image segmentation and perceptual grouping. *Proceedings of the Tenth IEEE International Conference on Computer Vision*, 1:670–677, 2005.
- [138] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.
- [139] R. Urtasun and T. Darrell. Discriminative Gaussian process latent variable models for classification. In *International Conference on Machine Learning (ICML)*, June 2007.
- [140] R. Urtasun, D. J. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking from small training sets. *Proceedings of the Tenth IEEE International Conference on Computer Vision*, 1:403–410, 2005.
- [141] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
- [142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Volume 1*, page 511, 2001.

- [143] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [144] T. Wang, Y. Rui, S. Hu, and J. Sun. Adaptive tree similarity learning for image retrieval. *Multimedia System*, 9(2):131–143, 2003.
- [145] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NIPS) 18*, pages 1473–1480. 2006.
- [146] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 988–995, June-2 July 2004.
- [147] C. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, Dec 1998.
- [148] C. K. I. Williams. Gaussian processes. In M. A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 466–470. The MIT Press, 2nd edition, 2002.
- [149] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems (NIPS) 8*, pages 514–520. The MIT Press, 1996.
- [150] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS) 15*. 2003.
- [151] L. Yang, R. Jin, R. Sukthankar, and Y. Liu. An efficient algorithm for local distance metric learning. In *AAAI Conference on Artificial Intelligence*, 2006.
- [152] J. Yin, D. Hu, and Z. Zhou. Noisy manifold learning using neighborhood smoothing embedding. *Pattern Recognition Letters*, 29(11):1613–1620, 2008.

-
- [153] J. You, W. Zhu, E. Pissaloux, and H. Cohen. Hierarchical image matching: a chamfer matching algorithm using interesting points. In *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, pages 70–75, Nov 1995.
- [154] S. X. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation by graph partitioning. *Advances in Neural Information Processing Systems 15*, pages 1383–1390, 2003.
- [155] S. X. Yu and J. Shi. Object-specific figure-ground segregation. *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 02:39, 2003.
- [156] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1491–1498, 2006.