MULTI-VIEW LEARNING FOR CLASSIFICATION

JINXING LI

PhD

The Hong Kong Polytechnic University

2019

The Hong Kong Polytechnic University
Department of Computing

Multi-view Learning for Classification

Jinxing Li

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

July 2018

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signature)

Jinxing Li
_____(Name of Student)

ABSTRACT

Multi-view learning methods often achieve performance improvement compared with single-view based methods in many applications. In this thesis, we mainly focus on multi-view learning and its application of classification. Four novel multi-view based methods are proposed.

Considering the similarity and diversity existing in various views, the Joint Similar and Specific Learning (JSSL) is first proposed to jointly and sparsely represent different views. In this method, each view is sparsely represented by the production of the view-specific dictionary and representation coefficient. Particularly, the coefficient is divided into two parts: the similar one and specific one. Thanks to this structure, we can not only exploit the correlation among various views, but also extract individual components, representing multiple views in an reasonable way. An efficient algorithm is proposed to alternatively update the similar and specific parts in JSSL.

However, in many real-world datasets, the distributions are complex and linearly representing the data, like JSSL, can not meet our requirement in many applications. To address this problem, we propose another multi-view learning method based on the Gaussian Process Latent Variable Model (GPLVM) to learn a set of non-linear and non-parametric mapping functions. GPLVM is a generative model which assumes there is a mapping function that can project a latent variable from a low-dimensional subspace to the observed high-dimensional space. In order to apply GPLVM to the multi-view data, we assume that there is a shared latent variable among various views and the conditionally independent mapping function with Gaussian Process (GP) prior can be learned for each view which can project the shared variable to its specific observed space. Here we denote this as the decoding part. Furthermore, we also introduce an additional mapping from the observed data to the shared space. Here

we denote this as the encoding part. Due to this auto-encoder framework, both non-linear projections from and to the observation are considered simultaneously. Furthermore, different from the conventional auto-encoder methods, we enforce the GP prior on the encoder, which remarkably reduces the number of estimated parameters and avoids the phenomenon of over-fitting. In this thesis, we call this method as Shared Auto-encoder Gaussian Process latent variable model (SAGP). To apply SAGP to the classification, a discriminative regularization is also embedded to encourage the latent variables belonging to the same class to be close while these belonging to different classes to be far. In the optimization process, the alternating direction method (ADM) and gradient decent technique are combined to update the encoder and decoder effectively.

Although SAGP achieves an improvement compared with JSSL, it makes a strong assumption on the kernel function, such as Radial Basis Function (RBF), which limits the capacity of real data modeling. In order to address this issue, we further propose another GPLVM based method, named Multi-Kernel Shared Gaussian Process latent variable model (MKSGP). Instead of designing a deterministic kernel function, like SAGP, multiple kernels are adaptively selected to construct the encoder and decoder. In MKSGP, weights corresponding to different kernel functions are automatically learned for different input datasets, which can represent the data in a more reasonable way. Furthermore, different from SAGP which uses the classifer offline, a hinge loss is embedded into the model to jointly learn the classification hyperplane, encouraging the latent variables belonging to different classes to be separated. In this way, the learned classifier would be more suitable for the input data. Being similar to SAGP, we optimize the encoder and decoder alternatively by using ADM and gradient decent method. Experimental results demonstrate that MKSGP is more powerful than SAGP.

JSSL, SAGP and MKSGP can well exploit the correlation among different views. However, in many real-world applications, each view often contains multiple features, which means this type of data has a hierarchical structure, while JSSL, SAGP and MKSGP do not

take these features with the multi-layer structure into account. In this thesis, we propose a probabilistic hierarchical model to tackle this issue and apply it to classification. Here, this approach is named as Hierarchical Multi-view Multi-feature Fusion (HMMF). In this model, the multiple features extracted from a common view are first fused as a latent variable. Concretely, we assume that the extracted multiple features are generated from this latent variable. To achieve this goal, projection matrices corresponding to a certain view are learned to map the latent variable from a shared space to the multiple observations. Since we prefer to applying this method to classification, the estimated latent variables corresponding to different views are influenced by their ground-truth label. In this way, this kind of multi-view and multi-feature data are fused hierarchically. In order to effectively solve the proposed method, the Expectation-Maximization (EM) algorithm is applied to estimate the parameters and latent variables.

To quantitatively evaluate the effectiveness of our four proposed multi-view methods, some experiments are conducted on several datasets, including synthetic dataset and real-world datasets. Experimental results on these datasets substantiate the effectiveness and superiority of our approaches as compared with some state-of-the-art methods.

Keywords: Multi-view, Sparse representation, Gaussian process, Latent variable model, Classification.

PUBLICATIONS

Journal Papers

1. **Jinxing Li**, Bob Zhang, Guangming Lu, David Zhang, "Visual Classification with Multi-Kernel Shared Gaussian Process Latent Variable Model," IEEE Transactions on Cybernetics, 2018.

2. **Jinxing Li**, Bob Zhang, Guangming Lu, David Zhang, "Generative Multi-view and Multi-feature Learning for Classification," Information Fusion, 2018.

3. **Jinxing Li**, Bob Zhang, David Zhang, "Shared Auto-encoder Gaussian Process Latent Variable Model for Visual Classification," IEEE Transactions on Neural Networks and Learning Systems, 2017.

4. **Jinxing Li**, Bob Zhang, Guangming Lu, Jane You, David Zhang, "Body Surface Feature-based Multi-modal Learning for Diabetes Mellitus detection," Information Sciences, 2018.

5. **Jinxing Li**, David Zhang, Yongcheng Li, Jian Wu, Bob Zhang, "Joint Similar and Specific Learning for Diabetes Mellitus and Impaired Glucose Regulation Detection," Information Sciences, 2017.

6. **Jinxing Li**, Bob Zhang, David Zhang, "Joint Discriminative and Collaborative Representation for Fatty Liver Disease Diagnosis," Expert Systems with Applications, 2017.

7. **Jinxing Li**, Bob Zhang, Guangming Lu, David Zhang, "Dual Asymmetric Deep Hashing Learning ." arXiv, 2018.

Conference Papers

1. **Jinxing Li**, Bob Zhang, Guangming Lu, and David Zhang, "Shared Linear Encoder-based Gaussian Process Latent Variable Model for Visual Classification," ACM Multimedia, 2018.

2. **Jinxing Li**, Hongwei Yong, Bob Zhang, Mu Li, Lei Zhang, and David Zhang, "A Probabilistic Hierarchical Model for Multi-view and Multi-feature Classification," AAAI, 2018.

ACKNOWLEDGEMENTS

First and foremost, I want to express my gratitude to my supervisors, Prof. David Zhang and Prof. Jane You, for their wonderful guidance, support and generosity. It is my great pleasure to be a student of them, who are always showing me the right direction, but allowing me to walk the path myself.

I would like to express my gratitude to Prof. Bob Zhang, Prof. Guangming Lu, and Prof. Wangmeng Zuo in my Ph.D study. Their valuable suggestions have greatly help me to do good research, and I am very thankful for their very constructive suggestions on my articles and research.

I want to express my gratitude to Hongwei Yong, Mu Li, Shuhang Gu, Kai Zhang, Linxiao Yang, Xixi Jia, Feng Li, and Keze Wang, for their assistance, for their encouragement and for the wonderful time in The Hong Kong Polytechnic University I have shared with them.

Finally, I want to thank my wife Xiaoxia Li and my parents for inspiring me to pursue this route. I want to thank them for their endless love, support, and encouragement.

TABLE OF CONTENTS

CERTIFICATE OF ORIGINALITY ii

ABSTRACT iii

PUBLICATIONS vi

ACKNOWLEDGEMENTS viii

LIST OF FIGURES xii

LIST OF TABLES xv

CHAPTER 1. INTRODUCTION 1
1.1 Main Problems 2
1.2 Main Contributions 7

CHAPTER 2. RELATED WORKS AND BACKGROUND KNOWLEDGE 10
2.1 Multi-view Learning 10
2.2 Sparse Representation Classifier 12
2.3 Gaussian Process Latent Variable Model 13
2.4 Shared Gaussian Process Latent Variable Model 14

CHAPTER 3. JOINT SIMILAR AND SPECIFIC LEARNING MODEL 16
3.1 Proposed Method 16
3.2 Optimization of JSSL 18
3.3 The Classification Rule of JSSL 21
3.4 Experimental Results 22
3.4.1 Dataset 22
3.4.2 Healthy Versus DM Classification 22
3.4.3 Healthy Versus IGR Classification 25
3.4.4 Parameter Analysis 28
3.5 Conclusion 28

CHAPTER 4. SHARED AUTO-ENCODER GAUSSIAN PROCESS LATENT VARIABLE MODEL 30
4.1 Proposed Method 30
4.1.1 Discriminative Prior 35
4.2 Optimization 36
4.3 Inference 38
4.4 Computational Complexity Analysis 39
4.5 Experiments 39
4.5.1 Dataset description 39
4.5.2 Experimental settings 40
4.5.3 Performance on the three datasets 42
4.5.4 Parameter analysis 49
4.6 Conclusion 50

CHAPTER 5. MULTI-KERNEL SHARED GAUSSIAN PROCESS LATENT VARIABLE MODEL 52
5.1 Problem Formulation 52
5.2 Priors for Classification 54
5.3 Back-Constraints 55
5.4 Optimization 56
5.4.1 update the latent variable $\mathbf{X}$ 57
5.4.2 update the latent variable $\mathbf{X}_g$ 57
5.4.3 update the kernel parameters γ and $\boldsymbol{\theta}$ 57
5.4.4 update the weight of different types of kernel functions 58
5.4.5 update the Lagrange multiplier $\mathbf{Z}$ and parameter μ 58
5.4.6 Inference 58
5.5 Computational Complexity Analysis 59
5.6 Experiments 60
5.6.1 Experimental Settings 60
5.6.2 Experimental Results 60
5.6.3 Parameter analysis 68
5.6.4 Time cost analysis 69
5.7 Conclusion 70

CHAPTER 6. HIERARCHICAL MULTI-VIEW MULTI-FEATURE FUSION 71
6.1 The Hierarchical probabilistic Model 71
6.2 Optimization 74
6.3 Prediction 77
6.4 Experiments 78
6.4.1 Datasets and Experimental Setting 78
6.4.2 Experimental Results on Three Datasets 80
6.5 Conclusion 84

CHAPTER 7. DISCUSSION, CONCLUSION AND FUTURE WORKS 85
7.1 Discussion 85
7.2 Conclusion 89
7.3 Future Works 90

LIST OF FIGURES

1.1 An example of the multi-view and multi-feature object. A person can be diagnosed through his or her tongue, face and sublingual vessel. Also, these modalities can be represented with different features. 6

3.1 The framework of JSSL. There are three parts in this strategy, which are constructing the dictionary for each view, representing the data sparsely, and jointly doing classification, respectively. For the first part, we directly use the training samples of each view to be the corresponding dictionary; for the second part, given a test sample, we sparsely represent its tongue, facial and sublingual features by using the constructed dictionary, where the representation coefficients are separated into two components: similar and specific ones; for the third part, we can get the estimated label based on the reconstruction error for each class. 17

3.2 The classification accuracy obtained by different methods based on the single view and multi-view data under the change of different training numbers in the Healthy Vs DM dataset. (a) The comparison between using multi-view data (JSSL) and the only tongue based feature. (b) The comparison between using multi-view data (JSSL) and the only face based feature. (c) The comparison between using multi-view data (JSSL) and the only sublingual image based feature. 23

3.3 ROC curves of different methods and different features for DM detection..... 25

3.4 The classification accuracy obtained by different methods based on the single view and multi-view data under the change of different training numbers in the Healthy Vs IGR dataset. (a) The comparison between using multi-view data (JSSL) and the only tongue based feature. (b) The comparison between using multi-view data (JSSL) and the only face based feature. (c) The comparison between using multi-view data (JSSL) and the only sublingual image based feature. 26

3.5 ROC curves of different methods and different features for IGR detection. ... 27

4.1 The framework of SAGP. For the multiple observations, we first assume that there exists a shared variable which can be projected to these observations by using different mapping functions. Additionally, we also assume that there is another projection from the multiple observations to this shared variable. .. 31

4.2 The graphic representation of the GPLVM, Shared-GPLVM (SGPLVM) and the proposed approach (SAGP). Here, the latent variable is denoted as $\mathbf{X}$, the observed single view is denoted as $\mathbf{Y}^s$, and the observed multi-view data for the encoder and decoder is denoted as $\mathbf{Y}_\gamma^v$ and $\mathbf{Y}^v$, respectively, where $v \in \{1, \cdots, V\}$. (a) In the GPLVM, a projection function $\boldsymbol{f}^s$ is learned to project the the latent variable $\mathbf{X}$ to the observed single view $\mathbf{Y}^s$. (b) In the SGPLVM, various projection functions $\{\boldsymbol{f}^v\}_{v=1}^V$ are learned to project the shared variable $\mathbf{X}$ to multiple observations $\{\mathbf{Y}^v\}_{v=1}^V$. (c) In the SAGP(SE), two types of mapping functions are learned. For the first type, we study a back constraint to project various observed data $\{\mathbf{Y}_\gamma^v\}_{v=1}^V$ to the shared variable $\mathbf{X}$ under the gaussian process priors $\{\boldsymbol{g}^v\}_{v=1}^V$; for the second type, being similar to SGPLVM, various projection functions $\{\boldsymbol{f}^v\}_{v=1}^V$ are learned to project the shared variable $\mathbf{X}$ to multiple observations $\{\mathbf{Y}^v\}_{v=1}^V$. (d) In the SAGP(IE), another two types of mapping functions are learned. The first one is to learn $\boldsymbol{g}^v$ to project $\mathbf{Y}_\gamma^v$ to $\mathbf{X}$ and the second type of projections is similar to that in SAGP(SE). 33

4.3 The classification accuracies for each class by using various single-view or multi-view based approaches on the Wiki Test-Image dataset. The percentages with bold font means our method obtains the best result, and the percentage with bold italic font means SAGP obtains the second best result in comparison to other approaches. 42

4.4 Confusion matrix of the category recognition results on the AWA Dataset. ... 46

4.5 Confusion matrix of the category recognition results computed by different methods on the NUS-WIDE-LITE dataset. The vertical axis shows the true labels and the horizontal axis shows the predicted labels. 49

4.6 SAGP(SE). The percentage of the overall and average accuracies under the change of the dimensionality of the latent space. (a) Wiki Text-Image dataset. (b) AWA dataset. (c) NUS-WIDE-LITE dataset. 51

4.7 SAGP(SE). The percentage of the overall and average accuracies under the change of the β. (a) Wiki Text-Image dataset. (b) AWA dataset. (c) NUS-WIDE-LITE dataset. 51

5.1 The Framework of MKSGP. Firstly, different types of features are obtained from a same object. The projection from these features to a shared latent variable is learned with multiple kernels. Simultaneously, projections from this latent variable to multiple observations are also learned with the multiple kernels. Particularly, to achieve the classification goal, a large margin prior is embedded into the latent variable. 53

5.2 The confusion matrix conducted on the AWA dataset when the dimensionality of $\mathbf{X}$ is set to 60. 62

5.3 The weight values of various kinds of kernels in multiple mappings on the AWA dataset. Here 'v1:Y2X' denotes the weights from the observed data to $\mathbf{X}$ in the first view. So do other symbols. 62

5.4 The average accuracy of selected 9 classes computed by different approaches. 64

5.5 The weight values of various kinds of kernels in multiple mappings on the NUS-WIDE-LITE dataset. 65

5.6 The accuracy obtained by different methods under different values of the dimensionality of the latent variable on the Biomedical dataset. 67

5.7 The weight values of various kinds of kernels in multiple mappings on the Biomedical dataset. 67

5.8 MKSGP. The percentage of the overall accuracy and average accuracy with the change of the value of τ on the three datasets. 69

5.9 MKSGP. The percentage of the overall accuracy and average accuracy with the change of the value of λ on the three datasets. 69

6.1 (a) The framework of the proposed method, where the number of the latent variables is equal to the number of observed views. (b) The probabilistic framework of the proposed method. 72

6.2 The comparison of the distributions between the original data and generated data. 80

6.3 The ROC curves obtained by different methods in DM detection. 82

7.1 The examples of incomplete multi-view data. 90

LIST OF TABLES

3.1 The averaged classification accuracy as well as the error bar (percentage) in 5 independent experiments for the Healthy Vs DM dataset. 24

3.2 The area under curve (AUC) for the different methods in DM detection. 25

3.3 The averaged classification accuracy as well as the error bar (percentage) in 5 independent experiments for the Healthy Vs IGR dataset. 26

3.4 The area under curve (AUC) for the different methods in IGR detection. 27

3.5 The accuracy and error bar obtained by JSSL on the DM dataset when the training number is set to 100 for each class. Note that $\lambda = 0.0001$. 28

3.6 The accuracy and error bar obtained by JSSL on the DM dataset when the training number is set to 100 for each class. Note that $\tau = 0.001$. 28

4.1 The experimental results (overall accuracies) on three datasets computed by DCCA and DCCAE with different structures. 41

4.2 The experimental results including both overall and average accuracies on the Wiki Text-Image dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 1 to 10. 43

4.3 The experimental results (overall accuracies) computed by DCCA and D-CCAE using dropout and sparsity regularization on the Wiki Text-Image dataset with different structures. 44

4.4 The experimental results including both overall and average accuracies on the AWA dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 40 to 130. 45

4.5 The experimental results (overall accuracies) computed by DCCA and D-CCAE using dropout and sparsity regularization on the AWA dataset with different structures. 47

4.6 The experimental results including both overall and average accuracies on the NUS-WIDE-LITE dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 1 to 30. 48

4.7 The experimental results (overall accuracies) computed by DCCA and D-CCAE using dropout and sparsity regularization on the NUS-WIDE-LITE dataset with different structures. 50

5.1 The overall and average accuracies on the AWA dataset computed by various methods under the change of the dimensionality of the latent variable from 40 to 130. 61

5.2 The overall and average accuracies on the NUS-WIDE-LITE dataset computed by various methods when q changes from 1 to 30. 63

5.3 The accuracy, sensitivity and specificity values computed by various approaches on the Biomedical dataset when the number of training instances is 30, 40, and 50, respectively. 66

5.4 The time cost at the testing stages on different datasets 69

6.1 The classification accuracies on the synthetic dataset obtained by HMMF. 79

6.2 The accuracy, sensitivity and specificity values obtained by different methods on the Biomedical dataset when the number of training samples is 40, 50, 60, and 70, respectively. Best results are highlighted in bold. 81

6.3 The area under curve (AUC) obtained by the different methods in DM detection. 83

6.4 The accuracy obtained by the different methods in the Wiki Text-Image dataset. 83

6.5 The accuracy obtained by HMMF with the change of the dimensionality of the latent variable. 84

7.1 The accuracy, sensitivity and specificity values obtained by different methods on Biomedical dataset when the number of training samples is 30, 40, and 50, respectively. Best results are highlighted in bold. 86

7.2 The overall and average accuracies on the Wiki Test-Image dataset obtained by JSSL, SAGP and MKSGP when the dimensionality of the latent variable changes from 1 to 10. 87

7.3 The overall and average accuracies on the AWA dataset computed by JSSL, SAGP and MKSGP when the dimensionality of the latent variable changes from 40 to 130. 88

7.4 The overall and average accuracies on the NUS-WIDE-LITE dataset computed by JSSL, SAGP and MKSGP when q changes from 1 to 30. 89

CHAPTER 1
INTRODUCTION

Classification as a classical field has always attracted much attention in pattern recognition. Many works [63] [54] on classification have been done. The popular classifiers for pattern recognition including K-Nearest Neighbor (KNN) [2] and Support Machine Vector (SVM) [10] are widely used due to their effectiveness and simplicity. With the rapid development of L_1 norm, the sparse representation also has been exploited in computer vision (e.g. Image restoration [49], face recognition [73]). In [73], Wright et al. proposed so-called sparse representation based classification (SRC) for face recognition. In SRC, the sample is represented with a dictionary consisting of training samples, and SRC gets a great improvement in face recognition. Considering the samples that are corrupted by noise or outliers, some robust models are presented [80], [53]. In [80], the method could regress a given sample robustly with regularized regression coefficients, while Nie et al. [53] used $L_{2,1}$ norm for the loss function which could robustly selected the feature for classification. Many other models with sparse representation have been presented in [19], [88], [78], [57], [79] and [77]. Recently, with the rapid development of the deep learning, some deep learning based methods [31] [24] [62] also achieve remarkable performance in classification.

Although many methods are presented for classification, most methods only consider the single view. However, there always exists different views from a common sample in practice. For instance, a single object (e.g. face) can be captured from various angles or an image can be represented with different features, such as Scale Invariant Feature Transform (SIFT) [47] and Histogram of Oriented Gradient (HOG) [71]. These kinds of data are often named as multi-view or multi-modal data. It has been proven that exploiting multi-view data would contribute to the performance improvement for classification. Thus, it is important and necessary to make a deep research on multi-view learning. In this thesis, we focus on

how to well exploit the multi-view data and improve the performance for classification.

1.1 Main Problems

The key problem for the multi-view learning is how to model the correlations across different modalities to overcome the limitation of a single view. A naive way is to concatenate different single views as a large one. Then the single view based classifiers such as KNN, SVM and SRC can be used to process these concatenated vectors. Although this strategy is easy to achieve, a common limitation is that it fails to exploit the latent structure or correlation across multiple modalities. Generally, a reasonable assumption is that different views are the different representations from a same object. Therefore, it is common to assume that there exists a latent variable across various views, and what we should do is to estimate a view-specific mapping [23, 40, 42, 68, 72] to project this latent variable from the common space to each observation. Although some multi-view learning methods have been proposed and achieve a satisfactory performance in some situations, there are still several problems that should be solved:

(1) How to quantify the similarity and diversity existing in different views. The most of existing multi-view learning methods assume that there is only common or similar part across different views and several metrics are used to evaluate the similarity. For instance, Yuan et al. [83] exploited the joint sparse representation (JSR) to jointly represent multiple modalities with the L_{21} norm. This norm encourages the zero and non-zero elements of the representation coefficients corresponding to various views on the same rows. By contrast, Yang et al. [81] enforced the coefficients of different views to be close to their mean to measure the similarity. However, these kinds of assumptions are too restrictive. For different views, there are also view-specific components. Jointly taking the similar and specific parts into consideration can represent the data in a more reasonable way. Thus, in this thesis we propose a novel method named Joint Similar and Specific Learning (JSSL) [44] to extract the similarity and diversity simultaneously. Due to the effectiveness and robustness of the sparse representation, the input is first represented sparsely. Different from existing methods

which only consider the common or similar part, the representation coefficient in JSSL is divided into two parts: the similar one and the specific one. To exploit the correlation across various views, we also enforce the similar parts to be close to their mean value. Thanks to this similar and specific structure, JSSL achieves an improvement on classification compared with the state-of-the-art methods. This method has been published in [44].

(2) How to represent the data in a non-linear way. In general, the distributions of the real-world datasets are complex and simple linear representation can not meet our requirement in many situations. To adapt the non-linearity existing in the data, some kernel based methods, such as kernel canonical correlation analysis (CCA) [1, 18, 33], randomized non-linear CCA [46] etc. were proposed. Instead of assuming a specific set of deterministic or parametric functions, the Gaussian Process Latent Variable Model (GPLVM) [15] [37] [14] [69] as a non-parametric and generative strategy was proposed to non-linearly and effectively fit the data. Lawrence et al. [37] presented GPLVM in 2004 to learn a low-dimensional latent variable in a non-linear way. Due to the Gaussian Process (GP) [56] prior embedded on the mapping function, the covariance function of GPLVM often has a powerful variety, which contributes to characterizing the real-world data with complicated distributions and diverge content [64]. Because of the powerful capacity of GPLVM, in this thesis, we propose a novel method to extract the correlation across various views by using GPLVM. Being similar to shared GPLVM (SGPLVM) [14], a shared variable is learned for multiple views by using the view-specific mapping functions with the Gaussian Process (GP) prior. However, our method is far different from GPLVM. Although the GPLVM as well as its various extensions have been well learned in recent years, few of them consider the mappings from and to the observations in a joint model. In other words, conventional GPLVM only assume that there is a projection from the low-dimensional latent space to the high dimensional observed space. However, at the testing stage, given a testing sample, what we need is the latent variable corresponding to the testing sample. In GPLVM, to get this latent variable, we should combine the test sample and the training samples together and use some strategies such as the gradient descent method to optimize the model. Obviously, this strategy is quite time consuming and can not meet our requirement in some real-time applications. Thus, it is

necessary to learn another projection or a back constraint regularization from the observation to the latent variable. In this way, we can easily get the latent variable corresponding to a testing sample. Note that [38] also proposed a back constraint regularization for GPLVM. However, this technique only learns a matrix to hold the regression between the observed space and the latent space. Differently, we propose a more powerful back projection from the observed space to the latent space in a non-linear way. Here we denote the projection from the observations to the latent variables as the encoder [28] [50] [51], and the projection from the latent variables to the observations as decoder. Thus, we name this method as Shared Auto-encoder Gaussian Process latent variable model (SAGP) [43] in this thesis.

Specifically, for the multiple views, we assume that there is a shared latent variable in the subspace among them. Then mappings are learned to project the shared latent space to the multiple views, and another back-projection from the multiple views to the shared latent variable is also considered. From the perspective of the probabilistic model, the Gaussian Process (GP) [56] is utilized to estimate the parameter and learn the active function in contrast to the traditional Neural Networks (NN), which is more efficient and reliable. From the perspective of the auto-encoder [51], mappings from the observations to the latent space are similar to projecting the input data into the latent variable (encoder), while mappings from the shared variable to the observed data are a reconstruction operation (decoder). Although this framework is similar to the well known auto-encoder [50] [51], there are differences between them. In the same case (one hidden layer), SAGP is more efficient to fit the input data as the covariance function of GPLVM often has a powerful variety, which contributes to characterizing the real-world data with complicated distributions and diverge content. Particularly, Gaussian Process is a non-parametric model, which can estimate the kernel parameters more flexibly. Furthermore, referring to the number of estimated parameters, SAGP only needs to estimate the kernel parameters, being far fewer than that in traditional auto-encoders methods. It would greatly avoid the phenomenon of over-fitting when the number of the training samples is relatively small. Additionally, in order to apply SAGP to the classification task, we also impose a discriminative regularization on the latent variables to enforce the latent variables belonging to the same category to be close while

these belonging to different categories to be far. This method was published in [43].

(3) How to effectively construct the covariance using the kernel function and exploit the class-based information. In SAGP, the GP prior is embedded to non-linearly represent the data. The Radial Basis Function (RBF) is often used to construct the covariance matrix. Generally speaking, the selection of the kernel function in the covariance matrix construction plays a key role in GPLVM. A reasonable selection of the kernel function would be beneficial for performance improvement. However, in different applications, the real-world data is complex and a certain kernel may be incapable of fitting the data. In order to address this problem, we further extend SAGP to a multi-kernel version. We combine multi-kernel learning with the proposed SAGP for classification. Instead of applying a certain kernel function to covariance matrix construction, the multiple kernels are used to adaptively and automatically build the covariance in both encoding and decoding parts. In this way, the proposed method can well adapt the complex and non-linear distribution of the input data by automatically updating the weights of different kernel functions. Here we name this proposed method as Multi-Kernel Shared Gaussian Process latent variable model (MKSGP) [41].

In SAGP, the discriminative regularization is imposed on the latent variables to encourage the latent variables belonging to the same class to be close while these belonging to different classes to be far. However, the limitation of this strategy is that it requires to retrain a certain classifier, such as KNN and SVM, off-line. In this way, our learned variables in the training phase may not fully meet the assumption of the classifier. A combination of latent variable learning and classifier learning is necessary. Therefore, we also impose a large margin prior on the latent subspace to jointly learn a hyperplane for each category to separate the latent variables belonging to different classes. In contrast to SAGP which training the latent variables and classifier at two separated steps, our extension can take the variable learning model and classifier learning model into account simultaneously, which makes these two models both be adaptive for the input data. This method has been publish in [41].

(4) How to model the multi-view and multi-feature data. Although JSSL, SAGP and MKSGP are proposed for multi-view data in sparse representation, non-linear, and multi-

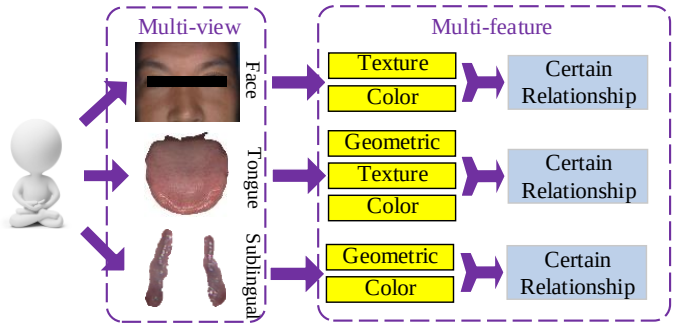


Figure 1.1. An example of the multi-view and multi-feature object. A person can be diagnosed through his or her tongue, face and sublingual vessel. Also, these modalities can be represented with different features.

kernel ways to get an outstanding performance, respectively, there are still some problems for us to tackle. One key difficulty is that except for collected multiple views from a single object, each view may also be represented with different features. Often, these multiple features from a view are fully beneficial for classification. Here, we name these data with multiple views and multiple features as the multi-view and multi-feature data. A typical example of multi-view and multi-feature is the person verification application. A person can be identified by using the fingerprint, palmprint, iris and face. Meanwhile, each view or modality can also be represented with various features, such as gabor and wavelet. Similarly, in the following chapters, we conduct experiments on a biomedical dataset. This dataset contains tongue, face and sublingual vessel modalities, while each modality includes different features, such as color, texture and geometric features, as shown in Fig.1.1. To the best of our knowledge, most existing methods as well as JSSL, SAGP and MKSGP build the models by only considering multiple views but ignoring the case that each view also can be represented by different features. A naive way to model this kind of data is to concatenate various features in each view as a single vector, then multi-view methods can be applied to process these vectors in multiple views. Although this strategy is easy to achieve, it has some limitations. One is that it may lose the correlation across different features in a view, while this correlation is valuable for classification. Another one is that it may encounter the

over-fitting if the dimension of concatenated vectors is large while the number of training data is relatively small. Therefore it is necessary for us to design a novel method to model the multi-view and multi-feature data to fully exploit the correlation among them.

In order to tackle this problem, a probabilistic and generative model [39] is proposed by modeling the multi-view and multi-feature data under a hierarchical structure. With the observed features from a view or modality, a shared and latent variable is learned as the fused feature. Additionally, since our model is constructed for classification, the learned variables associated with different views are assumed to be independently influenced by their ground-truth label. The Expectation Maximization (EM) [8] is introduced to optimize the proposed method. Specifically, a closed-form solution for each variable or parameter can be obtained. Here, we name this method as Hierarchical Multi-view Multi-feature Fusion (HMMF) and this is published in [39].

1.2 Main Contributions

The main contributions in this thesis are shown as follows:

- We first propose a joint sparse representation based method for multi-view data. Specifically, the input data in each view is first represented sparsely. Different from existing methods which only consider the common or similar part, we further take the specific parts into account. By considering the similarity and diversity simultaneously, the proposed method can better model the multiple views. To optimize the proposed method, an efficient algorithm is proposed to alternatively update the similar and specific parts.

- To tackle the non-linearity existing in many real-world dataset, the Shared Auto-encoder Gaussian Process latent variable model (SAGP) is proposed. Instead of assuming a specific mapping as done in many existing methods, SAGP introduces the GPLVM to non-linearly represent data in a generative and nonparametric way. In SAGP, a latent variable is shared among different views. Compared with GPLVM and its various extensions, SAGP jointly takes the projections from and to the observations

in to account. In other words, mapping functions are learned to project the variable from the shared subspace to the observed space. Meanwhile, a back constraint which maps the observations to the shared variable is also exploited. In this way, we are capable to easily obtain the latent variable when a testing sample comes. To apply SAGP to the classification, a discriminative regularization is embedded to encourage the latent variables belonging to the same class to be close while these belonging to different classes to be far.

- We further extend SAGP to a multi-kernel version. Although SAGP can non-linearly model the multi-view data, it only uses the RBF to construct the covariance matrix, while the selection of the kernel function in SAGP plays a key role and a certain kernel function may be incapable of modeling complex distributions in some real-world datasets. By contrast, the extended version Multi-Kernel Shared Gaussian Process latent variable model (MKSGP) combines the multi-kernel learning and SAGP into a joint model which can adaptively and automatically fit the data. Furthermore, different from SAGP which learns the latent variable and the classifier in two separated phases, MKSGP embeds a large margin prior into the model to jointly learn a hyperplane for each category to separate the latent variables belonging to different classes. In MKSGP, the classifier can be learned online, adapting the input data.

- We propose a novel method to model the multi-view and multi-feature data. In many applications, multiple views can be collected from a single object and each view can also be represented with different features. To our best knowledge, most existing multi-view learning methods can not be directly applied to this kind of multi-view and multi-feature data. To tackle this issue, a probabilistic generative model named Hierarchical Multi-view Multi-feature Fusion (HMMF) is proposed under a hierarchical structure. With the observed features from a view or modality, a shared and latent variable is learned as the fused feature. Additionally, since our model is constructed for classification, the learned variables associated with different views are assumed to be independently influenced by their ground-truth label. To optimize HMMF, EM al-

gorithm is exploited to obtain the closed-form solution for each variable or parameter.

Note that, some other works may also name different types of data from a same object as the multi-modal data and its corresponding works are called multi-modal learning. In fact, both multi-view learning and multi-modal learning are quite similar, which both aim to exploit the correlation among different views or modalities to improve the performance compared with single-view or single-modal based methods. Generally, multi-view learning is a wider definition since different views, modalities, features and angles are all belonging to multi-view data, while multi-modal data usually denotes different modalities such as the text and the image. Although they have slight difference, most of existing methods can be directly applied to both types of data.

The rest of this thesis is organized as follows. Some related works on multi-view learning and background knowledge including sparse representation, GPLVM and SGPLVM are briefly introduced in Chapter 2. In Chapter 3, 4, 5 and 6, we describe the proposed methods including JSSL, SAGP, MKSGP and HMMF, respectively. The thesis is concluded in Chapter 7.

CHAPTER 2

RELATED WORKS AND BACKGROUND KNOWLEDGE

In this chapter, we briefly introduce the related works about multi-view learning. Then some background knowledge including SRC, GPLVM and SGPLVM is further described.

2.1 Multi-view Learning

Various works based on the multi-feature learning have been proposed. Because of the effectiveness and robustness of the sparse representation, it has been widely extended to multi-view learning. Joint sparse representation (JSR) [82] was first introduced by Yuan et al. by introducing the $L_{2,1}$ norm to jointly represent the multiple features (MTJSRC). The $L_{2,1}$ norm makes the representation coefficients be sparse following the row direction which enables different features from the same sample to be sparsely represented by the training samples on the same positions. Additionally, a similar work based on the collaborative representation way proposed by Yang and Zhang et al. [81], which ensures the representation coefficients of different views to be close to their mean value (RCR). Considering the label information, a discriminant collaborative representation method (JDCR) was proposed in [42] for the multi-view data. In these three methods, the dictionary is fixed by using the training data. By contrast, Jing et al. [74] proposed a dictionary learning method to exploit the correlation across various features by imposing the low-rank prior on the learned dictionary. A sparse model was described in [6] (UMDL and SMDL) to learn a multi-modal dictionary which greatly exploit the correlation among different modalities. A convex subspace learning method (CSRL) was presented by Guo [22], in which a common space is

obtained by exploiting a group sparsity norm. To apply the multi-feature learning to semi-supervised field, a multi-feature shared learning approach was introduce by Zhang et al. [86] by introducing a new L_2 norm to achieve a global label consistency.

Except for the norm based methods, the Canonical Correlation Analysis (CCA) [25, 66, 84] has also attracted much attention. CCA aims to learn two mapping matrices by maximizing the correlation between two views. Besides, CCA is also extended to several methods by imposing some priors on the shared subspace, such as sparse CCA [4] and robust CCA [5, 52]. Specifically, Archambeau et al. [4] added the sparsity as a prior into CCA to reduce the influence of noise. [52] and [5] imposed the L_1 loss and a Student-t density on the CCA to remove outliers in the data. As CCA as well as its extensions are only adaptive for the two-view based application, Rupnik et al. proposed the multi-view CCA (MCCA) [59]. MCCA aims to estimate multiple projections to map each view onto a shared subspace, in which the sum of all pairwise correlations achieves the maximum. Despite of the CCA-based methods, the famous and supervised subspace learning-the Linear Discriminative analysis (LDA) is also extended to multi-view methods. Particularly, Kan et al. [29] described a Multi-view Discriminant Analysis (MvDA) to estimate a discriminative common space. Additionally, a generalized multi-view LDA (GMLDA) [61] was analyzed by learning a set of projections for each view. After projection, different categories are separated.

The ensemble learning [17][67][75] is another type of fusion strategies. Different from our proposed multi-view learning methods, the ensemble learning aims to fuse several simple classifiers together to get the final classifier with the stronger capability. Recently some ensemble learning works have been done for the multi-view data, which are called multi-view ensemble learning. Cuzzocrea et al. [12] proposed a multi-view ensemble learning method, which can not only exploit the multi-dimensional inputs, but also implement a stacking strategy. Additionally, the supervised feature set partitioning (SFSP) is introduced to the multi-view ensemble learning for the high dimensional data classification [32]. Furthermore, Xu et al. [76] proposed a novel method, named adaptive weighted fusion approach (AWFA), to combined several CRC classifiers together the for multi-view data classification.

Despite that various multi-view learning methods have been introduced and achieved a satisfactory performance, most of them assume that the collected data can be modeled linearly. In many applications, this assumption may be not reasonable, since non-linearity does exist in the real-world datasets. To represent the data non-linearly, some kernel based methods have been proposed. A typical example is the kernel CCA [1], which was proposed by mapping the data into a high-dimensional feature space. Taking the label information into account, a supervised approach denoted as Multi-view Fisher Discriminative Analysis (MFDA) [13] was introduced, which learns independent classifiers for multiple views. Also, another multi-view learning method was presented by exploiting a manifold regularization based on the sparse feature selection [48]. Due to the regularization, the manifold structure of the data can be well preserved and the performance on the scene image classification is greatly enhanced. Instead of assuming a specific set of deterministic or parametric functions, the Gaussian Process Latent Variable Model (GPLVM) [15] [37] [14] [69] as a non-parametric and generative strategy was proposed to non-linearly and effectively fit the data. GPLVM adds the Gaussian Process into the projection to smoothly learn a variable in a subspace. Thanks to Gaussian Process prior, GPLVM is more powerful in data representation compared with some dimensionality reduction methods. Besides, GPLVM has also been extended to some versions. For instance, a discriminative GPLVM (DGPLVM) was proposed by Urtasun et al. [69] through adding a discriminative prior. A shared GPLVM (SGPLVM) [14] was presented to process the multi-view data. SGPLVM was extended to a supervised version (DSGPLVM) [15] by exploiting a Laplacian matrix.

2.2 Sparse Representation Classifier

Since JSSL is based on the sparse representation, here we briefly introduce the sparse representation classifier (SRC). In SRC, the testing instance can be represented in a linear combination of the collected training instances. Particularly, the L_1-norm minimization is enforced on the representation coefficient to make it be sparse. In other words, a few of training samples are selected to linearly represent the testing sample.

Assume that matrix $\mathbf{D} = [\mathbf{D}_1, \mathbf{D}_2, \cdot, \cdot, \cdot, \mathbf{D}_J]$, which is usually named dictionary, consists of training samples, where J is the total number of categories, $\mathbf{D}_i \in \mathbb{R}^{m \times n_i}$ is the training samples belonging to the i-th class with the dimension m and number n_i. Here, we denote each column in $\mathbf{D}$ as the atom. A testing instance $\mathbf{y} \in \mathbb{R}^{m \times 1}$ can be represented through

$$\hat{\boldsymbol{\alpha}} = \arg\min \|\mathbf{y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \tag{2.1}$$

where λ is the penalty parameter, $\|\cdot\|_2^2$ is the L_2 norm and $\|\cdot\|_1$ is L_1 norm. $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}_1; \hat{\boldsymbol{\alpha}}_2; \cdot \cdot \cdot; \hat{\boldsymbol{\alpha}}_J]$ is the sparse coefficient, and $\hat{\boldsymbol{\alpha}}_i$ is the sparse coefficient corresponding to $\mathbf{D}_i$.

Suppose that the testing instance $\mathbf{y}$ belongs to the i-th class, then it can be well represented by the training samples from the i-th class. In other words, among its representation coefficients $\hat{\boldsymbol{\alpha}}$ over all the training samples, only coefficients in class i will be significant while others will be insignificant. Then we can get the prediction for the testing sample through following Eq.(2.2)

$$i^* = \arg\min_i \|\mathbf{y} - \mathbf{D}_i\hat{\boldsymbol{\alpha}}_i\|_2^2 \tag{2.2}$$

More information about the SRC can be found in [73].

2.3 Gaussian Process Latent Variable Model

For GPLVM, it aims to learn a latent variable $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times q}$ in a low dimensional subspace and a mapping function from $\mathbf{X}$ to the observed high-dimensional space $\mathbf{Y}^s = [\mathbf{y}_1^s, \cdots, \mathbf{y}_N^s]^T \in \mathbb{R}^{N \times D}$, where $q \ll D$, N is the number of samples, q and D are the dimensions of the latent variable and observation, respectively. The framework of Gaussian Process is exploited to estimate the latent space and its mapping function to the observed data. Mathematically, the raw data $\mathbf{y}^s$ can be represented as

$$\begin{aligned} \mathbf{y}^s &= \boldsymbol{f}^s(\mathbf{x}) + \varepsilon \\ \boldsymbol{f}^s(\mathbf{x}) &\sim \mathbf{GP}(\boldsymbol{\mu}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}')) \end{aligned} \tag{2.3}$$

where $\boldsymbol{f}^s$ is the projection function with the GP prior. In order to be simple, the mean variable $\boldsymbol{\mu}(\mathbf{x})$ is often set to be zero, and the element at the i-th row and j-th column of the covariance is defined by a Mercer kernel such as the radius basis function (RBF)

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \theta_1 \exp(-\frac{\theta_2}{2} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2) + \frac{\delta_{i,j}}{\theta_3} \tag{2.4}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the i-th and j-th latent variables, respectively; $\boldsymbol{\theta}^s = (\theta_1, \theta_2, \theta_3)$ denote the kernel parameters, governing the variance, the bandwidth of the RBF kernel, and the variance of additive noise, respectively; $\delta_{i,j}$ is the Kronecker delta function.

The conditional distribution of the $\mathbf{Y}^s$ w.r.t. the latent variables $\mathbf{X}$ can be obtained through integrating over $\boldsymbol{f}^s$,

$$\begin{aligned} p(\mathbf{Y}^s \mid \mathbf{X}, \boldsymbol{\theta}^s) &= \int p(\mathbf{Y}^s \mid \mathbf{X}, \boldsymbol{\theta}^s, \boldsymbol{f}^s) p(\boldsymbol{f}^s) d\boldsymbol{f}^s \\ &= \frac{1}{(2\pi)^{ND/2} |\mathbf{K}|^{D/2}} \exp(-\frac{1}{2} \mathrm{tr}(\mathbf{K}^{-1} \mathbf{Y}^s (\mathbf{Y}^s)^T)) \end{aligned} \tag{2.5}$$

According to the Bayesian theory, the posterior distribution of the latent space $\mathbf{X}$ can be computed as follows.

$$p(\mathbf{X}, \boldsymbol{\theta}^s \mid \mathbf{Y}^s) \propto p(\mathbf{Y}^s \mid \mathbf{X}, \boldsymbol{\theta}^s) p(\mathbf{X}) \tag{2.6}$$

where $p(\mathbf{X})$ is a certain prior such as the discriminative regularization. To estimate the variable $\mathbf{X}$ and parameters in the covariance matrix, the posterior or the negative log-posterior w.r.t. the latent variable $\mathbf{X}$ and parameter $\boldsymbol{\theta}^s$ should be maximized or minimized, shown as follows

$$\arg\min L = \frac{D}{2} \ln |\mathbf{K}| + \frac{1}{2} (\mathbf{K}^{-1} \mathbf{Y}^s (\mathbf{Y}^s)^T) + \frac{ND}{2} \ln 2\pi - \log(p(\mathbf{X})) \tag{2.7}$$

To optimize the Eq.(2.7), scalable gradient decent techniques can be used.

2.4 Shared Gaussian Process Latent Variable Model

However, GPLVM is only adaptive for the single-view based data. In [15] [64] [14], GPLVM was also extended to multi-view applications, which was named as shared GPLVM

(SGPLVM). In SGPLVM, various views share a common latent space and a projection function for each view is learned to map the shared variable to the observed data. Assume that the multi-view data $\{\mathbf{Y}^v \in \mathbb{R}^{N\times D^v}\}_v^V$ is collected from V views. Then the shared latent variable $\mathbf{X}$ among multiple views should be learned, instead of computing an independent one for each view as done in GPLVM. Furthermore, the distribution of the observed data $\mathbf{Y}^v$ given the shared $\mathbf{X}$ is conditionally independent across multiple views. Thus, we can factorize the joint likelihood w.r.t. observations as follows

$$p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}) = \prod_{v=1}^{V} p(\mathbf{Y}^v \mid \mathbf{X}, \boldsymbol{\theta}^v) \tag{2.8}$$

where $\boldsymbol{\theta}^v$ is the parameter of the kernel function for the v-th observation, being similar to that in the Eq.(2.4). From the Eq.(5.1), we can see that each input view $\mathbf{Y}^v$ is generated from the latent variable $\mathbf{X}$. Then the negative log-likelihood can be jointly minimized to estimate the shared variable $\mathbf{X}$, as shown in Eq.(2.9)

$$\arg\min L = \sum_{v=1}^{V} L^v = \sum_{v=1}^{V} \frac{D^v}{2} \ln |\mathbf{K}^v| + \frac{1}{2}\mathrm{tr}[(\mathbf{K}^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T] + \frac{ND^v}{2} \ln 2\pi \tag{2.9}$$

where $\mathbf{K}^v$ is the kernel or covariance matrix associated with the v-th view $\mathbf{Y}^v$. Besides, being similar to GPLVM, the Eq.(2.9) can be optimized by exploiting scalable gradient decent techniques.

CHAPTER 3

JOINT SIMILAR AND SPECIFIC LEARNING MODEL

Although various multi-view learning methods have been proposed, most of them only consider the common or similar parts across different views. In practice, there does exist view-specific components among them. Therefore, it is necessary to design a novel approach which takes both the similar and specific parts into account. In this chapter, the joint similar and specific learning (JSSL) is proposed to achieve this goal. This method has been published in [44].

3.1 Proposed Method

As mentioned above, different views from a same object may share the similarity. It is reasonable to assume that representation coefficients coded on their associated dictionaries of different modalities should be similar to each other across them. Here we use the biomedical data as an example. This dataset consists of tongue, face and sublingual vessel modalities. As shown in Fig.3.1, a testing sample, including tongue, facial and sublingual vectors, is a linear combination of the training instances; since all vectors belong to the same object, they will be well represented by the training samples from their associated category. Therefore, the position and non-zero elements of the representation coefficients in multiple views would be similar. To achieve the above goal, we utilize the following regularization [81] to measure the similarity among these three modalities.

$$\arg\min \sum_{k=1}^{K} \|\boldsymbol{\alpha}_k - \bar{\boldsymbol{\alpha}}\|_2^2 \tag{3.1}$$

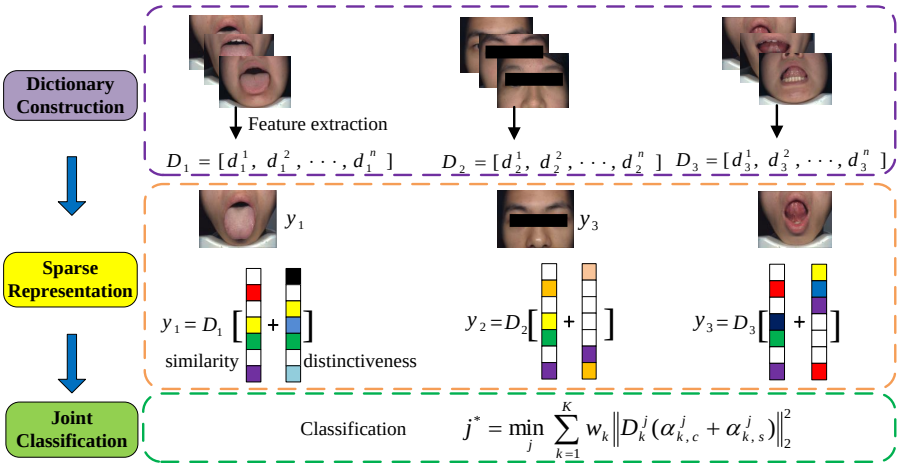


Figure 3.1. The framework of JSSL. There are three parts in this strategy, which are constructing the dictionary for each view, representing the data sparsely, and jointly doing classification, respectively. For the first part, we directly use the training samples of each view to be the corresponding dictionary; for the second part, given a test sample, we sparsely represent its tongue, facial and sublingual features by using the constructed dictionary, where the representation coefficients are separated into two components: similar and specific ones; for the third part, we can get the estimated label based on the reconstruction error for each class.

where $\boldsymbol{\alpha}_k$ is the representation coefficient corresponding to the k-th view and $\bar{\boldsymbol{\alpha}} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\alpha}_k$ is the mean variable of all $\boldsymbol{\alpha}_k$ (K means the number of views). It is easy to observe that Eq.(3.1) aims to reduce the variance of multiple coefficients $\boldsymbol{\alpha}_k$, encouraging them to be close to each one. However, as we have discussed, this assumption is too restrictive due to the existing of the specific components in different views. Therefore, it is necessary to not only utilize the similarity but also extract the diversity of each view. In this way, the balance between similarity and diversity will model the input instance more adaptive and accurate.

To tackle aforementioned issue, the representation coefficients $\boldsymbol{\alpha}_k$ are separated into two components: the similar one and the specific one. Mathematically, $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s$, where $\boldsymbol{\alpha}_k^c$ denotes the similarity, while $\boldsymbol{\alpha}_k^s$ denotes the diversity. Fig.3.1 shows the framework of JSSL. The formulation of this model is

$$\arg\min \sum_{k=1}^{K}\left\{\left\|\mathbf{y}_k - \mathbf{D}_k\left(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s\right)\right\|_2^2 + \tau\left\|\boldsymbol{\alpha}_k^c - \bar{\boldsymbol{\alpha}}^c\right\|_2^2\right\} + \sum_{k=1}^{K}\lambda\left(\left\|\boldsymbol{\alpha}_k^c\right\|_1 + \left\|\boldsymbol{\alpha}_k^s\right\|_1\right) \quad (3.2)$$

where $\mathbf{y}_k$ is the testing instance, $\mathbf{D}_k = [\mathbf{D}_k^1, \mathbf{D}_k^2, \cdots, \mathbf{D}_k^J]$ is training set belonging to the k-th view, and $\mathbf{D}_k^i \in \mathbb{R}^{m_k \times n_k^i}$ is the training set belonging to the k-th view of the i-th category with dimension m_k and sample number n_k^i; $\bar{\boldsymbol{\alpha}}^c = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\alpha}_k^c$ is the mean variable, and τ and λ are the non-negative parameters to trade off among different terms. From Eq.(3.2), it is easy to see that JSSL aims to extract the similar parts of each view through $\mathbf{x}_k^c$, while also preserves specific parts associated with different views through $\mathbf{x}_k^s$. Furthermore, since the testing instance can be linearly represented by training samples belonging to its own category, the L_1 norm is imposed on both $\boldsymbol{\alpha}_k^c$ and $\boldsymbol{\alpha}_k^s$ to keep the sparsity.

3.2 Optimization of JSSL

We alternatively update the similar coefficients $\boldsymbol{\alpha}_k^c$ and special coefficient $\boldsymbol{\alpha}_k^s$. For example, we update $\boldsymbol{\alpha}_k^c$ by fixing $\boldsymbol{\alpha}_k^s$, and vice versa.

Update $\boldsymbol{\alpha}_k^c$: By fixing $\boldsymbol{\alpha}_k^s$, the optimization of Eq.(3.2) with respect to $\boldsymbol{\alpha}_k^c$ equals to the following problem

$$\boldsymbol{\alpha}_k^c = \arg\min \left\|\mathbf{y}_k - \mathbf{D}_k\left(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s\right)\right\|_2^2 + \tau\left\|\boldsymbol{\alpha}_k^c - \bar{\boldsymbol{\alpha}}^c\right\|_2^2 + \lambda\left\|\boldsymbol{\alpha}_k^c\right\|_1 \quad (3.3)$$

we apply Augmented Lagrangian method (ALM) algorithm to modify Eq.(3.3).

Applying the ALM, the problem of (3.3) can be modified as follows.

$$\boldsymbol{\alpha}_k^c = \arg\min \|\mathbf{y}_k - \mathbf{D}_k\left(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s\right)\|_2^2 + \tau\left\|\boldsymbol{\alpha}_k^c - \bar{\boldsymbol{\alpha}}^c\right\|_2^2 + \lambda\left\|\boldsymbol{\alpha}_k^{c\prime}\right\|_1 + \frac{\mu}{2}\left\|\boldsymbol{\alpha}_k^c - \boldsymbol{\alpha}_k^{c\prime} + \frac{\mathbf{z}_k}{\mu}\right\|_2^2 \tag{3.4}$$

where $\boldsymbol{\alpha}_k^{c\prime}$ is the relaxed variable, $\mathbf{z}_k$ is the k-th lagrangian multiplier, and μ is the step value. Then we can optimize $\boldsymbol{\alpha}_k^c$ and $\boldsymbol{\alpha}_k^{c\prime}$ alternatively.

(a) Firstly, we fix $\boldsymbol{\alpha}_k^{c\prime}$ to get $\boldsymbol{\alpha}_k^c$

$$\boldsymbol{\alpha}_k^c = \arg\min \|\mathbf{y}_k - \mathbf{D}_k\left(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s\right)\|_2^2 + \tau\left\|\boldsymbol{\alpha}_k^c - \bar{\boldsymbol{\alpha}}^c\right\|_2^2 + \frac{\mu}{2}\left\|\boldsymbol{\alpha}_k^c - \boldsymbol{\alpha}_k^{c\prime} + \frac{\mathbf{z}_k}{\mu}\right\|_2^2 \tag{3.5}$$

Follow the Ref. [81], a closed-form solution of $\boldsymbol{\alpha}_k^c$ can be derived:

$$\boldsymbol{\alpha}_k^c = \boldsymbol{\alpha}_{0,k}^c + \frac{\tau}{K}\mathbf{P}_k\mathbf{Q}\sum_{\eta=1}^{K}\boldsymbol{\alpha}_{0,\eta}^c \tag{3.6}$$

where $\mathbf{P}_k = (\mathbf{D}_k^T\mathbf{D}_k + (\tau + \frac{\mu}{2})\mathbf{I})^{-1}$, $\boldsymbol{\alpha}_{0,k}^c = \mathbf{P}_k(\mathbf{D}_k^T(\mathbf{y}_k - \mathbf{D}_k\boldsymbol{\alpha}_k^s) + \frac{\mu}{2}\boldsymbol{\alpha}_k^{c'} - \frac{\mathbf{z}_k}{2})$, and $\mathbf{Q} = (\mathbf{I} - \frac{\tau}{K}\sum_{\eta=1}^{K}\mathbf{P}_\eta)^{-1}$.

(b) Secondly, after fixing $\boldsymbol{\alpha}_k^c$, the optimization solution of Eq.(3.4) can be reduced to Eq.(3.7) at the step of updating $\boldsymbol{\alpha}_k^{c\prime}$.

$$\boldsymbol{\alpha}_k^{c\prime} = \arg\min \lambda\left\|\boldsymbol{\alpha}_k^{c\prime}\right\|_1 + \frac{\mu}{2}\left\|\boldsymbol{\alpha}_k^c - \boldsymbol{\alpha}_k^{c\prime} + \frac{\mathbf{z}_k}{\mu}\right\|_2^2 \tag{3.7}$$

Then $\boldsymbol{\alpha}_k^{c\prime}$ could be derived by operating $\mathrm{Threshold}(\boldsymbol{\alpha}_k^c + \frac{\mathbf{z}_k}{\mu}, \frac{\lambda}{\mu})$. The operation of soft threshold is shown as follows.

$$\left[\mathbf{S}_{\lambda/\mu}(\boldsymbol{\beta})\right]_i = \begin{cases} 0 & |\boldsymbol{\beta}_j| \le \lambda/\mu \\ \boldsymbol{\beta}_i - \mathrm{sign}(\boldsymbol{\beta}_i)\lambda/\mu & \text{otherwise} \end{cases} \tag{3.8}$$

where $\boldsymbol{\beta}_i$ means the value of the i-th component of $\boldsymbol{\beta}$. After getting $\boldsymbol{\alpha}_k^c$ and $\boldsymbol{\alpha}_k^{c\prime}$, $\mathbf{z}_k$ and μ can be updated following $\mathbf{z}_k = \mathbf{z}_k + \mu(\boldsymbol{\alpha}_k^c - \boldsymbol{\alpha}_k^{c\prime})$ and $\mu = 1.2\mu$. In order to avoid μ being to large, we pre-fix the maximum of μ, following $\mu = \min(1.2\mu, 1000)$ and we set the initial value of μ as 0.01.

Algorithm 3.1 Algorithm of updating $\boldsymbol{\alpha}_k^s$ in JSSL

Input: $\sigma, \gamma = \lambda/2$, $\mathbf{y}_k$, $\mathbf{D}_k$, and $\boldsymbol{\alpha}_k^c$, $k = 1, \cdots, K$

Initialization: $\tilde{\boldsymbol{\alpha}}_k^{s(1)} = \mathbf{0}$ and $h = 1$,

1: **for** $k = 1, ..., K$ **do**

2: **while** not converged **do**

3: h=h+1

4: $\tilde{\boldsymbol{\alpha}}_k^{s(1)}$=$\mathbf{S}_{\gamma/\sigma}\left(\tilde{\boldsymbol{\alpha}}_k^{s(h-1)} - \frac{1}{\sigma}\nabla\mathbf{F}(\tilde{\boldsymbol{\alpha}}_k^{s(h-1)})\right)$

5: where $\nabla\mathbf{F}(\tilde{\boldsymbol{\alpha}}_k^{s(h-1)})$ is the derivative of the left of Eq. (3.9) $\|\mathbf{y}_k - \mathbf{D}_k(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s)\|_2^2$, and $\mathbf{S}_{\gamma/\sigma}$ is a soft threshold operator that defined in Eq. (3.8);

6: **end while**

7: **end for**

Output: $\boldsymbol{\alpha}_k^s = \tilde{\boldsymbol{\alpha}}_k^{s(h)}$, $k = 1, \cdots, K$

Update $\boldsymbol{\alpha}_k^s$: After acquiring $\boldsymbol{\alpha}_k^c$, the optimization of Eq.(3.2) can be reformulated to Eq.(3.9) at the step of updating $\boldsymbol{\alpha}_k^s$.

$$\boldsymbol{\alpha}_k^s = \arg\min \|\mathbf{y}_k - \mathbf{D}_k(\boldsymbol{\alpha}_k^c + \boldsymbol{\alpha}_k^s)\|_2^2 + \lambda \|\boldsymbol{\alpha}_k^s\|_1 \tag{3.9}$$

In fact, there are many methods to tackle the problem (3.9). For example, both ALM and Iterative Projection Method (IPM) [58] could deal with it. In this thesis, we use IPM to address Eq.(3.9), as described in Algorithm 3.1.

Algorithm 3.2 summarizes the details of the optimization in JSSL. The values of parameters λ and γ are selected through the cross validation.

Computational Complexity Analysis: To simplify the description of the computational complexity per iteration, here we firstly let $M = \max\{m_k\}_{m=1}^K$ and N be the number of training data. The main complexity is in Eq. (3.6) and Algorithm 3.1. The complexity is $O(N^2M + N^3)$ for $\mathbf{P}_k$ calculation. Similarly, the complexity is $O(N^2 + 2MN)$ and $O(N^3)$ for $\boldsymbol{\alpha}_{0,k}^c$ and $\mathbf{Q}$ computation, respectively. Thus, the complexity of Eq. (3.6) for all views is about $O(K(N^3 + N^2M))$. The complexity of Algorithm 3.1 is $O(2KMNt_1)$, where t_1

Algorithm 3.2 Joint Similar and Special Learning (JSSL)

Input: λ, τ, $\mathbf{y}_k$, $\mathbf{D}_k$, $k = 1, \cdots, K$

Initialization: $\boldsymbol{\alpha}_k^c = \mathbf{0}$, $\boldsymbol{\alpha}_k^s = \mathbf{0}$, $\boldsymbol{z}_k = \mathbf{0}$

1: **while** not converged **do**

2: **Update coefficients** $\boldsymbol{\alpha}_k^c$: fix $\boldsymbol{\alpha}_k^s$

3: (a) compute $\boldsymbol{\alpha}_k^c$ following Eq. (3.6)

4: (b) compute $\boldsymbol{\alpha}_k^{c\prime}$ following Eq. (3.7)

5: (c) $\mathbf{z}_k = \mathbf{z}_k + \mu(\boldsymbol{\alpha}_k^c - \boldsymbol{\alpha}_k^{c\prime})$

6: **Update coefficients** $\boldsymbol{\alpha}_k^s$: fix $\boldsymbol{\alpha}_k^c$, and solve $\boldsymbol{\alpha}_k^s$ following **Algorithm 3.1**

7: **end while**

Output: $\boldsymbol{\alpha}_k^c$ and $\boldsymbol{\alpha}_k^s$ $k = 1, \cdots, K$

is the number of iteration in Algorithm 3.1. In conclusion, the total complexity for JSSL is about $O(K(N^3 + N^2M + MNt_1)t_2)$, where t_2 is the number of iteration in Algorithm 3.2.

3.3 The Classification Rule of JSSL

After obtaining the representation coefficients, the decision is ruled in favor of the class with total lowest reconstruction residual over all K vectors.

$$j^* = \arg\min \sum_{k=1}^{K} w_k \left\| \mathbf{y}_k - \mathbf{D}_{k,j}(\boldsymbol{\alpha}_{k,j}^c + \boldsymbol{\alpha}_{k,j}^s) \right\|_2^2 \tag{3.10}$$

where $\mathbf{D}_{k,j}$, $\boldsymbol{\alpha}_{k,j}^c$ and $\boldsymbol{\alpha}_{k,j}^s$ are the elements of the dictionary $\mathbf{D}_k$, the similar coefficient $\boldsymbol{\alpha}_k^c$ and the specific coefficient $\boldsymbol{\alpha}_k^s$ of j-th category, respectively; w_k is the weight value corresponding to the k-th vector, which could be computed by using the method described in [82].

3.4 Experimental Results

In this section, two kinds of experiments are conducted. One is Healthy versus Diabetes Mellitus (DM) classification. Another is Healthy versus Impaired Glucose Regulation (IGR) classification. In both experiments, some single-view based methods, including KNN [11], SVM [26], [16], SRC [73], GSRC (group sprase) [7] are used as the comparison approaches. Since these methods can not be directly applied to multi-view data, we also concatenate different views containing the tongue, facial and sublingual features as a single one, symbolled as 'combination'. To quantatively demonstrate the superiority of the proposed method, three multi-view learning methods called multi-task joint sparse representation classifier (MTJSRC) [82], relax collaborative representation (RCR) [81] and adaptive weighted fusion approach (AWFA) [76] are also considered as the comparison approaches.

3.4.1 Dataset

This Biomedical dataset consists of 504 instances: 192 Healthy instances, 198 DM instances and 114 IGR instances. Each sample is comprised of three kinds of images: tongue, face and sublingual vessel, respectively. All images were collceted at the Guangdong Provincial TCM Hospital, Guangdong, China, from the early 2014 to the late 2015. Healthy samples were verified through a blood test and other examination according to the standard indicators which are set by the Guangdong Provincial TCM Hospital. DM or IGR samples are decided by using the FPG test.

3.4.2 Healthy Versus DM Classification

In this experiment, the number of training samples from 30 to 100 is randomly selected with 5 independent times, and the remaining instances are regarded as the testing samples. Fig.3.2 shows the experimental results computed by JSSL, KNN, SVM, SRC and GSRC. Note that Fig.3.2 only shows the averaged results of 5 independent experiments. It is easy to see that the JSSL consistently gains better performance between DM patients and

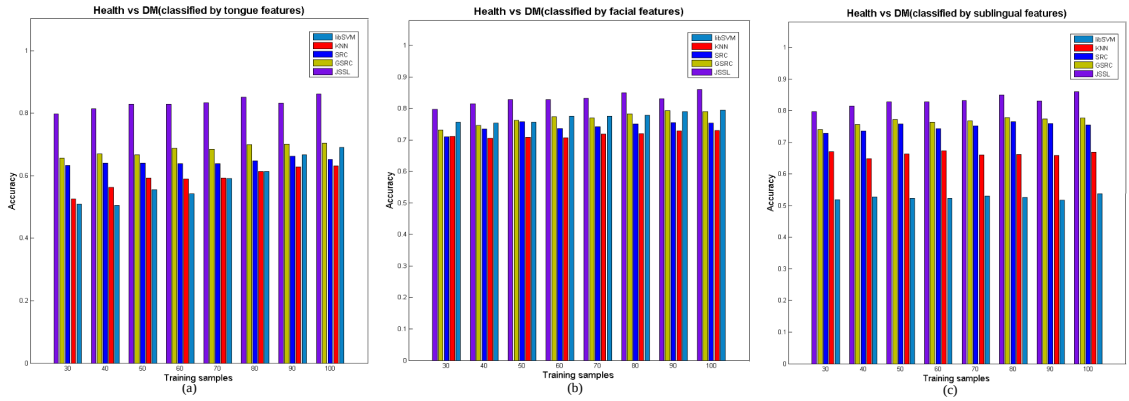


Figure 3.2. The classification accuracy obtained by different methods based on the single view and multi-view data under the change of different training numbers in the Healthy Vs DM dataset. (a) The comparison between using multi-view data (JSSL) and the only tongue based feature. (b) The comparison between using multi-view data (JSSL) and the only face based feature. (c) The comparison between using multi-view data (JSSL) and the only sublingual image based feature.

healthy controls. The accuracies obtained by JSSL along with the error bar and several state-of-the-art results directly from the single task are tabulated in Tab.3.1. JSSL performs much better than single-view based strategies not only on the average accuracy, but also on the error bar.

Additionally, the results by feature concatenation methods (concatenate the tongue, facial and sublingual features as a single vector) and another three fusion methods (MTJSR-C, RCR and AWFA) are also listed in Tab.3.1. As we can see, the results gained by JSSL are the best in most cases. The K-NN, SVM, SRC and GSRC combination based methods are much inferior to our model. Note that, the results obtained by simply concatenating different vectors may even suffer a large performance drop, such as K-NN and SRC. Meanwhile, in contrast to multi-view based methods: MTJSRC, RCR and AWFA, JSSL is also competitive. Fig.3.3 further plots the ROC curves obtained by various methods when the number of training samples is 100, as well as their associated AUC values listed in Tab.3.2. From Fig.3.3 and Tab.3.2, we can see that the area covered by the curve computed by JSSL is obviously

Table 3.1. The averaged classification accuracy as well as the error bar (percentage) in 5 independent experiments for the Healthy Vs DM dataset.

	Training samples							
Methods	30	40	50	60	70	80	90	100
JSSL	**79.82**±1.92	**81.45**±1.50	**82.82**±1.46	**82.88**±2.15	**83.27**±1.55	**85.06**±0.87	83.17±2.15	**86.07**±1.07
K-NN (face)	71.15±1.89	70.55±3.35	70.86±2.94	70.74±2.18	71.83±1.89	71.99±3.26	72.89±2.21	72.98±1.8
libSVM (face)	75.62±2.03	75.43±2.24	75.67±2.51	77.53±1.47	77.53±2.49	77.92±3.05	79.10±1.70	79.48±3.14
SRC (face)	71.00±3.22	73.51±3.35	75.84±1.67	73.62±3.79	74.14±2.90	75.11±3.84	75.59±2.60	75.39±2.37
GSRC (face)	73.14±1.51	74.59±2.74	76.32±2.26	77.45±2.19	76.93±1.82	78.31±3.85	79.34±2.11	79.00±2.39
K-NN (combine)	68.58 ±2.48	70.35±2.36	68.18 ±1.43	71.96 ±2.24	72.03 ±2.24	71.86 ±2.69	72.99 ± 1.21	74.56 ±1.84
libSVM (combine)	77.34 ±2.91	76.08 ±2.21	76.84 ±7.20	78.60 ±1.43	76.33 ±3.67	79.05 ±1.58	79.91 ±2.11	81.15 ±1.78
SRC (combine)	61.33 ± 8.96	65.40 ±6.43	66.32±7.59	68.63 ±9.35	75.14 ±4.72	74.03 ±2.31	77.54±4.17	74.35 ±7.70
GSRC (combine)	79.03 ± 1.72	80.51 ±2.80	79.11 ±4.05	77.05 ±6.53	80.39 ± 1.83	79.83 ±2.46	82.09 ±2.36	83.35 ±2.55
MTJSRC	77.76 ± 2.93	79.04±2.56	80.48 ±1.18	80.96±2.00	80.08 ±2.16	82.51 ±1.49	81.04 ±2.17	83.14 ±1.19
RCR	77.81 ± 2.49	78.64 ±2.10	80.51±2.60	80.91 ±3.33	82.53 ±2.03	83.36 ±1.12	**83.67**±1.82	83.21 ±2.76
AWFA	79.64±1.58	79.81± 1.15	81.17 ± 1.51	81.40 ± 2.46	81.59 ± 3.64	82.51 ± 0.84	82.09 ± 1.55	83.33 ± 2.70

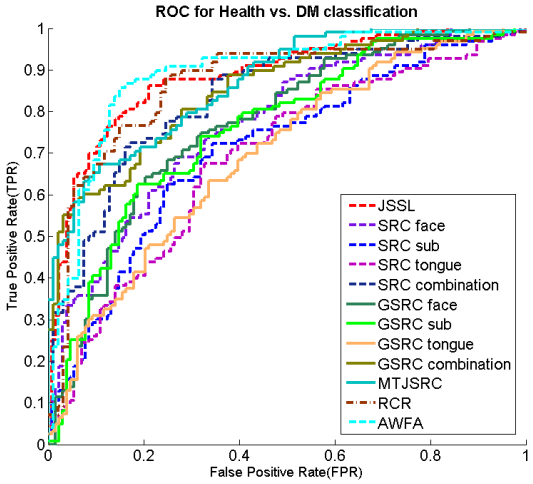


Figure 3.3. ROC curves of different methods and different features for DM detection.

Table 3.2. The area under curve (AUC) for the different methods in DM detection.

Methods	AUC	Methods	AUC	Methods	AUC
JSSL	0.8842	SRC(face)	0.7696	GSRC(face)	0.7686
SRC(sublingual)	0.7091	GSRC(sublingual)	0.7588	SRC(tongue)	0.6885
GSRC(tongue)	0.6944	SRC(combination)	0.8328	GSRC(combination)	0.8512
MTJSRC	0.8670	RCR	0.8633	AWFA	**0.8887**

larger than other curves except of that obtained by AWFA.

3.4.3 Healthy Versus IGR Classification

We then apply JSSL to IGR detection. Being similar to the setting in the DM experiment, we randomly select from 30 to 70 samples from each category with 5 times as the training set, and the remaining samples are used for testing. Fig.3.4 shows the averaged experimental results compared with KNN, SVM, SRC and GSRC. It is easy to see that our presented strategy is superior to these methods. Tab.3.3 tabulates the classification accuracy along with the error bar. We can see that our proposed method gains an obvious enhancement in comparison to K-NN, SRC and GSRC based combination methods. Similarly, a simple

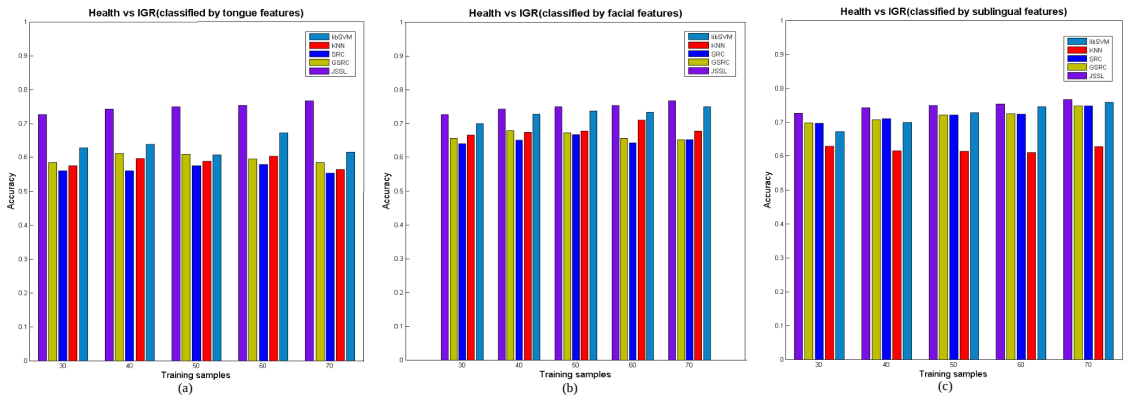


Figure 3.4. The classification accuracy obtained by different methods based on the single view and multi-view data under the change of different training numbers in the Healthy Vs IGR dataset. (a) The comparison between using multi-view data (JSSL) and the only tongue based feature. (b) The comparison between using multi-view data (JSSL) and the only face based feature. (c) The comparison between using multi-view data (JSSL) and the only sublingual image based feature.

Table 3.3. The averaged classification accuracy as well as the error bar (percentage) in 5 independent experiments for the Healthy Vs IGR dataset.

	Training samples				
Methods	30	40	50	60	70
JSSL	72.63±1.64	**74.23**±2.46	74.88±1.32	**75.37**±1.85	**76.68**±3.45
K-NN (sub)	62.87±2.85	61.54±2.24	61.35±3.49	60.96±3.05	62.81±3.63
libSVM (sub)	67.21±0.93	69.95±1.50	72.80±1.17	74.49±1.52	75.87±1.51
SRC (sub)	69.68±2.96	71.01±4.32	72.08±2.08	72.41±4.19	74.85±3.79
GSRC (sub)	69.80±4.62	70.71±2.75	72.13±3.25	72.57±3.94	74.79±3.21
K-NN (combine)	66.64 ± 3.35	66.96 ±8.78	65.41 ±3.99	68.02 ±3.58	66.59 ±4.68
libSVM (combine)	**72.87** ±1.69	73.30 ±2.80	72.37 ±5.63	72.83 ± 4.38	75.81 ±1.00
SRC (combine)	59.43 ±4.33	61.76 ±7.59	62.51 ±5.77	57.43 ±6.55	67.19 ±8.24
GSRC (combine)	69.64 ±0.95	71.89 ±3.52	72.66 ±5.38	71.76 ±3.90	73.65 ± 4.96
MTJSRC	71.74 ±4.84	70.57 ±2.60	73.8 ±2.54	74.76 ± 3.17	74.13 ± 4.70
RCR	72.53 ±4.55	71.81 ±3.55	**74.94** ±2.34	73.37± 2.78	73.14 ±1.63
AWFA	65.02 ± 5.34	66.43 ± 1.93	65.89 ± 3.70	67.06 ± 3.70	67.78 ± 4.57

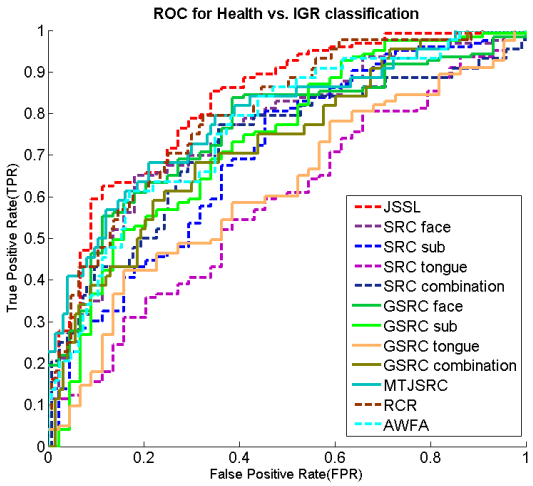


Figure 3.5. ROC curves of different methods and different features for IGR detection.

Table 3.4. The area under curve (AUC) for the different methods in IGR detection.

Methods	AUC	Methods	AUC	Methods	AUC
JSSL	**0.8278**	SRC(face)	0.7579	GSRC(face)	0.7596
SRC(sublingual)	0.6984	GSRC(sublingual)	0.7321	SRC(tongue)	0.5850
GSRC(tongue)	0.6155	SRC(combination)	0.7203	GSRC(combination)	0.7243
MTJSRC	0.7870	RCR	0.7982	AWFA	0.7646

concatenation may result in a large performance drop, e.g. SRC. In contrast to MTJSRC, R-CR and AWFA, JSSL has a noticeable increase in average accuracy except when the number of training samples is 50.

Fig.3.5 plots the ROC curves when the number of training samples is 70, followed by their associated AUC values in Tab.3.4. Compared with these methods, our proposed method JSSL is also outstanding.

Table 3.5. The accuracy and error bar obtained by JSSL on the DM dataset when the training number is set to 100 for each class. Note that $\lambda = 0.0001$.

	num=100						
τ	0	0.00001	0.0001	0.001	0.01	0.1	1
Accuracy	84.71%	84.82%	84.82%	86.07%	84.61%	83.46%	83.25%
Error Bar	1.36%	1.23%	1.23%	1.07%	1.37%	1.76%	1.39%

Table 3.6. The accuracy and error bar obtained by JSSL on the DM dataset when the training number is set to 100 for each class. Note that $\tau = 0.001$.

	num=100						
λ	0	0.00001	0.0001	0.001	0.01	0.1	1
Accuracy	84.08%	85.03%	86.07%	84.40%	79.43%	78.64%	73.61%
Error Bar	1.21%	1.37%	1.07%	0.23%	1.32%	2.55%	1.55%

3.4.4 Parameter Analysis

In JSSL, the τ and λ should be tuned. Here we conduct an experiment on the DM dataset to analyze the influence of these two parameters, as shown in Tab.3.5 and Tab.3.6. Note that, the training number for each category is set to 100. As we can see, both τ and λ have an influence on the classification performance, demonstrating the importance of the similar and specific components. From Tab.3.6 it is also easy to observe that λ is sensitive to the performance, which encourages us to tune it carefully through the cross-validation.

3.5 Conclusion

In this chapter, a novel multi-view learning method is proposed to extract the similarity and diversity across multiple views. The similar parts reflect their correlation while the specific parts indicate their difference. Thanks for this structure, the data can be represented in a more reasonable way. To optimize the proposed method, an efficient algorithm is designed to alternatively update the similar and specific components. We apply our proposed

approach to the DM and IGR detection. According to the experimental results, our method is superior to several state-of-the-art strategies.

CHAPTER 4

SHARED AUTO-ENCODER GAUSSIAN PROCESS LATENT VARIABLE MODEL

Although JSSL achieves better performance compared with some existing multi-view learning methods, it only represents the data linearly. However, there exists non-linearity in the real-world data and linearly modeling can not meet our requirement in many applications. To tackle this issue and instead of assuming a set of deterministic functions, in this chapter, a novel method is proposed based on the GPLVM, which non-linearly and smoothly model the data. To obtain the latent variable corresponding to the test sample in a simple way, another projection from the observations to the shared latent variable is considered. Furthermore, we also embedded a discriminative prior into the model to fully exploit the semantic information. This method was published in [43].

4.1 Proposed Method

Fig.4.1 shows the main framework of SAGP. The main purpose of the proposed method is to estimate a latent variable shared across multi-view data. Specifically, SAGP assumes that the observed views $\mathbf{y} = \{\mathbf{y}^v\}_{v=1}^V$ are the projections from a shared latent variable $\mathbf{x}$ in a subspace, where V is the number of views. To make the projection non-linear and smooth, GPLVM is applied to our method because of its powerful capability. Being similar to SGPLVM [14], the proposed method is conditionally dependent across various views w.r.t. the latent variable $\mathbf{X}$. Thus, the likelihood of the observed data $\{\mathbf{Y}^v\}_v^V$ given $\mathbf{X}$ is factorized as follows

$$p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V) = \prod_{i=1}^{V} p(\mathbf{Y}^v \mid \mathbf{X}, \boldsymbol{\theta}^v) \tag{4.1}$$

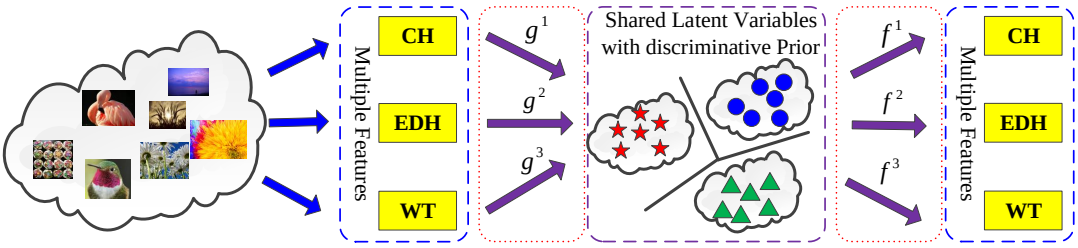


Figure 4.1. The framework of SAGP. For the multiple observations, we first assume that there exists a shared variable which can be projected to these observations by using different mapping functions. Additionally, we also assume that there is another projection from the multiple observations to this shared variable.

where $\mathbf{Y}^v = [\mathbf{y}_1^v, \cdots, \mathbf{y}_N^v]^T \in \mathcal{R}^{N \times D^v}$ is the set data of the v-th view, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T \in \mathcal{R}^{N \times q}$ is the set of N latent variables, and $\boldsymbol{\theta}^v = \{\theta_1^v, \theta_2^v, \theta_3^v\}$ are its corresponding kernel parameters.

Differently, in contrast the existing works, the proposed method defines another projection from the observed data $\mathbf{Y}^v$ to the corresponding latent variable $\mathbf{X}$ and then the observed inputs are reconstructed by the shared latent variables. In particular, using the gaussian process mapping function, $\mathbf{X}$ and $\mathbf{Y}^v$ can be represented as

$$\begin{aligned} \mathbf{x} &= \boldsymbol{g}^v(\mathbf{y}^v, \boldsymbol{\gamma}^v) + \varepsilon_1, \\ \mathbf{y}^v &= \boldsymbol{f}^v(\mathbf{x}, \boldsymbol{\theta}^v) + \varepsilon_2 \end{aligned} \tag{4.2}$$

where the function $\boldsymbol{g} = \{\boldsymbol{g}^v\}_v^V$ and $\boldsymbol{f} = \{\boldsymbol{f}^v\}_v^V$ are two groups of Gaussian Process with kernel parameters $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}^v\}_v^V$ ($\boldsymbol{\gamma}^v = \{\gamma_1^v, \gamma_2^v, \gamma_3^v\}$) and $\boldsymbol{\theta} = \{\boldsymbol{\theta}^v\}_v^V$, respectively, and ε_1 and ε_2 are the independent gaussian noises, whose mean and covariance are zero and $\delta_1^2\mathbf{I}$ and $\delta_2^2\mathbf{I}$. Through the auto-encoder view, it is easy to see that the former part in Eq.(4.2) is an encoder, while the later part is a decoder. Additionally, based on the GPs, we can get the marginal likelihood of the shared latent variable and the observations w.r.t. their

corresponding mapping functions,

$$
\begin{aligned}
p(\mathbf{X} \mid \boldsymbol{g}^v, \mathbf{Y}^v, \boldsymbol{\gamma}) &= \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{g}^v(\mathbf{y}_n^v), \delta_1^2 \mathbf{I}\right) \\
p(\mathbf{Y}^v \mid \boldsymbol{f}^v, \mathbf{X}, \boldsymbol{\theta}) &= \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{y}_n^v \mid \boldsymbol{f}^v(\mathbf{x}_n), \boldsymbol{\delta}_2^2 \mathbf{I}\right)
\end{aligned}
\tag{4.3}
$$

and the projection functions follow the gaussian distribution

$$
\begin{aligned}
p(\boldsymbol{g}^v) &\sim \mathcal{N}(\boldsymbol{g}^v \mid 0, \mathbf{K}_Y^v), \\
p(\boldsymbol{f}^v) &\sim \mathcal{N}(\boldsymbol{f}^v \mid 0, \mathbf{K}_X^v)
\end{aligned}
\tag{4.4}
$$

where the RBF kernel is applied to defined the kernel functions of the encoder and decoder

$$
\begin{aligned}
\mathbf{k}_Y^v(\mathbf{y}_i^v, \mathbf{y}_j^v) &= \gamma_1^v \exp(-\frac{\gamma_2^v}{2} \left\|\mathbf{y}_i^v - \mathbf{y}_j^v\right\|_2^2) + \frac{\delta_{i,j}^y}{\gamma_3^v} \\
\mathbf{k}_X^v(\mathbf{x}_i, \mathbf{x}_j) &= \theta_1^v \exp(-\frac{\theta_2^v}{2} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2) + \frac{\delta_{i,j}^x}{\theta_3^v}
\end{aligned}
\tag{4.5}
$$

In this thesis, two frameworks are proposed to achieve the encoding part. A single inverse projection from all the views to the latent subspace is first designed as shown in Fig.4.2(c). Theoretically, the covariance matrix of $p(\mathbf{X} \mid \{\mathbf{Y}^v\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V)$ is set as the sum of the kernel matrices defined on different views $\mathbf{Y}^v$: $\sum_v \mathbf{K}_Y^v$. Additionally, we also learn an independent projection from each view to the common subspaceas shown in Fig.4.2(d). The former one is named as a shared encoder (SE), and the latter one is referred as an independent encoder (IE). For the IE process, $p(\mathbf{X} \mid \{\mathbf{Y}^v\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V)$ is replaced by $p(\mathbf{X} \mid \mathbf{Y}^v, \boldsymbol{\gamma}^v)$. The reason why we design the second back projection is to adapt for the testing samples which miss some certain views. In other words, if all views are available for a testing sample, both inverse mappings can be adopted. However, it is also possible that a testing instance may miss one or more than one modalities in applications. At this situation, we can still get the latent variable by using the second back mapping approach. In order to be simple and without loss generality, only the SE-based approach is described in the following sections. For the second strategy, we only need to replace $\sum_v \mathbf{K}_Y^v$ with $\mathbf{K}_Y^v$. Compared with [38] which only learns the back-projection by estimating a mapping matrix that holds the parameters for the regression, SAGP achieves the projection with the GP prior from the observed space to the shared manifold. For the sake of the non-parametric and non-linear advantages of the GP

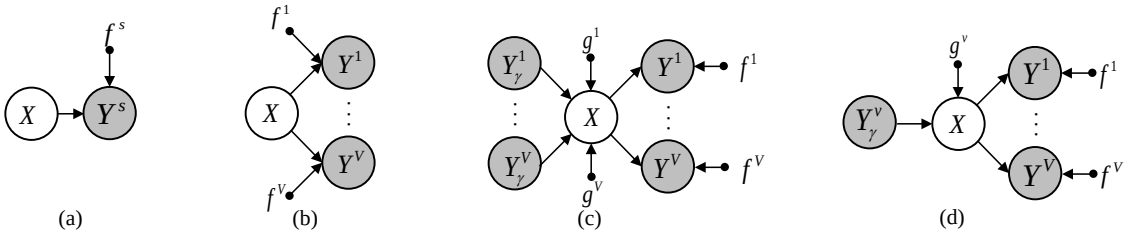


Figure 4.2. The graphic representation of the GPLVM, Shared-GPLVM (SGPLVM) and the proposed approach (SAGP). Here, the latent variable is denoted as $\mathbf{X}$, the observed single view is denoted as $\mathbf{Y}^s$, and the observed multi-view data for the encoder and decoder is denoted as $\mathbf{Y}_\gamma^v$ and $\mathbf{Y}^v$, respectively, where $v \in \{1, \cdots, V\}$. (a) In the GPLVM, a projection function $\boldsymbol{f}^s$ is learned to project the the latent variable $\mathbf{X}$ to the observed single view $\mathbf{Y}^s$. (b) In the SGPLVM, various projection functions $\{\boldsymbol{f}^v\}_{v=1}^V$ are learned to project the shared variable $\mathbf{X}$ to multiple observations $\{\mathbf{Y}^v\}_{v=1}^V$. (c) In the SAGP(SE), two types of mapping functions are learned. For the first type, we study a back constraint to project various observed data $\{\mathbf{Y}_\gamma^v\}_{v=1}^V$ to the shared variable $\mathbf{X}$ under the gaussian process priors $\{\boldsymbol{g}^v\}_{v=1}^V$; for the second type, being similar to SGPLVM, various projection functions $\{\boldsymbol{f}^v\}_{v=1}^V$ are learned to project the shared variable $\mathbf{X}$ to multiple observations $\{\mathbf{Y}^v\}_{v=1}^V$. (d) In the SAGP(IE), another two types of mapping functions are learned. The first one is to learn $\boldsymbol{g}^v$ to project $\mathbf{Y}_\gamma^v$ to $\mathbf{X}$ and the second type of projections is similar to that in SAGP(SE).

prior, our proposed method would be better to fit the training data exactly. Furthermore, there are only a few number of parameters that should be estimated in this kind of back-constraint compared with existing methods.

The marginal likelihood of $\mathbf{X}$ w.r.t. the multiple observations and the the observations w.r.t. $\mathbf{X}$ (SE) are then represented as follows

$$
\begin{aligned}
&p\left(\mathbf{X} \mid \{\mathbf{Y}^v\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V\right) = \frac{1}{\sqrt{(2\pi)^{Nq}\left|\sum_v^V \mathbf{K}_Y^v\right|^q}} \exp\{-\frac{1}{2}\mathrm{tr}((\sum_v^V \mathbf{K}_Y^v)^{-1}\mathbf{X}\mathbf{X}^T)\} \\
&p\left(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V\right) = \prod_{v=1}^V p(\mathbf{Y}^v \mid \mathbf{X}, \boldsymbol{\theta}^v) = \prod_{v=1}^V \frac{1}{\sqrt{(2\pi)^{ND^v}\left|\mathbf{K}_X^v\right|^{D^v}}} \exp\{-\frac{1}{2}\mathrm{tr}\big((\mathbf{K}_X^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T\big)\}
\end{aligned}
\tag{4.6}
$$

In order to clearly distinct the observed data in the encoder and the reconstructed data in the decoder, we re-symbol the data as $\left\{\mathbf{Y}_\gamma^v\right\}_v^V$ for the encoder. Noted that $\mathbf{Y}_\gamma^v$ is only a symbol and the numerical value of it can be set to be equal to $\mathbf{Y}^v$ in the implementation. Consequently, the joint likelihood function over the shared $\mathbf{X}$ based on the observations and the reconstructions can be represented as

$$
\begin{aligned}
&p(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V, \{\boldsymbol{\gamma}^v\}_V^V, \{\boldsymbol{\theta}^v\})_v^V \propto \\
&p(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V) p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V)
\end{aligned}
\tag{4.7}
$$

The Proof of Eq.(4.7):

$$
\begin{aligned}
&p\left(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V\right) \\
=&\frac{p\left(\mathbf{X}, \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V, \mid \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V\right)}{p\left(\left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V \mid \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V\right)} \\
=&\frac{p(\left\{\mathbf{Y}_\gamma^v\right\}_v^V) p(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V) p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V)}{p\left(\left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V \mid \{\gamma^v\}_v^V, \{\theta^v\}_v^V\right)} \\
=&\frac{p(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V) p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V)}{\frac{p(\left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\mathbf{Y}^v\}_v^V \mid \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V)}{p(\left\{\mathbf{Y}_\gamma^v\right\}_v^V)}} \\
=&\frac{p(\mathbf{X} \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V) p(\{\mathbf{Y}^v\}_v^V \mid \mathbf{X}, \{\boldsymbol{\theta}^v\}_v^V)}{p(\{\mathbf{Y}^v\}_v^V \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V)}
\end{aligned}
\tag{4.8}
$$

Since $p(\{\mathbf{Y}^v\}_v^V \mid \left\{\mathbf{Y}_\gamma^v\right\}_v^V, \{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V)$ is unrelated to the shared latent space, the result of Eq.(4.7) is obtained.

As mentioned above, the negative log likelihood of the proposed method is formed as follows,

$$\begin{aligned} L &= L_\gamma + L_\theta + const \\ L_\theta &= \sum_{i=1}^{V} L_\theta^v \\ L_\gamma &= \frac{1}{2}\{qN\log 2\pi + q\log\left|\sum_v \mathbf{K}_Y^v\right| + \mathrm{tr}((\sum_v \mathbf{K}_Y^v)^{-1}\mathbf{X}\mathbf{X}^T)\} \\ L_\theta^v &= \frac{D^v}{2}N\log 2\pi + \frac{D^v}{2}\log|\mathbf{K}_X^v| + \frac{1}{2}\mathrm{tr}((\mathbf{K}_X^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T) \end{aligned} \tag{4.9}$$

where $const$ means the constant part which is unrelated to the latent space. For the IE scenario, the kernel matrix is replaced by $\mathbf{K}_Y^v$ in L_γ.

4.1.1 Discriminative Prior

Since we prefer to apply SAGP to the classification, a discriminative prior is used to encourage the latent variables belonging to the same class to be close and those belonging to the different classes to be far on the latent space. Urtasun et al. [69] firstly utilized a Linear Discriminant Analysis (LDA) as the prior by maximizing the between-class while minimizing the within-class through

$$\arg\max J(\mathbf{X}) = \mathrm{tr}(\mathbf{S}_w^{-1}\mathbf{S}_b) \tag{4.10}$$

where $\mathbf{S}_b$ denotes the between-class matrix and $\mathbf{S}_w$ denotes the within-class matrix, respectively. Moreover, this prior is modified to the graph Laplacian matrix by Eleftheriadis et al. [15] and applied to multi-view data. In this thesis, we exploit this kind of discriminative prior in [15] for the classification task. Theoretically, a weight matrix $\mathbf{W}^v$ for each view is first calculated.

$$\mathbf{W}_{ij}^v = \begin{cases} \exp\left(-\frac{\|\mathbf{y}_i^v - \mathbf{y}_j^v\|_2^2}{t^v}\right) & \textbf{if } i \neq j \textbf{ and } c_i = c_j \\ 0 & \textbf{otherwise} \end{cases} \tag{4.11}$$

where $v = 1, \cdots, V$, $\mathbf{y}_i^v$ is the i-th feature in the sample set $\mathbf{Y}^v$, c_i is its corresponding class label, and t^v is the RBF kernel parameter to control the kernel width. Based on [60], we set it as the mean squared distance between the training samples. Then we can get the graph

Laplacian $\mathbf{L}^v = \mathbf{D}^v - \mathbf{W}^v$ for each view, where the matrix $\mathbf{D}^v$ is diagonal and its diagonal element is defined as $\mathbf{D}^v_{ii} = \sum_{j=1} \mathbf{W}^v_{ij}$. Since the graph Laplacian matrices from different views are different in scale, Eleftheriadis et al. [15] normalized them by

$$\mathbf{L}^v_N = (\mathbf{D}^v)^{-1/2}\mathbf{L}^v(\mathbf{D}^v)^{-1/2} \tag{4.12}$$

Consequently, the discriminative prior on $\mathbf{X}$ is defined as

$$p(\mathbf{X}) = \prod_{v=1}^{V} p(\mathbf{X} \mid \mathbf{Y}^v)^{1/V} = \frac{1}{V \cdot Z_q} \exp\left(-\frac{\beta}{2}\mathrm{tr}(\mathbf{X}^T\tilde{\mathbf{L}}\mathbf{X})\right) \tag{4.13}$$

where $\tilde{\mathbf{L}} = \sum_{v=1}^{V} \mathbf{L}^v_N + \xi\mathbf{I}$, ξ (e.g., 10^{-4}) is a regularization parameter with a small value to ensure $\tilde{\mathbf{L}}$ to be positive-definite, β is a non-negative penalty parameter to get a trade-off between the Eq.(4.9) and Eq.(4.13). In conclusion, by jointly considering Eq.(4.9) and Eq.(4.13), we can get the negative log posterior of SAGP as follows,

$$L = L_\gamma + L_\theta + L_d \tag{4.14}$$

where $L_d = \frac{\beta}{2}\mathrm{tr}(\mathbf{X}^T\tilde{\mathbf{L}}\mathbf{X})$.

4.2 Optimization

In this thesis, the gradient descend algorithm is applied to calculate the latent variable $\mathbf{X}$ and the hyperparameters $\{\boldsymbol{\gamma}^v\}_v^V, \{\boldsymbol{\theta}^v\}_v^V$. However, we find that the value of L_γ is much smaller than that of L_θ, which greatly influences the performance in experiments. Thus, we update the shared space alternatively in L_γ and $L_\theta + L_d$. The objective function Eq.(4.14) is transformed to Eq.(4.15).

$$\arg\min L_\gamma + L_\theta + L_d \quad s.t. \quad \mathbf{X} = \mathbf{X}_\gamma \tag{4.15}$$

where $\mathbf{X}_\gamma$ is the shared variable learned from L_γ, and $\mathbf{X}$ is the shared variable learned from $L_\theta + L_d$. To this end, the Augmented Lagrangian method (ALM) is used to solve this prob-

lem. As a result, we obtain the following augmented Lagrangian function shown as follows

$$\begin{aligned}\arg\min L_{ALM} =& \frac{1}{2}\{qN\log 2\pi + q\log\left|\sum_v \mathbf{K}_Y^v\right| + \mathrm{tr}((\sum_v \mathbf{K}_Y^v)^{-1}\mathbf{X}_\gamma\mathbf{X}_\gamma^T)\}+ \\ & \frac{1}{2}\sum_{v=1}^{V}\left\{D^v N\log 2\pi + D^v\log|\mathbf{K}_X^v| + \mathrm{tr}((\mathbf{K}_X^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T)\right\} \\ & + \frac{\beta}{2}\mathrm{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) + \langle\mathbf{Z}, \mathbf{X}-\mathbf{X}_\gamma\rangle + \frac{\mu}{2}\left\|\mathbf{X}-\mathbf{X}_\gamma\right\|_F^2\end{aligned} \tag{4.16}$$

where $\mathbf{Z}$ is the Lagrange multiplier, $\langle\cdot,\cdot\rangle$ is the inner production and μ is the non-negative penalty parameter.

Subsequently, the gradients w.r.t. $\mathbf{X}_\gamma$ and $\mathbf{X}$ can be acquired as follows:

$$\frac{\partial L_{ALM}}{\partial \mathbf{X}_\gamma} = (\mathbf{K}_Y)^{-1}\mathbf{X}_\gamma - \mu\left(\mathbf{X}-\mathbf{X}_\gamma+\frac{\mathbf{Z}}{\mu}\right) \tag{4.17}$$

$$\frac{\partial L_{ALM}}{\partial \mathbf{X}} = \sum_{v=1}^{V}\frac{\partial L_\theta^v}{\partial \mathbf{X}} + \beta\mathbf{L}\mathbf{X} + \mu\left(\mathbf{X}-\mathbf{X}_\gamma+\frac{\mathbf{Z}}{\mu}\right) \tag{4.18}$$

β is a scaling parameter to trade-off the significance of the discriminative prior, and we get it empirically. Since the likelihood term L_θ^v is a function of $\mathbf{K}_X^v$, the chain rule is applied to get the derivatives:

$$\frac{\partial L_\theta^v}{\partial x_{ij}} = \mathrm{tr}\left((\frac{\partial L_\theta^v}{\partial \mathbf{K}_X^v})^T\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}\right) \tag{4.19}$$

Similarly, the derivatives with the kernel parameters $\{\boldsymbol{\gamma}^v\}_v^V$ and $\{\boldsymbol{\theta}^v\}_v^V$ can be derived as follows:

$$\frac{\partial L_{ALM}}{\partial \gamma_i^v} = \mathrm{tr}\left((\frac{\partial L_{ALM}}{\partial \sum_v^V \mathbf{K}_Y^v})^T\frac{\partial \mathbf{K}_Y^v}{\partial \gamma_i^v}\right) \tag{4.20}$$

$$\frac{\partial L_{ALM}}{\partial \theta_i^v} = \mathrm{tr}\left((\frac{\partial L_\theta^v}{\partial \mathbf{K}_X^v})^T\frac{\partial \mathbf{K}_X^v}{\partial \theta_i^v}\right) \tag{4.21}$$

After obtaining the latent variable $\mathbf{X}_\gamma$, $\mathbf{X}$ and their corresponding parameters, the Lagrange multiplier $\mathbf{Z}$ and parameter μ can be updated as

$$\begin{aligned}\mathbf{Z}_{t+1} &= \mathbf{Z}_t + \mu_t(\mathbf{X}-\mathbf{X}_\gamma) \\ \mu_{t+1} &= \min(\mu_{max}, \rho\mu_t)\end{aligned} \tag{4.22}$$

from the t-th iteration to the $(t+1)$-th iteration, where ρ is a constant keeping the step size of μ (it is typically set as 1.1), and the μ_{max} is the predefined maximum of μ (it is typically set as 1000).

Algorithm 4.1 Shared Auto-encoder Gaussian Process latent variable model (SAGP)

Input: the observed training data $\{\mathbf{Y}^v\}_v^V$, testing data $\{\mathbf{y}^{v*}\}_v^V$, and the prior parameter β and the dimensionality of the latent space q.

Output: the latent variables $\mathbf{X}$ and $\mathbf{x}^*$.

Initialization: initialize the latent variable $\mathbf{X}$ using LDA.

1: **while** not converged **do**

2: **Encode-step**:

3: optimize the latent variable $\mathbf{X}_\gamma$ and hyperparameter $\{\boldsymbol{\gamma}\}_v^V$ by Eq.(4.17) and Eq.(4.20)

4: **Decode-step**:

5: compute the latent variable $\mathbf{X}$ and hyperparameter $\{\boldsymbol{\theta}\}_v^V$ by Eq.(4.18), Eq.(4.19) and Eq.(4.21)

6: **Update Lagrange multiplier and $\boldsymbol{\mu}$**:

7: update $\mathbf{Z}$ and $\boldsymbol{\mu}$ using Eq.(4.22).

8: **end while**

9: **Test**: get the latent variable for the test sample by Eq.(4.23) or Eq.(4.24).

4.3 Inference

After the optimization, the latent variable and parameters for the proposed model are obtained. Given the new test multi-modal observations $\mathbf{y}^* = \{\mathbf{y}^{v*}\}_v^V$, the inference procedure is quite simple and straightforward. Based on the result in [56] Chapter 2, the latent variable $\mathbf{x}^*$ can be computed by

$$\mathbf{x}^* = (\sum_v \mathbf{K}_{y^{v*}Y^v})(\sum_v \mathbf{K}_Y^v)^{-1}\mathbf{X} \tag{4.23}$$

where $\mathbf{K}_{y^{v*}Y^v}$ denotes the kernel matrix evaluated at all pairs of training and test points. For the independent encoder (IE) scenario, the latent variable $\mathbf{x}^*$ can be estimated by Eq.(4.24) if the v-th view is available.

$$\mathbf{x}^* = \mathbf{K}_{y^{v*}Y^v}(\mathbf{K}_Y^v)^{-1}\mathbf{X} \tag{4.24}$$

The optimization and inference of the proposed method are shown in Algorithm 1. After obtaining the latent variable $\mathbf{X}$ associated with training samples and $\mathbf{x}^*$ associated with the test sample through Eq. (4.23) or Eq. (4.24), the KNN (K is set to 1 in our experiments) classifier is applied to estimate the label of the test sample.

4.4 Computational Complexity Analysis

Without loss the generality, here we firstly denote $D = \max\{D_v\}_{v=1}^{V}$ and number of iterations in the gradient decent technique as t. Since SAGP(SE) is more complex than SAGP(IE), we only analyze SAGP(SE). From the optimization we can see that the main complexity exists in Eq.(4.18). For $\frac{\partial L_\theta^v}{\partial \mathbf{K}_X^v}$, the complexity is $O(N^3 + N^2 D)$. For $\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}$, the complexity is $O(Nq)$. Therefore, the complexity of Eq.(4.19) is $O((N^3 + N^2 D) + (Nq + N^2))$. As $\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}$ should be computed with N^2 times, the complexity of Eq.(4.18) is $O(V(N^2 D + N^2(Nq + N^2)) + qNC) = O(V(N^2 D + N^3 q + N^4) + qNC)$. Since N is much larger than q, D and C in most of the time, the computational complexity of our optimization is nearly $O(tVN^4)$. Despite that the computational complexity of our training step is relatively high, the complexity of our testing step is quite efficient which is only $O(VDN + N^2 + Nq)$ according to Eq.(4.23).

4.5 Experiments

In this section, the proposed method SAGP is applied to the object recognition on three real-world datasets to demonstrate its effectiveness and superiority compared with state-of-the-art approaches. We first introduce the datasets used in this section, followed by the experimental setting. The experimental results on both overall and average classification accuracy are then illustrated. The sensitivity of the parameter selection is finally analyzed.

4.5.1 Dataset description

The Wiki Text-Image dataset [55] was collected from the WikiPedia's featured articles collection. Since some of the collected classes are very scarce, only the 10 most populated ones are used here. Totally, this dataset consists of 2173 training samples and 693 testing samples. Simultaneously, each sample contains a single image and at least 70 words. Here, the 128-D SIFT feature is used to represent the image and the 10-D latent Dirichlet

allocation feature is used to represent the text.

There are 30,475 images, containing 50 classes in the animals with attributes (AWA) dataset [30, 34, 35]. Two kinds of Convolutional Neural Network (CNN) features: the fc7 of 7-layer CaffeNet (pretrained on ILSVRC2012) feature and the fc7 layer of very deep 19-layer CNN (pretrained on ILSVRC2014) feature are selected. Here, we randomly selected 50 samples from each class for training, and the remaining images are used for testing. Being similar to [15], we first exploit the Principal Component Analysis (PCA) to primarily reduce the dimension of the CNN features to 300, which greatly decreases the training time.

The NUS-WIDE-LITE dataset [9] consists of 27807 samples for training and 27,808 for testing. Since there are some classes with scarce images, 9 classes are chosen, including birds, boats, flowers, rocks, sun, tower, toy, tree, and vehicle. Furthermore, due to some multi-label samples existing in this dataset, we remove the samples with zero or more than one labels. Therefore, 16,377 samples are selected. Then we randomly select 200 samples from each class for training, and remaining samples are used for testing, respectively. Besides, three types of features including the color correlogram histogram (CH), edge direction histogram (EDH), and wavelet texture (WT) features are input into different methods after a PCA pre-process, being similar to [15]. The dimensions of three features are 33, 43 and 40, respectively.

4.5.2 Experimental settings

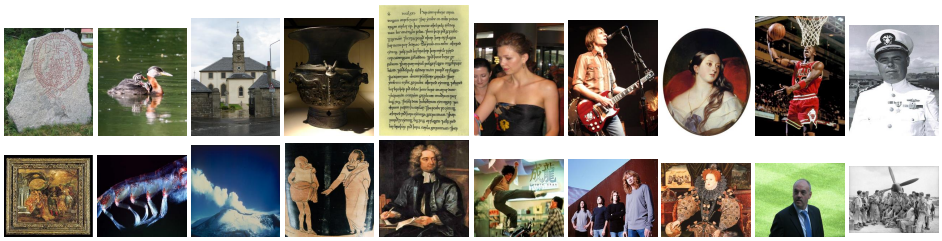
To quantitively evaluate the superiority of SAGP, we make a comparison among it, CCA [23, 68], JSSL [44], AWFA [76], GPLVM [37], DGPLVM [69], and DS-GPLVM [15]. Since GPLVM and DGPLVM only adapt for sing-view based data, we concatenate multiple views as a single vector as their input. Additionally, the number of outputs of CCA is based on the number of views. Here, we only show its best results. In [15], DS-GPLVM also has two back constraints denoted as DS-GPLVMI and DS-GPLVMS. Similarly, since DS-GPLVMI and the independent encoder based SAGP has different experimental results for each view, only the best performance is displayed in this thesis. Besides, apart from the

Table 4.1. The experimental results (overall accuracies) on three datasets computed by DCCA and DCCAE with different structures.

Dataset	Structure	DCCA	DCCAE
Wiki	$D_{in} \times 100 \times 10$	**66.4%**	60.0%
	$D_{in} \times 200 \times 10$	63.5%	63.5%
	$D_{in} \times 300 \times 10$	63.4%	62.8%
	$D_{in} \times 400 \times 10$	62.2%	**67.1%**
	$D_{in} \times 200 \times 100 \times 10$	63.9%	62.8%
AWA	$D_{in} \times 100 \times 100$	75.3%	74.7%
	$D_{in} \times 200 \times 100$	77.5%	76.5%
	$D_{in} \times 300 \times 100$	**78.2%**	**76.9%**
	$D_{in} \times 400 \times 100$	77.5%	76.7%
	$D_{in} \times 200 \times 100 \times 100$	66.9%	64.9%
NUS	$D_{in} \times 100 \times 10$	29.4%	34.1%
	$D_{in} \times 200 \times 10$	30.8%	35.0%
	$D_{in} \times 300 \times 10$	31.2%	34.8%
	$D_{in} \times 400 \times 10$	**31.5%**	**35.4%**
	$D_{in} \times 200 \times 100 \times 10$	28.9%	21.6%

JSSL, we use the 1-NN classifier with the latent variable as the input to classify different classes.

Since the auto-encoder structure is introduced in the proposed model, we also compare SAGP with multi-view deep learning methods containing deep canonical correlation analysis (DCCA) [3] and deep canonically correlated auto-encoders (DCCAE) [70]. Based on the experimental results as shown in Tab.4.1, we find that the these two deep learning structures with more than two latent layers would suffer from a performance degradation. Empirically, the structures of DCCA and DCCAE are set as $D_{in} \times D_m \times d_{out}$ and $D_{in} \times D_m \times d_{out} \times D_m \times D_{in}$, respectively, where D_{in} is the dimensionality of the input vector, D_m is the dimensionality of the first latent layer, and d_{out} is the dimensionality of the output used for classification by applying the 1-NN classifier. Based on the experimental results in Tab.4.1, we set D_m as 100 and 400 for DCCA and DCCAE in the Wiki Text-Image dataset, respectively. Similarly, D_m is set as 300 and 400 for both methods in the AWA and



	Art	Biology	Geography	History	Literature	Media	Music	Royalty	Sport	Warfare
CCA	41.2%	89.8%	57.3%	40.0%	49.2%	39.7%	52.9%	51.2%	90.1%	81.7%
JSSL	25.7%	91.0%	77.0%	28.2%	71.0%	51.3%	50.3%	78.7%	86.7%	89.3%
AWFA	14.7%	92.1%	77.1%	45.9%	69.2%	44.8%	60.8%	51.2%	87.3%	88.5%
GPLVM	47.1%	85.2%	57.3%	37.6%	60.0%	37.9%	45.1%	46.3%	91.5%	79.8%
DGPLVM	35.3%	87.5%	64.6%	43.5%	55.4%	39.7%	41.2%	58.5%	90.1%	79.8%
DSGPLVMI	38.2%	90.9%	71.9%	54.1%	49.2%	37.9%	56.7%	58.5%	88.7%	81.7%
DSGPLVMS	32.4%	92.1%	69.8%	64.7%	60.0%	41.4%	56.9%	61.0%	91.6%	81.7%
DCCA	50.0%	93.2%	64.6%	36.5%	55.4%	44.8%	52.9%	61.0%	90.1%	82.7%
DCCAE	44.1%	89.8%	58.3%	44.7%	58.5%	41.4%	51.0%	58.5%	88.7%	88.5%
SAGP(IE)	35.3%	**93.2%**	70.8%	***62.4%***	60.0%	48.3%	***56.9%***	53.7%	**93.0%**	82.7%
SAGP(SE)	35.3%	**93.2%**	68.8%	**65.9%**	58.5%	***50.0%***	**68.6%**	56.1%	***91.5%***	82.7%

Figure 4.3. The classification accuracies for each class by using various single-view or multi-view based approaches on the Wiki Test-Image dataset. The percentages with bold font means our method obtains the best result, and the percentage with bold italic font means SAGP obtains the second best result in comparison to other approaches.

the NUS-WIDE-LITE datasets, respectively.

4.5.3 Performance on the three datasets

Wiki Text-Image dataset: The overall and average classification accuracies on the Wiki Text-Image dataset are tabulated in Tab.4.2 when the dimension of $\mathbf{X}$ ranges from 1 to 10. We can see that SAGP achieves the highest values on both overall accuracy and average accuracy in most cases compared with CCA, JSSL, AWFA, GPLVM, DGPLVM, DS-GPLVM, DCCA and DCCAE. In comparison to the CCA and GPLVM, SAGP obtains at least a 7%-8% enhancement when the dimension is 8. Compared with DGPLVM, SAGP(IE) and SAGP(SE) have also a remarkable improvement at many times. Referring to DS-GPLVMI and DS-GPLVMS, SAGP is also competitive. In contrast to two deep learning methods: DCCA and DCCAE, our proposed method SAGP gains an obvious achievement due to the few parameter estimation and strong data representation ability of SAGP. Dif-

Table 4.2. The experimental results including both overall and average accuracies on the Wiki Text-Image dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 1 to 10.

		Dimensionality									
Methods	Result	d_{out}=1	d_{out}=2	d_{out}=3	d_{out}=4	d_{out}=5	d_{out}=6	d_{out}=7	d_{out}=8	d_{out}=9	d_{out}=10
JSSL	Overall	68.0%									
	Average	64.9%									
AWFA	Overall	68.7%									
	Average	63.2%									
CCA	Overall	28.1%	45.0%	56.0%	59.0%	61.8%	63.1%	63.8%	62.6%	63.8%	63.8%
	Average	27.0%	42.4%	51.8%	55.4%	58.4%	59.3%	61.1%	59.3%	60.4%	60.4%
GPLVM	Overall	14.9%	35.9%	46.9%	54.3%	58.6%	61.3%	61.9%	61.9%	60.2%	59.0%
	Average	12.8%	33.2%	42.8%	50.0%	53.2%	57.0%	57.4%	58.8%	56.2%	54.1%
DGPLVM	Overall	37.2%	55.0%	56.1%	61.9%	62.9%	63.5%	63.4%	63.4%	63.2%	61.8%
	Average	34.2%	51.7%	51.4%	57.5%	60.4%	58.5%	60.0%	59.6%	59.3%	57.6%
DS-GPLVMI	Overall	38.4%	53.8%	58.6%	63.4%	66.1%	66.8%	66.1%	66.8%	65.7%	64.4%
	Average	33.7%	50.0%	54.3%	58.9%	60.8%	62.6%	61.8%	62.8%	61.4%	60.6%
DS-GPLVMS	Overall	40.8%	55.8%	62.2%	65.4%	69.4%	69.0%	69.3%	69.4%	68.7%	**67.4%**
	Average	36.3%	52.7%	57.7%	60.5%	65.3%	65.1%	**65.9%**	65.1%	64.7%	**62.9%**
DCCA	Overall	21.5%	49.1%	53.1%	57.9%	62.9%	64.5%	64.9%	65.8%	65.2%	66.4%
	Average	20.4%	43.9%	49.1%	54.1%	58.3%	60.8%	61.6%	63.1%	62.0%	63.7%
DCCAE	Overall	24.0%	40.8%	57.7%	58.0%	59.0%	61.3%	60.2%	65.7%	62.9%	67.1%
	Average	22.5%	38.8%	52.9%	54.9%	55.0%	57.9%	57.0%	62.4%	58.3%	64.3%
SAGP(IE)	Overall	48.9%	58.6%	65.1%	67.5%	69.1%	**70.0%**	69.6%	70.0%	**69.7%**	**67.4%**
	Average	45.2%	53.7%	61.1%	63.6%	65.3%	**66.9%**	65.3%	65.6%	**65.5%**	62.4%
SAGP(SE)	Overall	**50.5%**	**58.9%**	**65.6%**	**67.7%**	**70.6%**	**70.0%**	**69.7%**	**71.0%**	69.3%	65.1%
	Average	**48.0%**	**54.9%**	**62.3%**	**63.8%**	**67.2%**	66.5%	65.5%	**67.1%**	64.6%	61.7%

Table 4.3. The experimental results (overall accuracies) computed by DCCA and DCCAE using dropout and sparsity regularization on the Wiki Text-Image dataset with different structures.

Dataset	Structure	DCCA	DCCAE
Dropout	$D_{in} \times 100_{0.3} \times 10$	62.1%	-
	$D_{in} \times 100_{0.4} \times 10$	64.2%	-
	$D_{in} \times 100_{0.5} \times 10$	64.1%	-
	$D_{in} \times 400_{0.3} \times 10$	-	62.3%
	$D_{in} \times 400_{0.4} \times 10$	-	66.4%
	$D_{in} \times 400_{0.5} \times 10$	-	63.8%
	$D_{in} \times 200_{0.5} \times 100 \times 10$	63.9%	63.4%
	$D_{in} \times 200 \times 100_{0.5} \times 10$	63.6%	65.8%
	$D_{in} \times 200_{0.5} \times 100_{0.5} \times 10$	62.2%	63.1%
Sparsity	$D_{in} \times 100 \times 10$	66.8%	61.2%
	$D_{in} \times 200 \times 10$	62.8%	62.8%
	$D_{in} \times 300 \times 10$	63.9%	63.5%
	$D_{in} \times 400 \times 10$	62.5%	65.4%
	$D_{in} \times 200 \times 100 \times 10$	61.5%	61.2%

ferently, DCCA and DCCAE are easy to have the over-fitting. Tab.4.3 shows the results obtained by DCCA and DCCAE if the dropout and sparsity regularization are used. Note that $D_{in} \times 100_{0.3} \times 10$ means the dropout probability is 0.3 in the second layer. So do other similar symbols. Despite that there is a slight improvement at some times by adding the sparsity, they are still inferior to our method.

Fig.4.3 displays the classification accuracies of each class computed by different approaches, when dimension q is set to be 8. It is clear to see that SAGP achieves the best performance in the biology, history, music, and sport. Particularly, almost all of the category classification accuracies computed by SAGP(SE) are equal to or larger than 50% except for the art. By contrast, CCA, DCCAE, GPLVM and DS-GPLVMI have at least three classes whose results are below 50%. Compared with JSSL, AWFA, and DS-GPLVMS, the proposed method SAGP is also competitive.

Table 4.4. The experimental results including both overall and average accuracies on the AWA dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 40 to 130.

		Dimensionality									
Methods	Result	d_{out}=40	d_{out}=50	d_{out}=60	d_{out}=70	d_{out}=80	d_{out}=90	d_{out}=100	d_{out}=110	d_{out}=120	d_{out}=130
JSSL	Overall	83.8%									
	Average	81.7%									
AWFA	Overall	83.4%									
	Average	81.3%									
CCA	Overall	71.8%	73.6%	74.3%	74.8%	75.1%	75.1%	75.5%	73.8%	74.1%	74.1%
	Average	69.2%	71.1%	72.8%	72.4%	72.8%	72.7%	73.2%	71.6%	72.0%	71.8%
GPLVM	Overall	70.9%	71.9%	71.4%	71.1%	69.1%	72.1%	71.4%	71.8%	72.1%	71.6
	Average	69.5%	70.6%	69.9%	70.3%	67.2%	69.7%	70.9%	70.7%	71.6%	70.8%
DGPLVM	Overall	80.1%	82.0%	80.2%	78.8%	77.6%	76.5%	75.4%	75.0%	74.5%	73.8%
	Average	78.3%	80.1%	78.6%	77.3%	76.1%	75.0%	74.0%	73.6%	73.2%	72.3%
DS-GPLVMI	Overall	81.8%	82.9%	82.8%	81.7%	82.8%	82.7%	82.4%	82.3%	82.4%	82.3%
	Average	**80.0%**	81.0%	80.9%	79.8%	81.1%	80.8%	80.6%	80.5%	80.7%	80.8%
DS-GPLVMS	Overall	82.0%	83.4%	83.1%	82.4%	82.5%	82.7%	82.7%	82.5%	**83.2%**	82.7%
	Average	**80.0%**	81.4%	**81.5%**	80.7%	80.8%	80.8%	80.8%	81.4%	**81.5%**	81.2%
DCCA	Overall	76.6%	78.3%	77.5%	77.9%	76.9%	77.9%	78.2%	78.4%	78.1%	78.3%
	Average	74.4%	76.3%	75.3%	76.0%	74.6%	75.6%	76.0%	76.1%	75.8%	76.1%
DCCAE	Overall	76.7%	77.1%	76.9%	76.7%	76.8%	77.0%	76.9%	76.9%	76.9%	76.8%
	Average	74.8%	75.0%	74.8%	74.9%	74.9%	74.9%	74.8%	74.8%	74.8%	74.7%
SAGP(IE)	Overall	82.0%	83.6%	83.0%	82.7%	82.2%	81.8%	81.4%	81.1%	80.4%	79.8%
	Average	79.9%	81.5%	81.1%	80.7%	80.3%	79.8%	79.5%	79.4%	78.7%	78.0%
SAGP(SE)	Overall	**82.4%**	**84.3%**	**83.4%**	**82.7%**	**83.3%**	**83.3%**	**83.1%**	**83.3%**	82.6%	**82.9%**
	Average	**80.0%**	**82.3%**	**81.5%**	**80.8%**	**81.4%**	**81.4%**	**81.0%**	**81.5%**	80.9%	**81.4%**

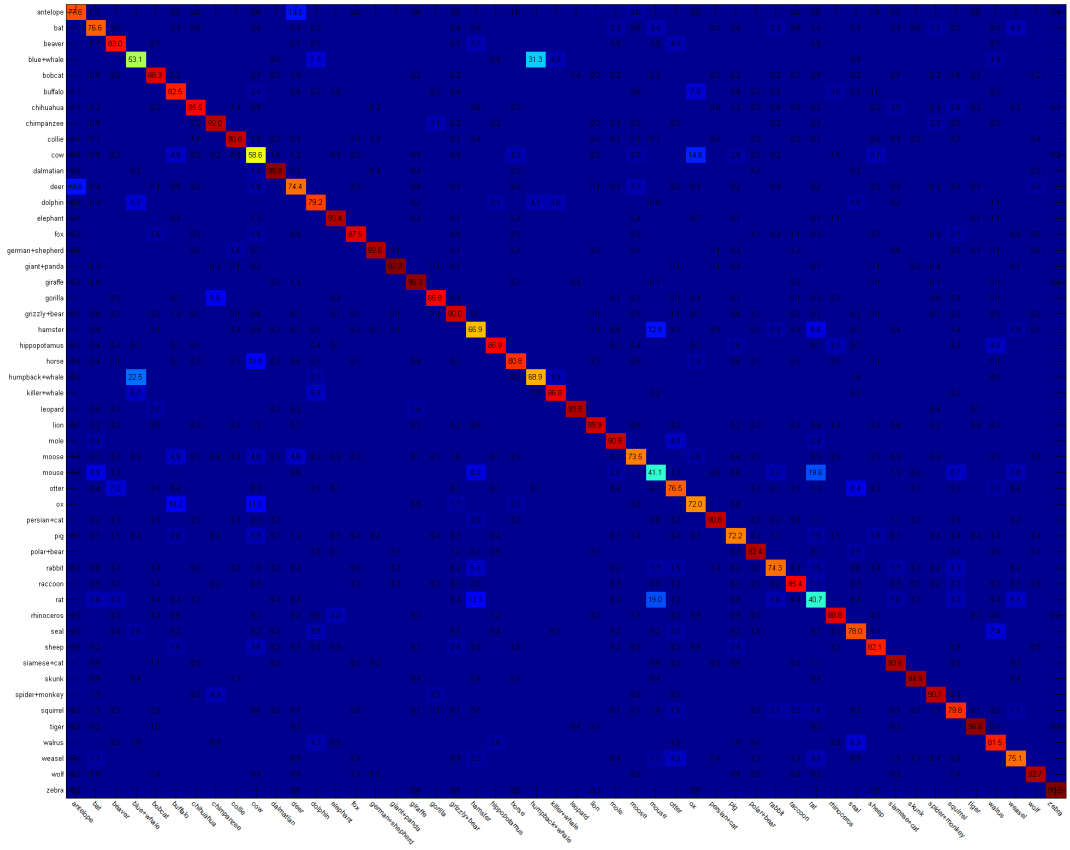


Figure 4.4. Confusion matrix of the category recognition results on the AWA Dataset.

AWA dataset: Tab.4.4 lists the overall classification accuracies and average accuracies conducted on the AWA dataset. In contrast to CCA and GPLVM, SAGP obtains much better results on both evaluation indexes. Compared with the DGPLVM, DCCA, DCCAE and DS-GPLVM, our proposed method SAGP also achieves competitive results. Specifically, there is almost a 2% improvement in comparison to DS-GPLVM. In contrast to DCCA and DCCAE, SAGP achieves almost 6 percents enhancement. Tab.4.5 displays experimental results computed by DCCA and DCCAE with dropout and sparsity regularization. It is clear to see that both methods gain a more or less improvement in some cases. However, SAGP is still much superior to them.

The confusion matrix calculated by SAGP(SE) on the AWA dataset is shown in Fig.4.4, whose diagonal elements mean the classification accuracy for each class. Note that

Table 4.5. The experimental results (overall accuracies) computed by DCCA and DCCAE using dropout and sparsity regularization on the AWA dataset with different structures.

Dataset	Structure	DCCA	DCCAE
Dropout	$D_{in} \times 300_{0.2} \times 100$	77.7%	77.5%
	$D_{in} \times 300_{0.3} \times 100$	77.5%	77.7%
	$D_{in} \times 300_{0.4} \times 100$	77.6%	78.1%
	$D_{in} \times 300_{0.5} \times 100$	77.0%	77.8%
	$D_{in} \times 200_{0.5} \times 100 \times 100$	67.5%	72.7%
	$D_{in} \times 200 \times 100_{0.5} \times 100$	57.0%	59.4%
	$D_{in} \times 200_{0.5} \times 100_{0.5} \times 100$	56.2%	58.8%
Sparsity	$D_{in} \times 100 \times 10$	75.4%	75.3%
	$D_{in} \times 200 \times 10$	77.6%	76.8%
	$D_{in} \times 300 \times 10$	77.8%	77.1%
	$D_{in} \times 400 \times 10$	77.8%	77.7%
	$D_{in} \times 200 \times 100 \times 10$	75.12%	68.6%

the dimensionality q of $\mathbf{X}$ is set to be 50. From this figure we can see that the accuracy of most classes outperforms 70% except for the blue+whale, cow, hamster, humpback+whale, moose and rat, which relatively demonstrates the effectiveness of our proposed model SAGP.

NUS-WIDE-LITE dataset: The overall and average accuracy on the NUS-WIDE-LITE dataset is tabulated in Tab.4.6, when the dimensionality of $\mathbf{X}$ ranges from 1 to 30. Since the CCA, DCCA, and DCCAE are designed for two-view based data, the second and third modalities are concatenated to be a single one empirically. From the Tab.4.6 we can see that the results obtained by SAGP are much better than that obtained by other methods. Similarly, we also conduct an experiment for DCCA and DCCAE with the dropout and sparsity regularization, and the results are listed in Tab.4.7. Obviously, the classification accuracy obtained by SAGP is still much higher than that computed by these two methods. Compared with the JSSL, AWF, GPLVM, DGPLVM and DS-GPLVMS, SAGP(SE) outperforms these four approaches remarkably.

The confusion matrices of various approaches are displayed in Fig.4.5 when the dimensionality q is 10. It is obvious to observe that the classification accuracy of different categories obtained by our proposed method is superior or competitive to that gained by other comparison methods.

Table 4.6. The experimental results including both overall and average accuracies on the NUS-WIDE-LITE dataset obtained by our proposed method and other comparison approaches under the changes of the dimensionality of the latent variable from 1 to 30.

		Dimensionality						
Methods	Result	d_{out}=1	d_{out}=5	d_{out}=10	d_{out}=15	d_{out}=20	d_{out}=25	d_{out}=30
JSSL	Overall	47.2%						
	Average	48.9%						
AWFA	Overall	51.3%						
	Average	50.1%						
CCA	Overall	15.1%	30.6%	35.6%	36.7%	37.4%	37.7%	37.9%
	Average	14.8%	29.6%	34.2%	34.7%	35.9%	35.3%	36.6%
GPLVM	Overall	16.3%	33.1%	36.7%	40.7%	42.1%	40.0%	40.3%
	Average	14.9%	32.8%	34.8%	39.8%	40.5%	37.9%	39.6%
DGPLVM	Overall	19.6%	43.9%	51.2%	50.6%	51.9%	46.2%	45.1%
	Average	21.5%	49.9%	52.8%	53.4%	54.0%	45.5%	44.8%
DS-GPLVMI	Overall	19.1%	24.5%	32.9%	33.8%	33.5%	32.6%	31.9%
	Average	17.6%	30.7%	34.9%	35.9%	37.5%	34.3%	36.0%
DS-GPLVMS	Overall	**25.4%**	46.1%	54.8%	54.4%	53.8%	50.5%	48.9%
	Average	25.0%	**52.5%**	**56.6%**	**56.1%**	55.6%	49.6%	48.6%
DCCA	Overall	14.6%	27.2%	31.5%	33.4%	34.0%	34.1%	35.0%
	Average	14.4%	26.2%	31.5%	32.5%	32.2%	32.3%	33.6%
DCCAE	Overall	17.1%	29.8%	35.4%	36.4%	36.2%	37.3%	36.5%
	Average	17.1%	28.4%	33.7%	34.4%	35.0%	36.1%	36.0%
SAGP(IE)	Overall	19.8%	29.1%	32.4%	34.7%	34.6%	37.8%	33.2%
	Average	17.7%	33.7%	37.2%	36.4%	34.2%	37.5%	34.0%
SAGP(SE)	Overall	23.6%	**52.6%**	**58.7%**	**56.2%**	**56.9%**	**53.1%**	**51.8%**
	Average	**25.2%**	51.9%	**56.6%**	55.0%	**55.7%**	**50.4%**	**49.8%**

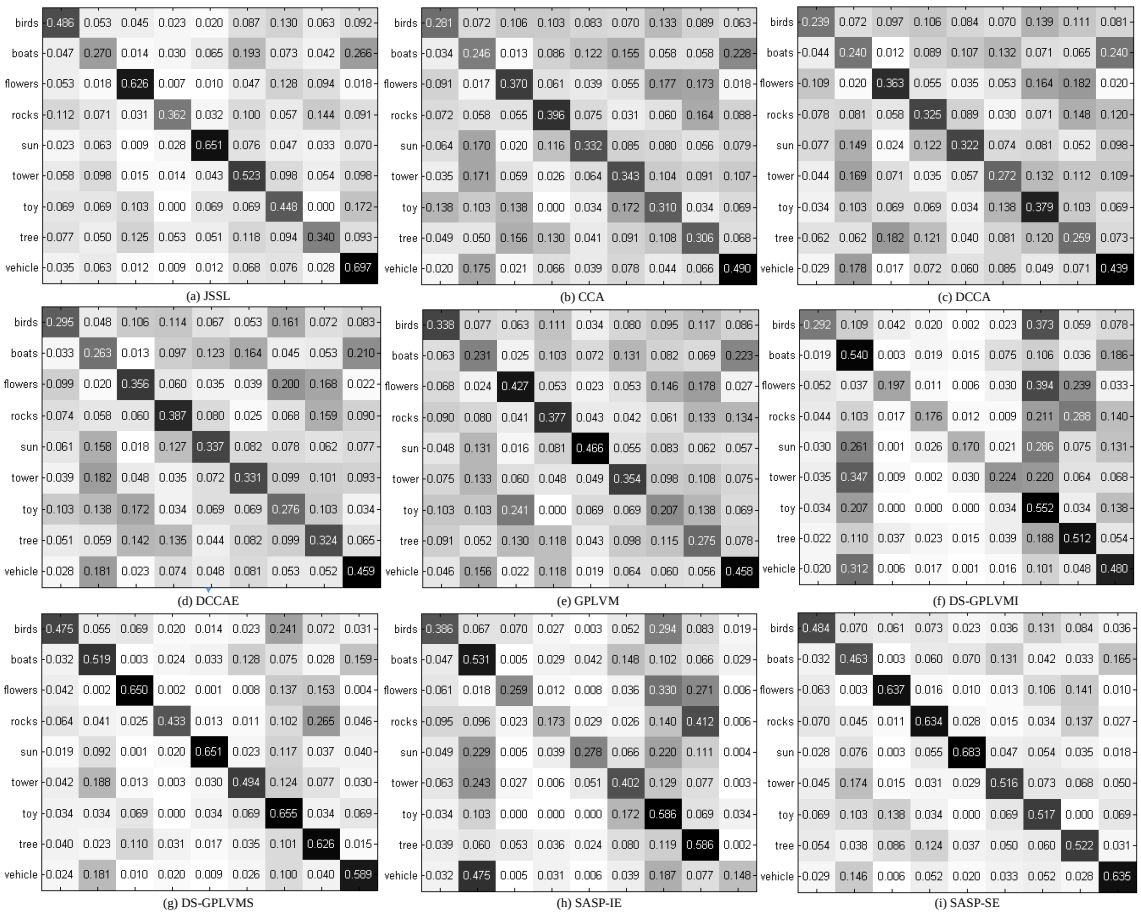


Figure 4.5. Confusion matrix of the category recognition results computed by different methods on the NUS-WIDE-LITE dataset. The vertical axis shows the true labels and the horizontal axis shows the predicted labels.

4.5.4 Parameter analysis

In our proposed model, the selection of different parameters including q and β is important. Here we give a discussion on these two parameters.

Fig.4.6 shows how the different values of dimensions of the latent variable affect the classification accuracy. From this figure we can see that SAGP(SE) is optimal on the 8-D, 50-D, and 10-D for the three datasets, respectively. In fact, the value of q is quite significant to the performance. With the increasing of q, the classification accuracy has a remarkable increase. The reason is that the lower dimensional shared space may lose

Table 4.7. The experimental results (overall accuracies) computed by DCCA and DCCAE using dropout and sparsity regularization on the NUS-WIDE-LITE dataset with different structures.

Dataset	Structure	DCCA	DCCAE
Dropout	$D_{in} \times 400_{0.2} \times 10$	31.3%	35.6%
	$D_{in} \times 400_{0.3} \times 10$	30.2%	32.8%
	$D_{in} \times 400_{0.4} \times 10$	30.0%	32.6%
	$D_{in} \times 400_{0.5} \times 10$	29.7%	32.7%
	$D_{in} \times 200_{0.5} \times 100 \times 10$	26.7%	28.6%
	$D_{in} \times 200 \times 100_{0.5} \times 10$	26.6%	25.2%
	$D_{in} \times 200_{0.5} \times 100_{0.5} \times 10$	27.4%	25.6%
Sparsity	$D_{in} \times 100 \times 10$	29.3%	34.3%
	$D_{in} \times 200 \times 10$	31.2%	34.3%
	$D_{in} \times 300 \times 10$	30.8%	34.3%
	$D_{in} \times 400 \times 10$	31.7%	36.5%
	$D_{in} \times 200 \times 100 \times 10$	30.6%	26.7%

plenty of valuable information which is beneficial for the classification. Nevertheless, with the continuous increase of the dimensionality q, our proposed method meets a performance drop. In practice, a too large dimensionality q may be corrupted by other information which has no contribution to our classification task.

Fig.4.7 plots the experimental results under different values of β. Note that, q is set to be 8, 50 and 10 for the three datasets, respectively. The accuracy obtained by our proposed method reaches the highest points when β is set to be 50, 60 and 70 for the three datasets, respectively, while a further rise would result in an obvious drop on the performance. The reason is that if β is too small, the label information of the latent variables cannot be fully informed, resulting in a lower performance. By contrast, a too larger value of β may have a too strong influence on the label information during the training, resulting in over-fitting.

4.6 Conclusion

In this chapter, a multi-view learning method based on the GPLVM is proposed to learn a shared variable in a subspace. Different from conventional GPLVM and its extensions, the proposed method simultaneously takes the back projection from multiple observations

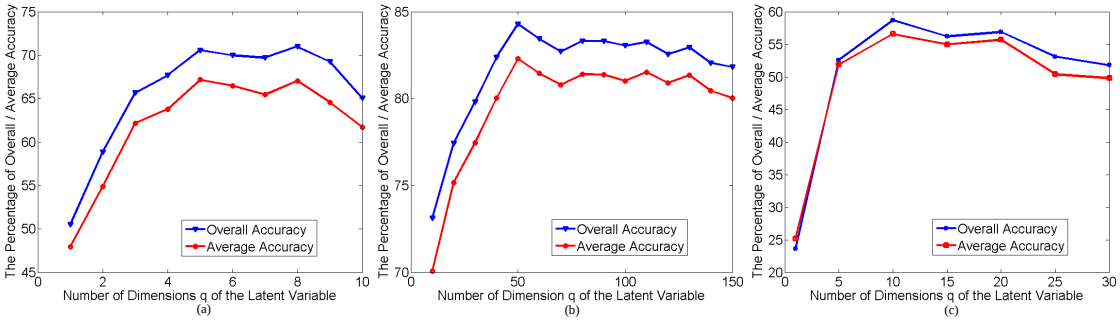


Figure 4.6. SAGP(SE). The percentage of the overall and average accuracies under the change of the dimensionality of the latent space. (a) Wiki Text-Image dataset. (b) AWA dataset. (c) NUS-WIDE-LITE dataset.

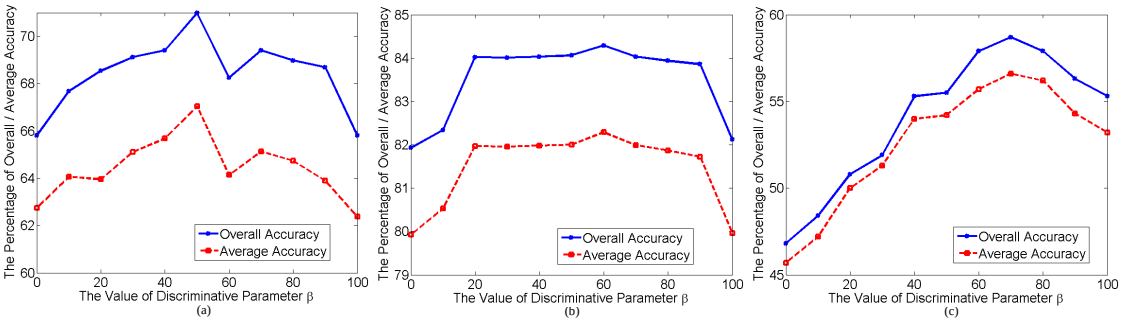


Figure 4.7. SAGP(SE). The percentage of the overall and average accuracies under the change of the β. (a) Wiki Text-Image dataset. (b) AWA dataset. (c) NUS-WIDE-LITE dataset.

to the shared variable into account. Thanks to this back constraint, we can get the latent variable simply but efficiently in the testing phase. Experimental results on three real-world datasets demonstrate the superiority of our approach compared with the state-of-the-art methods.

CHAPTER 5

MULTI-KERNEL SHARED GAUSSIAN PROCESS LATENT VARIABLE MODEL

As mentioned above, although SAGP achieves better performance compared with JSSL, it only uses the RBF to construct the covariance and separates the latent variable learning and classifier learning into two phases. In order to address these two issues, we further extend SAGP to MKSGP. In this chapter, we first introduce MKSGP including the encoder, decoder and prior, followed by its optimization, inference and computational complexity analysis. The experiments are then conducted to demonstrate its effectiveness. This method has been accepted in [41].

5.1 Problem Formulation

The framework of MKSGP is shown in Fig.5.1. Denote $\mathbf{Y} = \{\mathbf{Y}^v \in \mathbb{R}^{N\times D_v}\}$ $(v = 1, \cdots, V)$ be the observed modalities, where V is the number of views, N is the number of samples in each modality, D_v is the dimensionality of each view $\mathbf{Y}^v$, and $\mathbf{Y}^v = [\mathbf{y}_1^v, \cdots, \mathbf{y}_N^v]^T$. Being similar to the SGPLVM [14, 36] and SAGP [43], there is a latent variable $\mathbf{X} \in \mathbb{R}^{N\times q}$ shared among different views, instead of estimating a single one for each view as is done in GPLVM. In detail, the distribution of the observed data $\mathbf{Y}^v$ given the latent variable $\mathbf{X}$ is conditionally independent to other modalities. Thus, the likelihood of multiple modalities given $\mathbf{X}$ satisfies

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{v=1}^{V} p(\mathbf{Y}^v \mid \mathbf{X}, \boldsymbol{\theta}^v) = \sum_{v=1}^{V} \frac{1}{\sqrt{(2\pi)^{ND_v}\,|\mathbf{K}^v|^{D_v}}} \exp(-\frac{1}{2}\mathrm{tr}((\mathbf{K}_X^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T)) \quad (5.1)$$

where $\boldsymbol{\theta}^v$ and $\mathbf{K}_X^v$ are the kernel parameters and kernel matrix corresponding to the v-th view. In existing GP based works as well as SAGP, most of them exploit a certain kernel

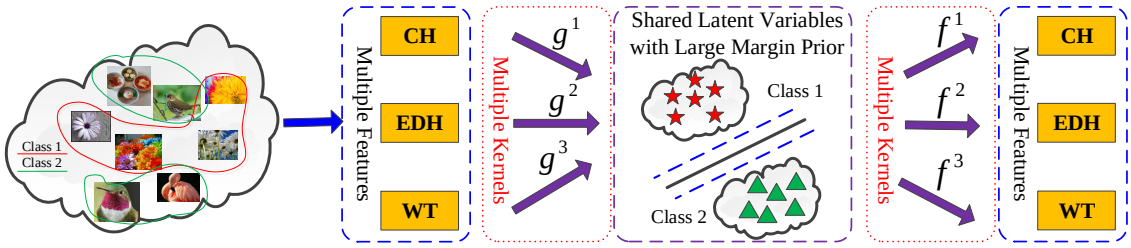


Figure 5.1. The Framework of MKSGP. Firstly, different types of features are obtained from a same object. The projection from these features to a shared latent variable is learned with multiple kernels. Simultaneously, projections from this latent variable to multiple observations are also learned with the multiple kernels. Particularly, to achieve the classification goal, a large margin prior is embedded into the latent variable.

function such as RBF to establish the kernel matrix $\mathbf{K}_X^v$. However, it may be an unreasonable estimation because of the complexity of the real-world data. Instead of assuming $\mathbf{K}_X^v$ to follow a determinative kernel function, we employ multiple kernels to automatically and adaptively construct the kernel matrix, which is a more reasonable way for various real-world dataset. Therefore, the kernel function $\mathbf{K}^v$ can be mathematically reformulated as follows.

$$\mathbf{K}_X^v = \sum_{k_f=1}^{K_f} w_{Xk_f}^v \mathbf{K}_{Xk_f}^v, \ s.t.\ w_{Xk_f}^v \geqslant 0, \ s.t. \sum_{k_f=1}^{K_f} w_{Xk_f}^v = 1 \tag{5.2}$$

where K_f is the number of selected kernel functions, and $w_{Xk_f}^v$ is the weight for the k_f-th kernel function $\mathbf{K}_{Xk_f}^v$ in the v-th view. Without loss generality, the number and types of the pre-defined kernel functions are set to be the same for all views.

To estimate the latent variable $\mathbf{X}$, we optimize the proposed model by minimizing

the negative log-likelihood function, as shown in Eq.(5.3).

$$
\begin{aligned}
&L_f = \sum_{v=1}^{V} L_f^v \\
&L_f^v = \frac{D^v}{2} N \log 2\pi + \frac{D^v}{2} \log |\mathbf{K}_X^v| + \frac{1}{2} \mathrm{tr}((\mathbf{K}_X^v)^{-1} \mathbf{Y}^v (\mathbf{Y}^v)^T) \\
&\mathbf{K}_X^v = \sum_{k_f=1}^{K_f} w_{Xk_f}^v \mathbf{K}_{Xk_f}^v, \\
&s.t. \quad w_{Xk_f}^v \geqslant 0, \sum_{k_f=1}^{K_f} w_{Xk_f}^v = 1 \ (v = 1, \cdots, V)
\end{aligned}
\tag{5.3}
$$

5.2 Priors for Classification

Since our purpose is to apply the proposed method to the classification task, some supervised priors would be embedded into the model to exploit the label information. Various kinds of discriminative priors are widely utilized in [15, 20, 43, 45, 65, 69, 85]. For instance, a discriminative prior is imposed on the SAGP to encourage the points belonging to the same category to be close, while the points belonging to different classes to be far. However, this prior requires an off-line classifier after training, which would make the model learning be not adaptive for this classifier. Differently, in MKSGP, the large margin prior is utilized to estimate the hyperplane for each class. In this way, we can not only learn our GPLVM model and but also get the classifier online.

Formally, the large margin prior with the latent variable $\mathbf{x}_i$ is represented as follows,

$$
\log p(\mathbf{x}_i) = \lambda \sum_{c=1}^{C} \mathfrak{L}(\mathbf{x}_i^T, \mathbf{t}_i, \mathbf{w}_c, b_c) \tag{5.4}
$$

where λ denotes the non-negative parameter to get a trade-off between Eq.(5.4) and Eq.(5.3), C is the number of categories, $\mathbf{w}_c$ is the hyperplane associated with the c-th category, b_c is its associated bias, and $\mathbf{t}_i = [t_i^1, \cdots, t_i^c, \cdots, t_i^C]$ means the label vector ($t_i^c = 1$ if the ground-truth label of the i-th sample is equal to c and otherwise $t_i^c = -1$).

Specifically, we define the large margin criterion as

$$
\mathfrak{L}(\mathbf{X}^T, \mathbf{T}, \mathbf{w}_c, b_c) = \frac{1}{2} \|\mathbf{w}_c\|_2^2 + \tau \sum_{i=1}^{N} l(\mathbf{x}_i^T, \mathbf{t}_i, \mathbf{w}_c, b_c) \tag{5.5}
$$

where $\mathbf{T}$ is the ground-truth matrix, τ is a non-negative parameter. To get a simple yet effective computation and keep a better smooth property, the quadratic hinge loss function is utilized in Eq.(5.5).

$$l(\mathbf{x}_i, \mathbf{t}_i, \mathbf{w}_c, b_c) = \left[\max(1 - t_i^c(\mathbf{w}_c^T \mathbf{x}_i + b_c), 0)\right]^2 \tag{5.6}$$

5.3 Back-Constraints

In this approach, we assume that each observation can be generated from the shared latent variable in a subspace. However, in the testing phase, the latent variable corresponding to the test sample is what we need. Being similar to SAGP, the auto-encoder based projection, followed by the multi-kernel learning, is proposed in MKSGP, to estimate a projection function to map the observations to the latent variable.

In detail, let $\boldsymbol{g}^v$ be a projection function from $\mathbf{Y}^v$ to $\mathbf{X}$, and $\boldsymbol{g}^v$ is embedded with a GP$\sim$(0, $\mathbf{K}_Y^v$) prior, where $\mathbf{K}_Y^v$ is the kernel matrix corresponding to the input $\mathbf{Y}^v$. Being similar to Eq.(5.2), the multiple kernel functions are utilized to establish $\mathbf{K}_Y^v$

$$\mathbf{K}_Y^v = \sum_{k_g=1}^{K_g} w_{Yk_g}^v \mathbf{K}_{Yk_g}^v, \; s.t. \; w_{Yk_g}^v \geqslant 0, \sum_{k_g=1}^{K_g} w_{Yk_g}^v = 1 \tag{5.7}$$

where K_g is the number of selected kernels, $\mathbf{K}_{Yk_g}^v$ is the k_g-th type of kernel function in v-th view, followed by its weight $w_{Yk_g}^v$. Since there are V number of views, the conditional distribution of the latent variable given these inputs $p(\mathbf{X} \mid \mathbf{Y} = \{\mathbf{Y}^v\}_{v=1}^V)$ is necessary to be calculated. In this thesis, the covariance of $p(\mathbf{X} \mid \mathbf{Y})$ is simply defined as $\mathbf{K}_Y = \sum_{v=1}^V \mathbf{K}_Y^v$, being similar to [43]. Therefore, the distribution of $p(\mathbf{X} \mid \mathbf{Y})$ can be represented as

$$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\gamma}) = \frac{1}{\sqrt{(2\pi)^{Nq} |\mathbf{K}_Y|^q}} \exp(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_Y^{-1}\mathbf{X}\mathbf{X}^T)) \tag{5.8}$$

where $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_{k_g}^v\}_{k_g,v}$ is the kernel parameter in function $\boldsymbol{g}$.

Consequently, the joint conditional probabilistic function over the shared variable $\mathbf{X}$ given the observations is represented as

$$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\gamma}, \boldsymbol{\theta}) \propto p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\gamma}) p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta}) \tag{5.9}$$

Finally, by adding the priors of $\mathbf{X}$ into Eq.(5.9), the objective functions is generalized as follows,

$$
\begin{aligned}
&\arg\min L = L_f + L_g + L_{pri}, \;\; L_f = \sum_{i=1}^{V} L_f^v \\
&\quad L_{pri} = \lambda \sum_{c=1}^{C} \mathfrak{L}(\mathbf{X}^T, \mathbf{T}, \mathbf{w}_c, b_c) \\
&\quad L_f^v = \frac{D^v}{2} N \log 2\pi + \frac{D^v}{2} \log |\mathbf{K}_X^v| + \frac{1}{2} \mathrm{tr}((\mathbf{K}_X^v)^{-1} \mathbf{Y}^v (\mathbf{Y}^v)^T) \\
&\quad L_g = \frac{1}{2} qN \log 2\pi + \frac{1}{2} q \log |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}((\mathbf{K}_Y)^{-1} \mathbf{X}\mathbf{X}^T) \\
&\quad \mathbf{K}_X^v = \sum_{k_f=1}^{K_f} w_{Xk_f}^v \mathbf{K}_{Xk_f}^v, \; s.t. \; w_{Xk_f}^v \geqslant 0, \; \sum_{k_f=1}^{K_f} w_{Xk_f}^v = 1, \\
&\quad \mathbf{K}_Y^v = \sum_{k_g=1}^{K_g} w_{Yk_g}^v \mathbf{K}_{Yk_g}^v, \; s.t. \; w_{Yk_g}^v \geqslant 0, \; \sum_{k_g=1}^{K_g} w_{Yk_g}^v = 1 \\
&\quad (v = 1, \cdots, V)
\end{aligned}
\tag{5.10}
$$

5.4 Optimization

Being similar to SAGP, the latent variable $\mathbf{X}$ in L_f and L_g is alternatively updated. Therefore, the objective function of Eq.(5.12) can be reformulated as follows.

$$
\begin{aligned}
&\arg\min L = L_f + L_g + L_{pri} \\
&\quad L_f = \sum_{i=1}^{V} L_f^v \\
&\quad L_{pri} = \lambda \sum_{c=1}^{C} \mathfrak{L}(\mathbf{X}^T, \mathbf{T}, \mathbf{w}_c, b_c) \\
&\quad L_f^v = \frac{D^v}{2} N \log 2\pi + \frac{D^v}{2} \log |\mathbf{K}_X^v| + \frac{1}{2} \mathrm{tr}((\mathbf{K}_X^v)^{-1} \mathbf{Y}^v (\mathbf{Y}^v)^T) \\
&\quad L_g = \frac{1}{2} qN \log 2\pi + \frac{1}{2} q \log |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}((\mathbf{K}_Y)^{-1} \mathbf{X}_g \mathbf{X}_g^T) \\
&\quad \mathbf{K}_X^v = \sum_{k_f=1}^{K_f} w_{Xk_f}^v \mathbf{K}_{Xk_f}^v, \; s.t. \; w_{Xk_f}^v \geqslant 0, \; \sum_{k_f=1}^{K_f} w_{Xk_f}^v = 1, \\
&\quad \mathbf{K}_Y^v = \sum_{k_g=1}^{K_g} w_{Yk_g}^v \mathbf{K}_{Yk_g}^v, \; s.t. \; w_{Yk_g}^v \geqslant 0, \; \sum_{k_g=1}^{K_g} w_{Yk_g}^v = 1 \\
&\quad (v = 1, \cdots, V) \;\; s.t. \quad \mathbf{X} = \mathbf{X}_g
\end{aligned}
\tag{5.11}
$$

To this end, the argument lagrange multiplier method (ALM) is utilized and the objective function is then transformed as

$$\begin{aligned}\arg\min L = &\frac{1}{2}\{qN\log 2\pi + q\log|\mathbf{K}_Y| + \mathrm{tr}((\mathbf{K}_Y)^{-1}\mathbf{X}_g\mathbf{X}_g^T)\}+ \\ &\sum_{v=1}^{V}\{\frac{D^v}{2}N\log 2\pi + \frac{D^v}{2}\log|\mathbf{K}_X^v| + \frac{1}{2}\mathrm{tr}((\mathbf{K}_X^v)^{-1}\mathbf{Y}^v(\mathbf{Y}^v)^T)\} \\ &+\lambda\sum_{c=1}^{C}\mathfrak{L}(\mathbf{X}^T,\mathbf{T},\mathbf{w}_c,b_c) + \langle\mathbf{Z},\mathbf{X}-\mathbf{X}_g\rangle + \frac{\mu}{2}\|\mathbf{X}-\mathbf{X}_g\|_F^2\end{aligned} \tag{5.12}$$

where $\mu > 0$ (we initially set it to 0.01 in the experiments) and $\mathbf{Z}$ is the Lagrange multiplier, and $\langle\cdot,\cdot\rangle$ denotes the inner product.

5.4.1 update the latent variable $\mathbf{X}$

The derivative with respect to $\mathbf{X}$ is calculated as

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L_f}{\partial \mathbf{X}} + \frac{\partial L_{pri}}{\partial \mathbf{X}} + \mu\left(\mathbf{X} - \mathbf{X}_g + \frac{\mathbf{Z}}{\mu}\right) \tag{5.13}$$

where $\frac{L_f}{\mathbf{X}} = \sum_{v=1}^{V}\frac{L_f^v}{\mathbf{X}}$. In order to get the gradient of L_f^v with respect to $\mathbf{X}$, the chain rule is utilized. Therefore,

$$\frac{\partial L_f^v}{\partial x_{ij}} = \mathrm{tr}\left((\frac{\partial L_f^v}{\partial \mathbf{K}_X^v})^T\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}\right) \tag{5.14}$$

5.4.2 update the latent variable $\mathbf{X}_g$

The derivative of Eq.(5.12) with respect to $\mathbf{X}_g$ is calculated as

$$\frac{\partial L}{\partial \mathbf{X}_g} = (\mathbf{K}_Y)^{-1}\mathbf{X}_g - \mu\left(\mathbf{X} - \mathbf{X}_g + \frac{\mathbf{Z}}{\mu}\right) \tag{5.15}$$

5.4.3 update the kernel parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$

Being similar to Eq.(5.14), the derivatives of Eq.(5.12) with respect to the kernel parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ are calculated by using the chain rule.

$$\frac{\partial L}{\partial \gamma_{k_g,i}^v} = \mathrm{tr}\left((\frac{\partial L_g}{\partial \mathbf{K}_Y})^T\frac{\partial \mathbf{K}_Y^v}{\partial \gamma_{k_g,i}^v}\right) \tag{5.16}$$

$$\frac{\partial L}{\partial \theta^v_{k_f,i}} = \mathrm{tr}\left((\frac{\partial L^v_f}{\partial \mathbf{K}^v_X})^T \frac{\partial \mathbf{K}^v_X}{\partial \theta^v_{k_f,i}}\right) \tag{5.17}$$

where $\gamma^v_{k_g,i}$ and $\theta^v_{k_f,i}$ are the i-th parameter in $\boldsymbol{\gamma}^v_{k_g}$ and $\boldsymbol{\theta}^v_{k_f}$, respectively.

5.4.4 update the weight of different types of kernel functions

Since there is no closed-form solution for the weight values $\boldsymbol{w}^v_X$ and $\boldsymbol{w}^v_Y$, we use the chain rule to get the gradient of the objective function with respect to the $\boldsymbol{w}^v_X$ and $\boldsymbol{w}^v_Y$, and then exploit the gradient decent algorithm to estimate the weight values. In detail,

$$\frac{\partial L}{\partial w^v_{Yk_g}} = \mathrm{tr}\left((\frac{\partial L_g}{\partial \mathbf{K}_Y})^T \frac{\partial \mathbf{K}^v_Y}{\partial w^v_{Yk_g}}\right) \tag{5.18}$$

$$\frac{\partial L}{\partial w^v_{Xk_f}} = \mathrm{tr}\left((\frac{\partial L^v_f}{\partial \mathbf{K}^v_X})^T \frac{\partial \mathbf{K}^v_X}{\partial w^v_{Xk_f}}\right) \tag{5.19}$$

Due to the constraints of $\sum_{k_g=1}^{K_g} w^v_{Yk_g} = 1$, $\sum_{k_f=1}^{K_f} w^v_{Xk_f} = 1$ and $w^v_{Yk_g} \geqslant 0$, $w^v_{Xk_f} \geqslant 0$, we set $w^v_{Yk_g} = 0$ and $w^v_{Xk_f} = 0$ if their values are smaller than 0. Then we normalize the sum of $w^v_Y = \{w^v_{Yk_g}\}_{k_g}$ or $w_X = \{w^v_{Xk_f}\}_{k_f}$ to 1 after each iteration.

5.4.5 update the Lagrange multiplier Z and parameter μ

The Lagrange multiplier $\mathbf{Z}$ and parameter μ are updated following Eq.(5.20).

$$\begin{aligned} \mathbf{Z}_{t+1} &= \mathbf{Z}_t + \mu_t(\mathbf{X} - \mathbf{X}_g) \\ \mu_{t+1} &= \min(\mu_{max}, \rho\mu_t) \end{aligned} \tag{5.20}$$

where $\mathbf{Z}_t$ and $\mathbf{Z}_{t+1}$ are the current and updated value, respectively; so do μ_t and μ_{t+1}; μ_{max} is the maximization of μ (it is typically set to 1000); ρ is the a constant, keeping the step size of μ (it is typically set to 1.1).

5.4.6 Inference

Refer to [56] in Chapter 2, the latent variable $\mathbf{x}^*$ can be obtained through a simple yet effective way when a testing sample is given.

$$\mathbf{x}^* = (\sum_v \mathbf{K}_{y^{v*}Y^v})(\sum_v \mathbf{K}^v_Y)^{-1}\mathbf{X} \tag{5.21}$$

Algorithm 5.1 Multi-kernel Shared Gaussian Process Latent Variable Model (MKSGP)

Input: the observed training data $\mathbf{Y}$, testing data $\mathbf{y}^*$, and the prior parameter λ and τ, and dimension q of the latent variable.

Output: the latent variables $\mathbf{X}$ and $\mathbf{x}^*$.

Initialization: initialize the latent variable $\mathbf{X}$ using LDA.

1: **while** not converged **do**

2: **Update $\mathbf{X}$ and parameters $\boldsymbol{\theta}$**: optimize the latent variable $\mathbf{X}$ and parameter $\{\boldsymbol{\theta}^v\}_v^V$ using Eq.(5.13) and Eq.(5.17)

3: **Update $\mathbf{X}_g$ and parameters $\boldsymbol{\gamma}$**: compute the latent variable $\mathbf{X}_g$ and parameter $\{\boldsymbol{\gamma}^v\}_v^V$ using Eq.(5.15) and Eq.(5.16)

4: **Update the weight values $\mathbf{w}_Y$ and $\mathbf{w}_X$**: estimate the weights of each type of the kernel function using Eq.(5.18) and Eq.(5.19)

5: **Update Lagrange multiplier $\mathbf{Z}$ and $\boldsymbol{\mu}$**: update $\mathbf{Z}$ and $\boldsymbol{\mu}$ using Eq.(5.20).

6: **end while**

7: **Test**: get the latent variable of the test sample by Eq.(5.21).

where $\mathbf{K}_{y^{v*}Y^v}$ denotes the kernel matrix evaluated on all pairs of training and testing points. The optimization and inference of the proposed method is shown in Algorithm 5.1.

5.5 Computational Complexity Analysis

As RBF is complex enough in contrast to remaining kernel functions, we use it as the typical one in our complexity analysis per iteration. Without loss the generality, let $D = \max\{D_v\}_{v=1}^V$ and the number of iterations in the gradient decent technique be t. Referring to the optimization, it is easy to see that the main complexity exists in Eq.(5.13) and Eq.(5.14). It is easy to gain that the complexity of $\frac{\partial L_f^v}{\partial \mathbf{K}_X^v}$ is $O(N^3 + N^2 D)$ and the complexity of $\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}$ is $O(Nq)$. Therefore, the complexity of Eq.(5.14) is $O((N^3 + N^2 D) + (Nq + N^2))$. From Eq.(5.13) we can see that $\frac{\partial \mathbf{K}_X^v}{\partial x_{ij}}$ should be computed with N^2 times. Thus the complexity of Eq.(5.13) is $O(V(N^2 D + N^2(Nq + N^2)) + qNC) = O(V(N^2 D + N^3 q + N^4) + qNC)$. Since N is much larger than q, D and C, the complexity is nearly $O(tVN^4)$. Despite that the complexity of our training step is relatively high, it is quite efficient at the testing step, which is only is only $O(VDN + N^2 + Nq)$ according to Eq.(5.21).

5.6 Experiments

In this section, we conduct experiments on AWA, NUS-WIDE-LITE, and biomedical (Health VS DM) datasets to demonstrate the effectiveness of MKSGP.

5.6.1 Experimental Settings

To quantatively evaluate the superiority of MKSGP, we conduct the experiments compared with CCA, randomized nonlinear CCA (RCCA) [46], DCCA [3], DCCAE [70], JSSL [44], AWFA [76], GPLVM, DGPLVM [69], DSGPLVM [15] and SAGP [43]. Since there are multiple kernels in the covariance construction for MKSGP, six kinds of kernels containing **Linear, RBF, Polynomial, Rational Quadratic (Ratquad), Multilayer Perceptron (MLP) and Matern** kernels are used. Note that, for each kernel function, our method can automatically estimate its corresponding weight. All experiments are implemented by using the MATLAB R2014b on a CPU with 32G RAM.

In experiments, we tune the discriminative prior λ and τ through 3-fold cross-validation by using small subsets selected from the training data. For the parameter τ, it is sensitive and we tune it very carefully for different datasets. According to our cross-validation, we set it to 0.1, 2, and 3 on the AWA, NUS-WIDE-LITE and Biomedical datasets, respectively.

5.6.2 Experimental Results

AWA dataset: Tab.5.1 displays the classification results on the AWA dataset obtained by different methods, when the q ranges from 40 to 130. Note that in this experiment λ and τ are set to be 40 and 0.1, respectively. It is easy to observe that the proposed method is much superior to CCA, RCCA, DCCA, DCCAE, and GPLVM, achieving a remarkable improvement. Compared with JSSL, AWFA, DGPLVM, DSGPLVM, and SAGP, MKSGP is also competitive. Concretely, there is almost more or less 2% enhancement on the overall and average accuracies. The main reason is that MKSGP jointly takes the multi-kernel and online classifier learning into account, which are beneficial for the performance improvement.

Table 5.1. The overall and average accuracies on the AWA dataset computed by various methods under the change of the dimensionality of the latent variable from 40 to 130.

		Dimensionality									
Methods	Result	d_{out}=40	d_{out}=50	d_{out}=60	d_{out}=70	d_{out}=80	d_{out}=90	d_{out}=100	d_{out}=110	d_{out}=120	d_{out}=130
JSSL	Overall	83.8%									
	Average	81.7%									
AWFA	Overall	83.4%									
	Average	81.3%									
CCA	Overall	71.8%	73.6%	74.3%	74.8%	75.1%	75.1%	75.5%	73.8%	74.1%	74.1%
	Average	69.2%	71.1%	72.8%	72.4%	72.8%	72.7%	73.2%	71.6%	72.0%	71.8%
RCCA	Overall	78.1%	78.8%	78.9%	79.2%	79.3%	79.3%	79.4%	79.4%	79.2%	79.3%
	Average	75.6%	76.3%	76.5%	76.8%	76.9%	76.8%	76.9%	77.1%	76.8%	76.9%
GPLVM	Overall	70.9%	71.9%	71.4%	71.1%	69.1%	72.1%	71.4%	71.8%	72.1%	71.6%
	Average	69.5%	70.6%	69.9%	70.3%	67.2%	69.7%	70.9%	70.7%	71.6%	70.8%
DGPLVM	Overall	80.1%	82.0%	80.2%	78.8%	77.6%	76.5%	75.4%	75.0%	74.5%	73.8%
	Average	78.3%	80.1%	78.6%	77.3%	76.1%	75.0%	74.0%	73.6%	73.2%	72.3%
DSGPLVMI	Overall	81.8%	82.9%	82.8%	81.7%	82.8%	82.7%	82.4%	82.3%	82.4%	82.3%
	Average	**80.0%**	81.0%	80.9%	79.8%	81.1%	80.8%	80.6%	80.5%	80.7%	80.8%
DSGPLVMS	Overall	82.0%	83.4%	83.1%	82.4%	82.5%	82.7%	82.7%	82.5%	83.2%	82.7%
	Average	**80.0%**	81.4%	81.5%	80.7%	80.8%	80.8%	80.8%	81.4%	81.5%	81.2%
DCCA	Overall	76.6%	78.3%	77.5%	77.9%	76.9%	77.9%	78.2%	78.4%	78.1%	78.3%
	Average	74.4%	76.3%	75.3%	76.0%	74.6%	75.6%	76.0%	76.1%	75.8%	76.1%
DCCAE	Overall	76.7%	77.1%	76.9%	76.7%	76.8%	77.0%	76.9%	76.9%	76.9%	76.8%
	Average	74.8%	75.0%	74.8%	74.9%	74.9%	74.9%	74.8%	74.8%	74.8%	74.7%
SAGP(IE)	Overall	82.0%	83.6%	83.0%	82.7%	82.2%	81.8%	81.4%	81.1%	80.4%	79.8%
	Average	79.9%	81.5%	81.1%	80.7%	80.3%	79.8%	79.5%	79.4%	78.7%	78.0%
SAGP(SE)	Overall	**82.4%**	84.3%	83.4%	82.7%	83.3%	83.3%	83.1%	83.3%	82.6%	82.9%
	Average	**80.0%**	82.3%	81.5%	80.8%	81.4%	81.4%	81.0%	81.5%	80.9%	81.4%
MKSGP	Overall	81.6%	**85.3%**	**85.4%**	**85.3%**	**85.4%**	**85.2%**	**85.2%**	**84.9%**	**85.0%**	**85.1%**
	Average	78.7%	**83.2%**	**83.4%**	**83.3%**	**83.3%**	**83.0%**	**83.0%**	**82.6%**	**82.8%**	**83.0%**

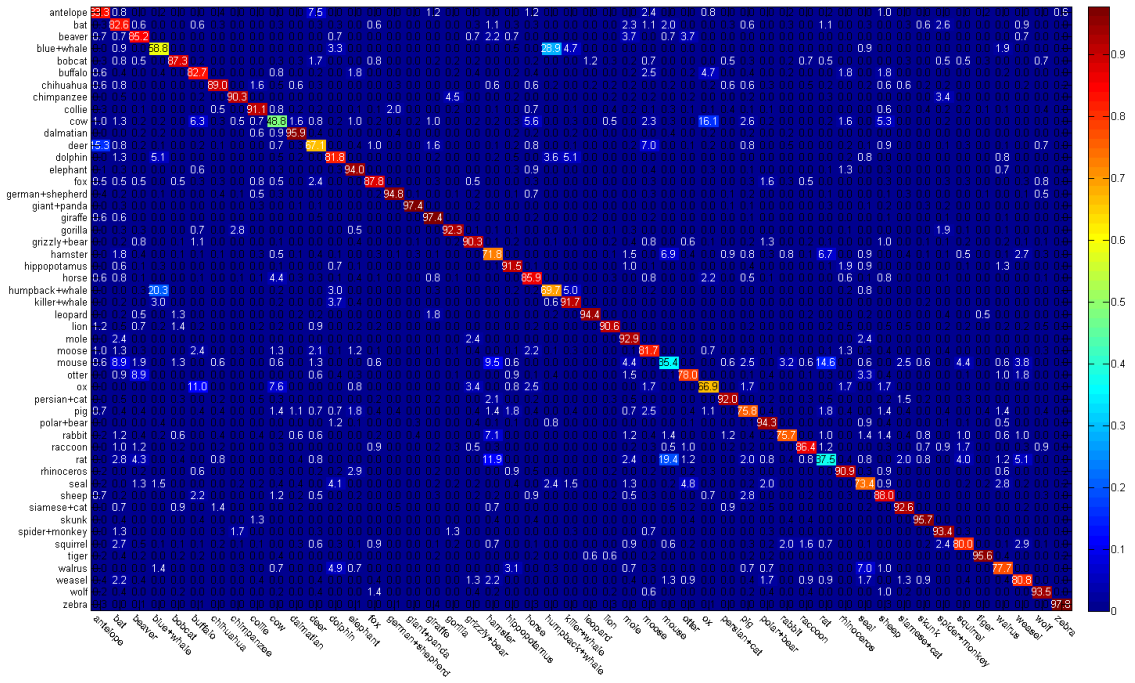


Figure 5.2. The confusion matrix conducted on the AWA dataset when the dimensionality of $\mathbf{X}$ is set to 60.

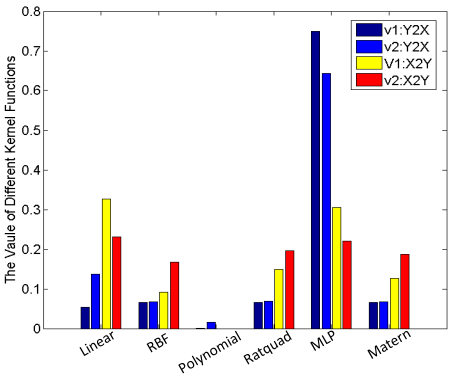


Figure 5.3. The weight values of various kinds of kernels in multiple mappings on the AWA dataset. Here 'v1:Y2X' denotes the weights from the observed data to $\mathbf{X}$ in the first view. So do other symbols.

Table 5.2. The overall and average accuracies on the NUS-WIDE-LITE dataset computed by various methods when q changes from 1 to 30.

		Dimensionality						
Methods	Result	d_{out}=1	d_{out}=5	d_{out}=10	d_{out}=15	d_{out}=20	d_{out}=25	d_{out}=30
JSSL	Overall	47.2%						
	Average	48.9%						
AWFA	Overall	51.3%						
	Average	50.1%						
CCA	Overall	15.1%	30.6%	35.6%	36.7%	37.4%	37.7%	37.9%
	Average	14.8%	29.6%	34.2%	34.7%	35.9%	35.3%	36.6%
RCCA	Overall	17.8%	35.6%	40.4%	42.8%	43.6%	44.3%	44.0%
	Average	18.9%	32.9%	38.4%	39.6%	41.4%	42.4%	41.1%
GPLVM	Overall	16.3%	33.1%	36.7%	40.7%	42.1%	40.0%	40.3%
	Average	14.9%	32.8%	34.8%	39.8%	40.5%	37.9%	39.6%
DGPLVM	Overall	19.6%	43.9%	51.2%	50.6%	51.9%	46.2%	45.1%
	Average	21.5%	49.9%	52.8%	53.4%	54.0%	45.5%	44.8%
DSGPLVMI	Overall	19.1%	24.5%	32.9%	33.8%	33.5%	32.6%	31.9%
	Average	17.6%	30.7%	34.9%	35.9%	37.5%	34.3%	36.0%
DSGPLVMS	Overall	25.4%	46.1%	54.8%	54.4%	53.8%	50.5%	48.9%
	Average	25.0%	52.5%	56.6%	56.1%	55.6%	49.6%	48.6%
DCCA	Overall	14.6%	27.2%	31.5%	33.4%	34.0%	34.1%	35.0%
	Average	14.4%	26.2%	31.5%	32.5%	32.2%	32.3%	33.6%
DCCAE	Overall	17.1%	29.8%	35.4%	36.4%	36.2%	37.3%	36.5%
	Average	17.1%	28.4%	33.7%	34.4%	35.0%	36.1%	36.0%
SAGP(IE)	Overall	19.8%	29.1%	32.4%	34.7%	34.6%	37.8%	33.2%
	Average	17.7%	33.7%	37.2%	36.4%	34.2%	37.5%	34.0%
SAGP(SE)	Overall	23.6%	52.6%	58.7%	56.2%	56.9%	53.1%	51.8%
	Average	25.2%	51.9%	56.6%	55.0%	55.7%	50.4%	49.8%
MKSGP	Overall	**26.2%**	**59.6%**	**62.2%**	**62.5%**	**63.0%**	**62.6%**	**58.6%**
	Average	**27.1%**	**54.9%**	**58.6%**	**59.2%**	**59.9%**	**59.6%**	**56.4%**

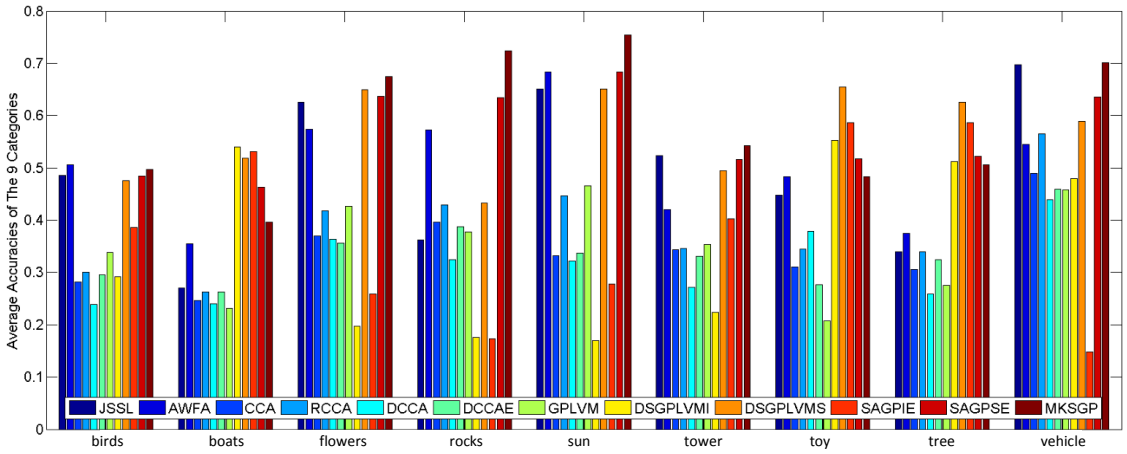


Figure 5.4. The average accuracy of selected 9 classes computed by different approaches.

Fig.5.2 further displays the confusion matrix when the dimensionality q is set to be 60. As we can see, the accuracy for each category gained by our presented method MKSGP is larger than 70% for most classes. Furthermore, there are as many as 20 classes whose percentages outperform 90%.

Fig.5.3 shows the weight values belonging to different types of kernels. From Fig.5.3 we can observe that MKSGP can adaptively estimate the weight values for different views and different types of projections (encoder and decoder). The 'MLP' (Multilayer Perceptron) kernel function gains the highest weights for both views in the encoder. In the decoding part, the 'Linear', 'RBF', 'Ratquad' (Rational Quadratic), 'MLP' and 'Matern' kernels are mainly used for establishing the covariance matrices, while 'Polynomial' almost has no contribution.

NUS-WIDE-LITE dataset: The overall and average percentages calculated by various single-view or multi-view methods are tabulated in Tab.5.2, when the dimensionality q ranges from 1 to 30. Note that in this experiment we set λ and τ to be 70 and 2, respectively. It is easy to see that MKSGP achieves the best performance compared with other comparison methods. In comparison to CCA, RCCA, GPLVM, DSGPLVMI, DCCA, DCCAE and SAGPIE, MKSGP has almost 15% improvement on the two indexes. In contrast to DGPLVM, DSGPLVMS and SAGPSE, our strategy MKSGP also obtains the remarkable enhancement

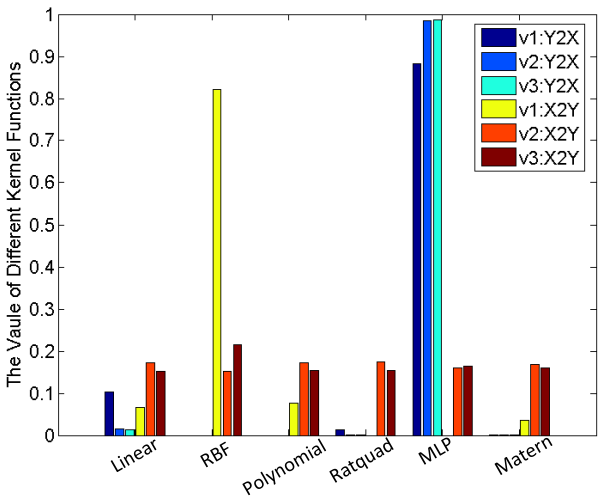


Figure 5.5. The weight values of various kinds of kernels in multiple mappings on the NUS-WIDE-LITE dataset.

since it jointly takes the multi-view correlation, multi-kernel learning, the auto-encoder back constraint and large margin prior into account. Fig.5.4 shows the classification accuracy for different classes, when q is set to be 10. From Fig.5.4 we can see that our presented strategy obtains the best results in most of categories including birds, flowers, rocks, sun, tower and vehicle.

Fig.5.5 shows the weight values belonging to different types of kernels, when $q = 10$, $\lambda = 70$ and $\tau = 2$, respectively. We can see that 'RBF' plays the main role at the decoding step in the first view, followed by 'Polynomial', 'Linear' and 'Matern' kernels. For the second and third views, these six kernels give a similar contribution. On the other hand, the 'MLP' kernel is mainly utilized to construct the covariance matrices at the encoding step in all views. Also, the 'Linear' kernel has more or less contribution to model the back constraint.

Biomedical dataset: In this experiment, we use the accuracy, sensitivity and specificity to evaluate our proposed method and the experimental results are displayed in Tab.5.3. As DCCA and DCCAE need a large number of training data, they do not perform well on this dataset. Therefore, we ignore these two methods in this experiment. Being similar to the setting in the NUS-WIDE-LITE dataset, the facial and sublingual features are empirically

Table 5.3. The accuracy, sensitivity and specificity values computed by various approaches on the Biomedical dataset when the number of training instances is 30, 40, and 50, respectively.

	Number of Training Samples								
Num	num=30			num=40			num=50		
Methods	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
CCA	76.1%	78.4%	73.8%	76.7%	78.6%	74.9%	79.5%	80.0%	79.1%
RCCA	76.3%	81.1%	71.5%	77.7%	80.7%	74.8%	78.6%	81.0%	76.2%
JSSL	79.8%	78.7%	**81.0%**	81.5%	**83.1%**	79.9%	82.8%	82.0%	83.6%
AWFA	79.6%	78.7%	80.6%	79.8%	75.6%	**83.9%**	81.2%	77.2%	**85.0%**
GPLVM	73.7%	71.7%	75.7%	75.8%	73.6%	78.0%	78.1%	78.5%	77.8 %
DGPLVM	74.9%	76.8%	73.1%	78.7%	79.5%	78.0%	78.8%	79.2%	78.4 %
DSGPLVMI	75.2%	77.7%	72.7%	76.8%	79.0%	74.7%	75.7%	78.3%	73.2 %
DSGPLVMS	76.9%	79.6%	74.2%	80.8%	82.9%	78.9%	81.7%	84.2%	79.3%
SAGP(IE)	76.7%	77.9%	75.6%	76.7%	81.4%	72.2%	78.4%	82.1%	74.9%
SAGP(SE)	79.0%	78.9%	79.2%	79.2%	81.3%	77.2%	82.5%	83.9%	81.2%
MKSGP	**80.7%**	**81.1%**	80.2%	**82.1%**	82.5%	81.8%	**83.6%**	**84.8%**	82.4%

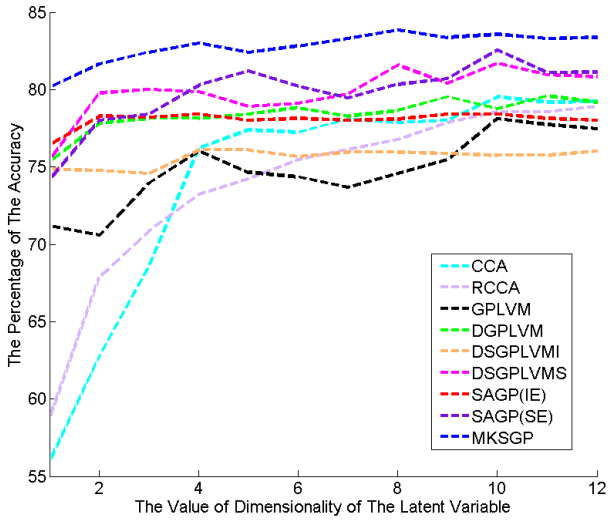


Figure 5.6. The accuracy obtained by different methods under different values of the dimensionality of the latent variable on the Biomedical dataset.

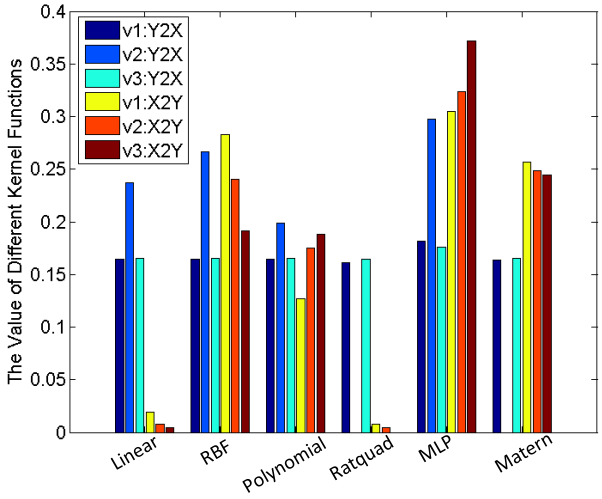


Figure 5.7. The weight values of various kinds of kernels in multiple mappings on the Biomedical dataset.

concatenated as a single one for CCA and RCCA. Furthermore, except for JSSL and AWFA, q is set to 10 in all methods and $\lambda = 30$ and $\tau = 3$ for MKSGP, respectively. Compared with CCA, RCCA, GPLVM, DSGPLVMI and SAGPI, MKSGP is far superior. In contrast to DGPLVM and DSGPLVMS, our proposed approach also enjoys a large performance improvement. In comparison to SAGPS, JSSL and AWFA, MKSGP is still competitive. Additionally, we further plots the accuracy curves in Fig.5.6 when q ranges. Here number of training samples for each class is 50. We can see that our proposed method MKSGP get outstanding performance in all situations.

Fig.5.7 displays the weight values of different kernels from and to the shared space, when the training number is 50 and $q = 10$. As we can see, weights corresponding to the six kernels are quite similar in the first and third views of the encoder. In the second view, 'RBF' and 'MLP' kernel functions play the main role, followed by the 'Linear' and 'Polynomial' kernels. In the decoding part, 'RBF', 'Polynomial', 'MLP' and 'Matern' kernels obtain higher weights in three views, while 'Linear' and 'Ratquad' kernels give a slight contribution.

5.6.3 Parameter analysis

Being similar to SAGP, we discuss the parameter sensitivity of MKSGP.

Fig.5.8 shows the influence of τ on the overall and average classification accuracy. Here, q is set to 60, 10, and 10, and λ is set to 40, 70, and 30 for the three datasets, respectively. It is easy to see that there is a performance increase with the rise of τ. However, with the continuous increase of τ, MKSGP meets a slight performance degradation. In practice, a too large value of τ may corrupt the influence of the constraint $\frac{1}{2}\|\mathbf{w}_c\|_2^2$, resulting in learning an inferior classifier.

The influence of λ is displayed in Fig.5.9. Note that q is set to 60, 10, and 10, and τ is set to 0.1, 2, and 3 for the three datasets, respectively. With the increase of λ, MKSGP has a performance increase, while a further increase leads to a noticeable drop on the performance. In fact, λ makes a trade-off between the discriminative term and the latent variable learning model. If it is too large, MKSGP may meet an over-fitting phenomenon since the model pays

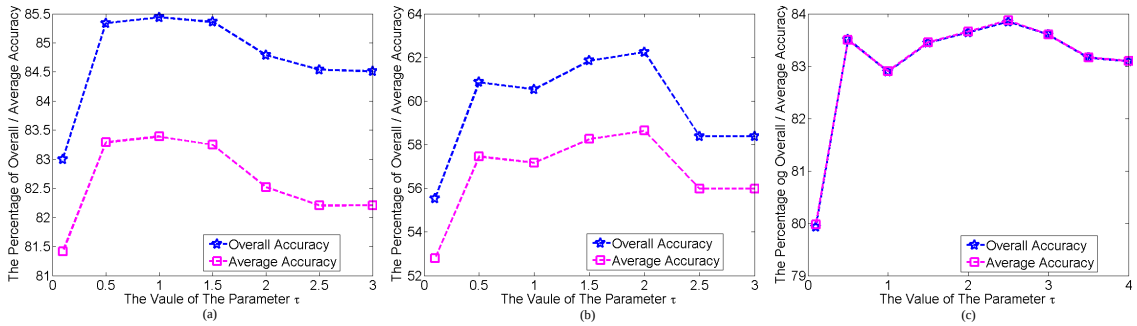


Figure 5.8. MKSGP. The percentage of the overall accuracy and average accuracy with the change of the value of τ on the three datasets.

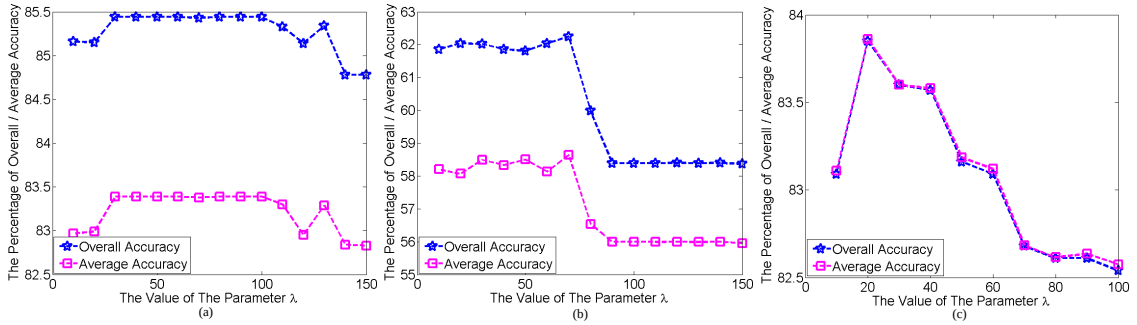


Figure 5.9. MKSGP. The percentage of the overall accuracy and average accuracy with the change of the value of λ on the three datasets.

much attention on exploiting the label information. By contrast, the semantic information cannot be fully utilized if λ is too small.

5.6.4 Time cost analysis

To obtain the latent variable at the testing stage, only Eq.(5.21) should be computed. Its time complexity is only $O(VDN + N^2 + Nq)$. In order to further demonstrate the

Table 5.4. The time cost at the testing stages on different datasets

Dataset	AWA	NUS-WIDE-LITE	Biomedical
Time(sec.)	7.7787×10^{-4}	7.0064×10^{-4}	3.1032×10^{-5}

effectiveness of the proposed method, we record the time consumption on the three datasets at the testing stage. Note that the dimension of the latent variable is set to 60, 10 and 10 for the AWA, NUS-WIDE-LITE and Biomedical datasets, respectively. The time cost for each testing sample is shown in Tab. 5.4. It is easy to observe that MKSGP can obtain the latent variable and predict the label in real-time.

5.7 Conclusion

In this chapter, an extension of SAGP is proposed. Different from SAGP which only uses a certain kernel function, multi-kernel learning is introduced to construct the covariance matrices in both encoder and decoder, which can better model the data with complex distributions. Furthermore, the large margin prior is also imposed on the latent variable to learn the classifier online, which is more adaptive for our data and model.

CHAPTER 6

HIERARCHICAL MULTI-VIEW MULTI-FEATURE FUSION

Although JSSL, SAGP and MKSGP have been proposed to process the multi-view data and achieve satisfactory performance in classification, they can not be directly applied to the multi-view and multi-feature data. Here the multi-view and multi-feature data means an object can be represented with multiple views, and each view can also be represented with multiple features. To address this problem, we propose a generative and probabilistic model for multi-view and multi-feature classification in this chapter, which can hierarchically exploit the correlation among them. This method is accepted in [39].

6.1 The Hierarchical probabilistic Model

The framework and graphic model of the proposed method are shown in Fig.6.1. As we can see, an object is observed from J views, and the j-th ($j \in \{1, \cdots, J\}$) view is represented by K_j types of features. Specifically, the data and the label of the i-th sample can be represented as $\{\mathbf{x}_{ijk_j} \in \mathbb{R}^{D_{jk_j}}\}_{j,k_j=1}^{J,K_j}$ and a one-hot categorical variable $\mathbf{z}_i \in \mathbb{R}^P$, respectively, where D_{jk_j} is the dimension of the k_j-th feature in the j-th view, P is the number of the classes, and $\mathbf{z}_i$ satisfies $z_{pi} \in \{0, 1\}$ and $\sum_{p=1}^{P} z_{pi} = 1$.

Then some probability assumptions for these variables are made to construct our hierarchical model. Firstly, the categorical distribution is introduced for the categorical variable $\mathbf{z}_i$, which has the following form:

$$p(\mathbf{z}_i \mid \boldsymbol{\theta}_{\boldsymbol{Z}}) = \prod_{p=1}^{P} \pi_p^{z_{pi}} \tag{6.1}$$

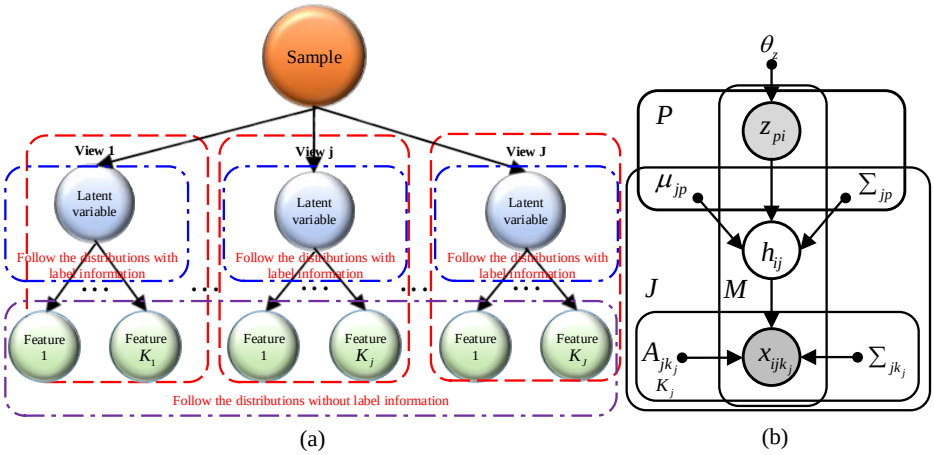


Figure 6.1. (a) The framework of the proposed method, where the number of the latent variables is equal to the number of observed views. (b) The probabilistic framework of the proposed method.

where $\boldsymbol{\theta_Z} = \{\pi_p\}_{p=1}^{P}$, $z_{pi} = 1$ if the i-th sample belongs to the p-th class, otherwise $z_{pi} = 0$ and the variable $\pi_p \in [0, 1]$ follows $\sum_{p=1}^{P} \pi_p = 1$, which means the probability of a sample belonging to its corresponding category.

In order to exploit the discriminative information, a latent variable $\mathbf{h}_{ij} \in \mathbb{R}^{D_j}$ corresponding to the j-th view of the i-th sample is then learned by imposing the ground-truth label on it. Specifically, for distinctive categories, the distributions of the latent variables belonging to different views are different, greatly exploiting the complementary information across multiple views. The most common and simple assumption for $\mathbf{h}_{ij}$ is that

$$p(\mathbf{h}_{ij} \mid \mathbf{z}_i, \boldsymbol{\theta_H}) = \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_{ij} \mid \boldsymbol{\mu}_{jp}, \boldsymbol{\Sigma}_{jp})^{z_{pi}} \tag{6.2}$$

where $\boldsymbol{\theta_H} = \{\boldsymbol{\mu}_{jp}, \boldsymbol{\Sigma}_{jp}\}_{j,p=1}^{J,P}$, meaning that its distribution for the p-th category is a Gaussian distribution with mean $\boldsymbol{\mu}_{jp}$ and covariance matrix $\boldsymbol{\Sigma}_{jp}$.

In general, it is reasonable to assume that multiple features are the projections from a shared variable through different mapping functions. Thus, in the proposed model, a mapping matrix for each feature in a same modality is learned to transform the the latent variable

$\mathbf{h}_{ij}$ to the observed data $\mathbf{x}_{ijk_j}$ by a linear Gaussian model, which can be presented as following equation:

$$p(\mathbf{x}_{ijk_j} \mid \mathbf{h}_{ij}, \boldsymbol{\theta}_{\boldsymbol{X}}) = \mathcal{N}(\mathbf{x}_{ijk_j} \mid \mathbf{A}_{jk_j}\mathbf{h}_{ij} + \mathbf{b}_{jk_j}, \boldsymbol{\Sigma}_{jk_j}) \tag{6.3}$$

where $\boldsymbol{\theta}_{\boldsymbol{X}} = \{\mathbf{A}_{jk_j}, \mathbf{b}_{jk_j}, \boldsymbol{\Sigma}_{jk_j}\}_{j,k_j=1}^{J,K_j}$, $\mathbf{A}_{jk_j}$ is the learned mapping matrix, $\mathbf{b}_{jk_j}$ is the bias and $\boldsymbol{\Sigma}_{jk_j}$ denotes the covariance matrix.

Moreover, we also make some reasonable conditional independence assumptions about different features and different views, including

$$\begin{aligned} p(\{\mathbf{x}_{ijk_j}\}_{k_j=1}^{K_j} \mid \mathbf{h}_{ij}, \boldsymbol{\theta}_{\boldsymbol{X}}) &= \prod_{k_j=1}^{K_j} p(\mathbf{x}_{ijk_j} \mid \mathbf{h}_{ij}, \boldsymbol{\theta}_{\boldsymbol{X}}) \\ p(\{\{\mathbf{x}_{ijk_j}\}_{k_j=1}^{K_j}, \mathbf{h}_{ij}\}_{j=1}^{J} \mid \boldsymbol{\theta}_{\boldsymbol{X}}, \boldsymbol{\theta}_{\boldsymbol{H}}, \mathbf{z}_i) &= \prod_{j=1}^{J} p(\{\mathbf{x}_{ijk_j}\}_{k_j=1}^{K_j}, \mathbf{h}_{ij} \mid \boldsymbol{\theta}_{\boldsymbol{X}}, \boldsymbol{\theta}_{\boldsymbol{H}}, \mathbf{z}_i) \end{aligned} \tag{6.4}$$

In order to acquire a simple representation derivation, let $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{M}$, $\mathbf{H} = \{\mathbf{h}_{ij}\}_{i,j=1}^{M,J}$ and $\mathbf{X} = \left\{\mathbf{x}_{ijk_j}\right\}_{i,j,k_j=1}^{M,J,K_j}$. Taking the aforementioned independent and identically distributed (i.i.d.) assumption into account, the join distribution w.r.t. all variables is obtained:

$$P(\mathbf{X}, \mathbf{Z}, \mathbf{H} \mid \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) = \prod_{i=1}^{M}\{p(\mathbf{z}_i \mid \boldsymbol{\theta}_Z) \prod_{j=1}^{J}\{p(\mathbf{h}_{ij} \mid \mathbf{z}_i, \boldsymbol{\theta}_H) \prod_{k_j=1}^{K_j} p(\mathbf{x}_{ijk_j} \mid \mathbf{h}_{ij}, \boldsymbol{\theta}_X)\}\} \tag{6.5}$$

which is a probabilistic hierarchical model. Generally speaking, it is infeasible to estimate the covariance matrix $\boldsymbol{\Sigma}_{jk_j}$ and $\boldsymbol{\Sigma}_{jp}$, when the dimensions of the features and the latent variables are large. To avoid overfitting, $\boldsymbol{\Sigma}_{jp}$ and $\boldsymbol{\Sigma}_{jk_j}$ can be set to be $\sigma_{jp}^2\mathbf{I}$ and $\sigma_{jk_j}^2\mathbf{I}$ in this case, where σ_{jp} and σ_{jk_j} are two 1-D variables to control their variances of all dimensions, and $\mathbf{I}$ is identical matrix.

To estimate the parameters of this probabilistic method, the log-likelihood function w.r.t. all variables should be optimized. Since it is difficult to directly observe the latent variable $\mathbf{H}$, the log-likelihood function only related to the multi-view and multi-feature data $\mathbf{X}$ and its label variable $\mathbf{Z}$ is considered. Therefore, the objective function is

$$\arg\min \log P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) \tag{6.6}$$

However, it is quite different between marginalizing $\mathbf{H}$ in Eq.(6.5) and optimizing the objective function (6.6). Fortunately, the Expectation-Maximization(EM) [8] algorithm can be readily utilized for efficiently solving this kind of problem with latent variables.

6.2 Optimization

As analyzed above, the EM algorithm, which is a two-stage iterative optimization technique for finding maximum likelihood solutions, is employed to estimate the model parameters. Specifically, the posterior probability of latent variable $\mathbf{H}$ will be calculated in E-step, followed by the estimation of the value of parameters $\boldsymbol{\theta}_Z$, $\boldsymbol{\theta}_H$ and $\boldsymbol{\theta}_X$ in M-step.

E Step: Primarily, we use the current values of all parameters to evaluate the posterior probabilities of $\mathbf{H}$. The log-posterior function

$$\begin{aligned}
&\log P(\mathbf{H} \mid \mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) \propto \log P(\mathbf{X}, \mathbf{H}, \mathbf{Z} \mid \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) \\
&\propto \sum_{i=1}^{M}\{\sum_{j=1}^{J}\{-\frac{1}{2}\mathbf{h}_{ij}^T(\sum_{p=1}^{P} z_{pi}\boldsymbol{\Sigma}_{jp}^{-1} + \sum_{k_j=1}^{K_j}\mathbf{A}_{jk_j}^T\boldsymbol{\Sigma}_{jk_j}^{-1}\mathbf{A}_{jk_j})\mathbf{h}_{ij} \\
&+ \mathbf{h}_{ij}^T(\sum_{p=1}^{P} z_{pi}\boldsymbol{\Sigma}_{jp}^{-1}\boldsymbol{\mu}_{jp} + \sum_{k_j=1}^{K_j}\mathbf{A}_{jk_j}^T\boldsymbol{\Sigma}_{jk_j}^{-1}(\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j}))\}\}
\end{aligned} \tag{6.7}$$

is an a quadratic form function w.r.t. $\mathbf{h}_{ij}$. Thus the posterior probability of $\mathbf{h}_{ij}$ follows a Gaussian distribution, which can be rewritten as follows:

$$p(\mathbf{h}_{ij} \mid \mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) = \mathcal{N}(\mathbf{h}_{ij} \mid \boldsymbol{\mu}_{ij}^H, \boldsymbol{\Sigma}_{ij}^H) \tag{6.8}$$

where

$$\begin{aligned}
\boldsymbol{\Sigma}_{ij}^H &= (\sum_{p=1}^{P} z_{pi}\boldsymbol{\Sigma}_{jp}^{-1} + \sum_{k_j=1}^{K_j}\mathbf{A}_{jk_j}^T\boldsymbol{\Sigma}_{jk_j}^{-1}\mathbf{A}_{jk_j})^{-1} \\
\boldsymbol{\mu}_{ij}^H &= \boldsymbol{\Sigma}_{ij}^H\{\sum_{p=1}^{P} z_{pi}\boldsymbol{\Sigma}_{jp}^{-1}\boldsymbol{\mu}_{jp} + \sum_{k_j=1}^{K_j}\mathbf{A}_{jk_j}^T\boldsymbol{\Sigma}_{jk_j}^{-1}(\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j})\}
\end{aligned} \tag{6.9}$$

M Step: In M-step, all parameters are re-estimated by optimizing a concave low-bound function for (6.6), being

$$L(\boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H) = E_H[\log p(\mathbf{X}, \mathbf{Z}, \mathbf{H} \mid \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H)] \tag{6.10}$$

with a unique maximum point. In Eq.(6.10), the format is similar to log joint density $\log p(\mathbf{X}, \mathbf{Z}, \mathbf{H} \mid \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X)$, except to replace $\mathbf{h}_{ij}$ and $\mathbf{h}_{ij}\mathbf{h}_{ij}^T$ with $E[\mathbf{h}_{ij}]$ and $E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T]$, respectively. Since the mean and covariance of the posterior probability for $\mathbf{h}_{ij}$ are $\boldsymbol{\mu}_{ij}^H$ and $\boldsymbol{\Sigma}_{ij}^H$, which have been calculated in E-step, $E[\mathbf{h}_{ij}]$ and $E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T]$ can be obtained through the following equation:

$$\begin{aligned} E[\mathbf{h}_{ij}] &= \boldsymbol{\mu}_{ij}^H \\ E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T] &= \boldsymbol{\Sigma}_{ij}^H + \boldsymbol{\mu}_{ij}^H(\boldsymbol{\mu}_{ij}^H)^T \end{aligned} \tag{6.11}$$

By calculating the derivative of the low-bound function $L(\boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X)$ w.r.t. $\boldsymbol{\theta}_Z$, $\boldsymbol{\theta}_H$, and $\boldsymbol{\theta}_X$, and setting it to be zero, the parameters can be estimated with closed-form solutions. The results are listed as follows.

For the parameters corresponding to $\boldsymbol{\theta}_X$, the solutions are

$$\begin{aligned} \mathbf{A}_{jk_j} &= \{\sum_{i=1}^{M}(\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j})E[\mathbf{h}_{ij}^T]\}\{\sum_{i=1}^{M} E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T]\}^{-1} \\ \mathbf{b}_{jk_j} &= \frac{1}{M}\sum_{i=1}^{M}\{\mathbf{x}_{ijk_j} - \mathbf{A}_{jk_j}E[\mathbf{h}_{ij}]\} \\ \boldsymbol{\Sigma}_{jk_j} &= \frac{1}{M}\sum_{i=1}^{M}\{\mathbf{A}_{jk_j}E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T]\mathbf{A}_{jk_j}^T - 2\mathbf{A}_{jk_j}E[\mathbf{h}_{ij}] \\ &\quad (\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j})^T + (\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j})(\mathbf{x}_{ijk_j} - \mathbf{b}_{jk_j})^T\} \end{aligned} \tag{6.12}$$

For the parameters corresponding to $\boldsymbol{\theta}_H$, the solutions are

$$\begin{aligned} \boldsymbol{\mu}_{jp} &= \frac{1}{\sum_{i=1}^{M} z_{pi}}\sum_{i=1}^{M} z_{pi}E[\mathbf{h}_{ij}] \\ \boldsymbol{\Sigma}_{jp} &= \frac{1}{\sum_{i=1}^{M} z_{pi}}\sum_{i=1}^{M} z_{pi}\{E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T] - 2E[\mathbf{h}_{ij}]\boldsymbol{\mu}_{jp}^T + \boldsymbol{\mu}_{jp}\boldsymbol{\mu}_{jp}^T\} \end{aligned} \tag{6.13}$$

To estimate the parameter $\boldsymbol{\theta}_{\boldsymbol{Z}} = \{\pi_p\}_{p=1}^P$, the Lagrange Multiplier term is introduced to meet $\sum_{p=1}^P \pi_p = 1$. By calculating the derivative of the Lagrange function w.r.t. π_p and setting it to 0, the solution of π_p is then obtained according to the following equation:

$$\pi_p = \frac{\sum_{i=1}^{M} z_{pi}}{\sum_{p=1}^{P}\sum_{i=1}^{M} z_{pi}} \tag{6.14}$$

From Eq. (6.12), Eq. (6.13) and Eq. (6.14), we can see that each step has a closed-form solution which would greatly facilitate the parameter estimation process. According to the

Algorithm 6.1 [HMMF]Hierarchical Multi-view Multi-feature Fusion

Input: Observed data: $\mathbf{X}$; label: $\mathbf{Z}$;

Initialization: $\boldsymbol{\theta}_Z$;$\boldsymbol{\theta}_H$; $\boldsymbol{\theta}_X$

(Calculate $\boldsymbol{\theta}_Z$)
for $p = 1, ..., P$ **do**
    Calculate π_p by Eq (6.14)
end for
while not converged **do**
    E-step:
    for $i = 1, ..., M$ **do**
        for $j = 1, ..., J$ **do**
            Evaluate $\boldsymbol{\Sigma}_{ij}^H$ and $\boldsymbol{\mu}_{ij}^H$ by Eq.(6.9), and Calculate $E[\mathbf{h}_{ij}]$ and $E[\mathbf{h}_{ij}\mathbf{h}_{ij}^T]$ by Eq.(6.11).
        end for
    end for
    M-step: (re-estimate $\boldsymbol{\theta}_H$ and $\boldsymbol{\theta}_X$)
    for $j = 1, ..., J$ **do**
        for $p = 1, ..., P$ **do**
            calculate $\boldsymbol{\mu}_{jp}$ and $\boldsymbol{\Sigma}_{jp}$ through Eq.(6.13)
        end for
        for $k_j = 1, ..., K_j$ **do**
            calculate $\boldsymbol{\Sigma}_{jk_j}$, $\mathbf{b}_{jk_j}$ and $\mathbf{A}_{jk_j}$through Eq.(6.12)
        end for
    end for
end while

Output: $\boldsymbol{\theta}_Z$;$\boldsymbol{\theta}_H$; $\boldsymbol{\theta}_X$

convergence theory of EM algorithm, each update of the parameters acquired from an E-step followed by an M-step can guarantee the increase of the log likelihood function. Hence, to obtain a local maximin point, we alternatively execute E-step and M-step until convergence. The Algorithm 6.1 illustrates the details of the optimization. In this thesis, our proposed method is named as Hierarchical Multi-view Multi-feature Fusion (HMMF).

Computational Complexity Analysis: Without loss the generality per iteration, we let $D_1 = \max(D_j)$, $D_2 = \max(D_{jk_j})$, and $K = \max(K_j)$. Thus the complexity of our algorithm is $O(MC + J(CD_1^3 + K(MD_2D_1 + D_2^2D_1 + D_2^3)))$ for the general covariance matrix and $O(MC + JD_1^3 + JKMD_1D_2)$ for the diagonal covariance matrix.

6.3 Prediction

According to the Bayesian principle, the posterior probability of a given test sample $\mathbf{x} = \{\mathbf{x}_{jk_j}\}_{j,k_j=1}^{J,K_j}$ belonging to the p-th class is calculated through

$$p(z_p = 1 \mid \mathbf{x}, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X) = \frac{p(\mathbf{x} \mid z_p = 1, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X)p(z_p = 1)}{\sum_{p=1}^{P} p(\mathbf{x} \mid z_p = 1, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X)p(z_p = 1)}. \tag{6.15}$$

Since the second term of the numerator $p(z_p = 1) = \pi_p$, the key problem is how to calculate the first term of the numerator. Actually, we get its value by the following process:

$$\begin{aligned}
&\log p(\mathbf{x} \mid z_p = 1, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X) \\
=&\log \int p(\mathbf{x}, \mathbf{h} \mid z_p = 1, \boldsymbol{\theta}_Z, \boldsymbol{\theta}_H, \boldsymbol{\theta}_X) d\mathbf{h} \\
=&\sum_{j=1}^{J} \{(-\frac{1}{2} \log \frac{|\boldsymbol{\Sigma}_{jp}|}{\left|\boldsymbol{\Sigma}_{jp}^H\right|} - \frac{1}{2} \boldsymbol{\mu}_{jp}^T \boldsymbol{\Sigma}_{jp}^{-1} \boldsymbol{\mu}_{jp}) + \\
&\frac{1}{2} (\boldsymbol{\mu}_{jp}^H)^T (\boldsymbol{\Sigma}_{jp}^H)^{-1} \boldsymbol{\mu}_{jp}^H + \sum_{k_j}^{K_j} (-\frac{D_{jk_j}}{2} \log(2\pi) - \\
&\frac{1}{2} \log \left|\boldsymbol{\Sigma}_{jk_j}\right| - \frac{1}{2} (\mathbf{x}_{jk_j} - \mathbf{b}_{jk_j})^T \boldsymbol{\Sigma}_{jk_j}^{-1} (\mathbf{x}_{jk_j} - \mathbf{b}_{jk_j}))\}
\end{aligned} \tag{6.16}$$

where

$$\begin{aligned}
\boldsymbol{\Sigma}_{jp}^H &= (\boldsymbol{\Sigma}_{jp}^{-1} + \sum_{k_j=1}^{K_j} \mathbf{A}_{jk_j}^T \boldsymbol{\Sigma}_{jk_j}^{-1} \mathbf{A}_{jk_j})^{-1} \\
\boldsymbol{\mu}_{jp}^H &= \boldsymbol{\Sigma}_{jp}^H \{\boldsymbol{\Sigma}_{jp}^{-1} \boldsymbol{\mu}_{jp} + \sum_{k_j=1}^{K_j} \mathbf{A}_{jk_j}^T \boldsymbol{\Sigma}_{jk_j}^{-1} (\mathbf{x}_{jk_j} - \mathbf{b}_{jk_j})\}
\end{aligned} \tag{6.17}$$

Every parameter or variable is well-defined in Eq.(6.16). Thus first term of the numerator in Eq(6.15) as well as the posterior probability of this new sample belonging to the p-th class are easily gained. If only requiring the predicted label, the logarithm of the numerator in Eq(6.15) for all classes can be calculated, and predicted category is the one with the max value.

6.4 Experiments

In this section, experiments are conducted on both synthetic and real-world datasets to show the effectiveness of HMMF. We first introduce the datasets as well as the experimental setting. The comparison across various methods is then analyzed. Note that, since HMMF is not a kernel based method, we do not make a comparison between it and other GPLVM based approaches.

6.4.1 Datasets and Experimental Setting

The synthetic data is generated according to the assumption of the proposed method. Particularly, given the values of D_j and D_{k_j}, the parameters $\mathbf{A_{jk_j}}$, $\mathbf{\Sigma_{jk_j}}$, $\mathbf{\Sigma_{jp}}$ and $\mu_{\mathbf{jp}}$ are randomly generated. The latent variable $\mathbf{h}_{ij}$ is then obtained by following $\mathcal{N}(\mathbf{h}_{ij} \mid \sum_{p=1}^{P} z_{pi}\boldsymbol{\mu}_{jp}$, $\sum_{p=1}^{P} z_{pi}\mathbf{\Sigma}_{jp})$. Consequently, the observations are acquired according to $\mathcal{N}(\mathbf{x}_{ijk_j} \mid \mathbf{A}_{jk_j}\mathbf{h}_{ij}, \mathbf{\Sigma}_{jk_j})$. Without loss generality, we set the dimensionality D_j to be the same for each view. So does D_{jk_j}. In this experiment, we set D_j and D_{jk_j} to be 10 and 20, respectively.

The biomedical dataset (Health VS DM) [44] is also used to measure the effectiveness of HMMF. 40, 50, 60, and 70 samples in each class are randomly selected with five independent times as the training set, and the remaining samples are utilized for testing. Different from the setting in SAGP and MKSGP, we do not concatenate features in the tongue, face and sublingual vessel as a single one for HMMF. In fact, we regard the raw data as the multi-view and multi-feature data. Concretely, the face image can be represented by the 24-dimensional color feature (4 block$\times$6 dimension) and another 5-dimensional texture fea-

Table 6.1. The classification accuracies on the synthetic dataset obtained by HMMF.

	(J, K_j)			
Model	(2,2)	(2,1)	(1,4)	(4,1)
Accuracy	**82.2%**	80.6%	63.6%	80.4%
Model	(3,3)	(3,1)	(1,9)	(9,1)
Accuracy	**86.8%**	85.0%	68.6%	84.8%
Model	(4,4)	(4,1)	(1,16)	(16,1)
Accuracy	**93.6%**	92.4%	82.0%	92.0%
Model	(5,5)	(5,1)	(1,25)	(25,1)
Accuracy	**94.4%**	93.0%	90.2%	91.0%

ture; the tongue image can be represented with 12-dimensional color feature, 9-dimensional texture feature and 13-dimensional geometry feature; and the sublingual image can be represented with 6-dimensional color feature and 6-geometrical feature.

The third one is the Wiki Text-Image dataset [55]. As mentioned above, the image view is described with the 128-D SIFT histogram image feature and the text view is presented with 10-D latent Dirichlet features. In order to make this dataset be multi-view and multi-feature style, we also apply the Alexnet [31] to extract a CNN feature from the provided images. Note that, the dimensionality of the output of the Alexnet is reduced from 4096 to 30 in this thesis to decrease the training time.

Since HMMF is a subspace learning based linear method, we make it compare with some similar algorithms containing DPL [21], MDL (UMDL and SMDL) [6], JDCR [42], AWFA [76], MTJSRC [82], RCR [81], and CSRL [22] on the real-world datasets. For DPL, we concatenate all features in all views as a single one.

Here only D_j should be tuned through the 5-fold cross-validation. For the Biomedical dataset, since the dimension of several features is around 5 and according to results of the first and third datasets, we set D_j to be 5 empirically.

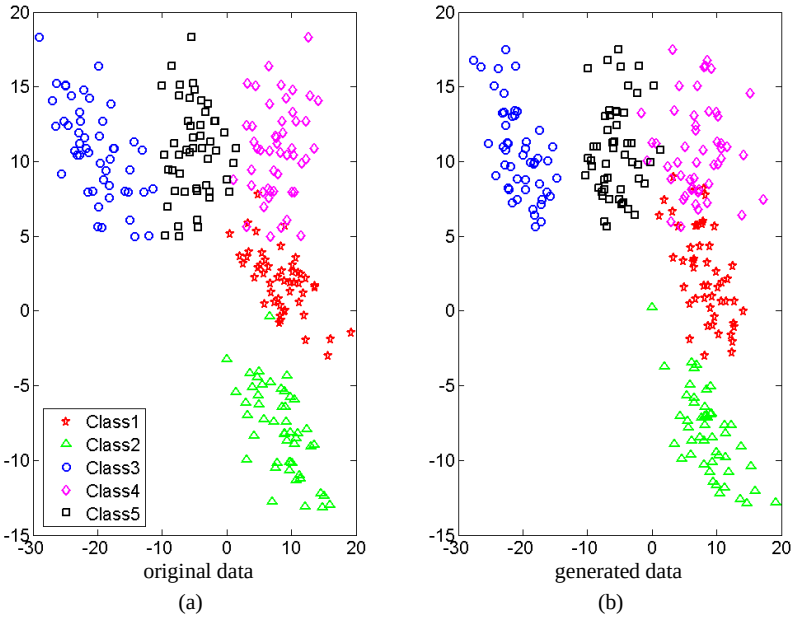


Figure 6.2. The comparison of the distributions between the original data and generated data.

6.4.2 Experimental Results on Three Datasets

Synthetic Dataset: In this experiment, we randomly generate four types of synthetic datasets, which are ($J = 2$, K_j=2), ($J = 3$, K_j=3), ($J = 4$, K_j=4) and ($J = 5$, K_j=5). As mentioned above, without loss generality, we set the number K_j of types of features in each view to be same. Note that D_j and D_{jk_j} are set to be 10 and 20, respectively. Additionally, we randomly generate 5 categories whose number of training and test samples is 20 and 100 in each class, respectively. In order to demonstrate the superiority of the hierarchical fusion, we reconstruct the inputs in another three types. For instance, as shown in Tab.6.1, when the number of views and their corresponding features are ($J = 3$, K_j=3), we concatenate the features in each view as a single one. Thus a novel input, whose J is 3 and K_j is 1 (3,1) is obtained. Additionally, we also follow the input as many existing multi-view methods do. We regard the sample with 9 types of features as the input and two cases including ($J = 1$, K_j=9 (1,9)) and ($J = 9$, K_j=1 (9,1)) are consequently acquired. From Tab.6.1 we can see that HMMF always obtains remarkable results compared with other methods. The performance obtained by HMMF is particularly better than that in other columns, indicating the significance of hierarchical fusion structure.

Table 6.2. The accuracy, sensitivity and specificity values obtained by different methods on the Biomedical dataset when the number of training samples is 40, 50, 60, and 70, respectively. Best results are highlighted in bold.

	Number of Training Samples											
Num	num=40			num=50			num=60			num=70		
Methods	Acc	Sen	Spe	Acc	Sen	Spe	Acc	Sen	Spe	Acc	Sen	Spe
DPL	77.2%	75.4%	79.0%	79.9%	78.2%	81.5%	79.4%	77.7%	81.0%	82.4%	80.3%	84.4%
AWFA	79.8%	75.6%	**83.9%**	81.2%	77.2%	**85.0%**	81.4%	75.5%	87.1%	81.6%	77.7%	85.3%
MTJSRC	79.0%	**82.2%**	76.0%	80.5%	80.6%	80.3%	81.0%	**86.0%**	76.1%	80.1%	82.1%	78.2%
RCR	78.6%	77.0%	80.3%	80.5%	80.3%	80.7%	80.9%	78.1%	83.6%	82.5%	80.6%,	84.4%
UMDL	77.4%	79.0%	75.8%	74.7%	73.4%	76.0%	79.1%	80.3%	78.0 %	79.6%	81.5%	77.8%
SDML	77.4%	79.0%	76.0%	73.6%	76.1%	71.2%	74.7%	70.4%	78.8 %	74.5%	75.9%	73.1%
JDCR	78.1%	75.8%	80.4%	80.1%	78.0%	82.0%	79.9%	78.8%	81.0 %	82.7%	80.7%	84.7%
CSRL	71.3%	73.5%	69.2%	72.2%	74.0%	70.5%	73.1%	77.9%	68.4%	75.4%	77.2%	73.7%
HMMF	**82.0%**	80.3%	83.7%	**82.7%**	**80.7%**	84.6%	**83.2%**	78.7%	**87.5%**	**84.9%**	**83.1%**	**86.6%**

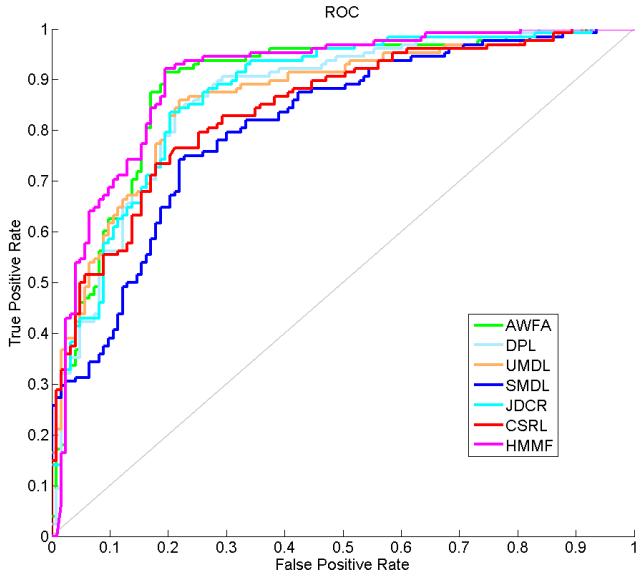


Figure 6.3. The ROC curves obtained by different methods in DM detection.

Additionally, we also visualize the data generated through the estimated parameters. In order to display distributions of different categories more clearly, we randomly re-generate the synthetic data by enlarging the mean of distributions belonging to different classes. The Fig.6.2(a) shows the locations of the first two dimensional points of the synthetic data in the first type of features in the first view when $J = 3$ and $K_j = 3$. Inputting this data into our model, we can subsequently obtain the parameters $\boldsymbol{\theta}_Z$, $\boldsymbol{\theta}_H$ and $\boldsymbol{\theta}_X$. Then these estimated parameters are exploited in our model to re-generate the five-class data, as shown in Fig.6.2(b). It is easy to see that the distributions of different categories generated according to the estimated parameters are quite similar to that in original data, which relatively substantiates the effectiveness and superiority of our hierarchical fusion model.

Biomedical Dataset: The experimental results computed by multiple approaches are listed in Tab.6.2. It is easy to observe that HMMF always achieves the better results compared with other strategies. Compared with UMDL, SMDL and CSRL, HMMF is obviously superior. In contrast to DPL, AWFA, RCR, MTJSRC and JDCR, the proposed method is also competitive, gaining about 2% improvement in classification accuracy.

The ROC curves and their associated AUC values are shown in Fig.6.3 and Tab.6.3,

Table 6.3. The area under curve (AUC) obtained by the different methods in DM detection.

Methods	AUC	Methods	AUC
DPL	0.8639	JDCR	0.8703
UMDL	0.8639	CSRL	0.8420
SMDL	0.8089	AWFA	0.8859
HMMF	**0.9027**		

Table 6.4. The accuracy obtained by the different methods in the Wiki Text-Image dataset.

Method	DPL	MTJSCR	RCR	UMDL	AWFA
Accuracy	65.1%	67.7%	66.1%	67.2%	69.0%
Method	SMDL	JDCR	CSRL	HMMF	
Accuracy	69.1%	68.5%	64.2%	**71.1%**	

respectively, when the training number is 70. From this figure we can see that the area covered by the ROC curve calculated by the proposed method is much larger than that computed by SMDL and CSRL. Compared DPL, AWFA, UMDL, and JDCR, HMMF also has the more or less enhancement. From the Tab.6.3, we can see that HMMF obtains the highest AUC values.

Wiki Text-Image Dataset: Tab.6.4 lists the overall accuracy conducted on the Wiki Text-Image Dataset. In this experiment, the dimension of the latent variable is set to be 8 for our proposed method. For CSRL, the dimension is 10 since it obtains the best result in this situation. From Tab.6.4 we can see that our presented hierarchical method gains the best performance compared with other methods. Compared with CSRL, DPL, AWFA, RCR, MTJSRC and UMDL, the presented approach achieves a noticeable enhancement. In contrast SMDL and JDCR, our algorithm also obtains about 2.0% improvement.

The Tab.6.5 further demonstrates the overall accuracy under the change of the dimension of the latent variable. When the dimension changes from 1 to 8, HMMF obtains an obvious increase. The reason is that a too low dimension would lose some valuable infor-

Table 6.5. The accuracy obtained by HMMF with the change of the dimensionality of the latent variable.

Dimension	Accuracy	Dimension	Accuracy
D_j=1	44.0%	D_j=6	68.7%
D_j=2	53.4%	D_j=7	70.6%
D_j=3	59.0%	D_j=8	71.1%
D_j=4	64.4%	D_j=9	70.4%
D_j=5	66.8%	D_j=10	69.4%

mation. Subsequently, with the continuous increase of the dimension, HMMF meets a slight performance degradation, relatively reflecting that a reasonable selection of the dimension plays a key role in the classification.

6.5 Conclusion

In this chapter, a generative and probabilistic method is proposed to process the multi-view and multi-feature data. Different from most existing multi-view learning methods which are only for traditional multi-view data, our proposed method is capable of fusing multiple features from a same view hierarchically. These fused features associated with various views are also influenced by their ground-truth. Thanks for this hierarchical structure, the correlation and semantic information among multiple view and multiple features are fully exploited. Experimental results on both synthetic datates and real-world datasets show the effectiveness of our proposed method.

CHAPTER 7

DISCUSSION, CONCLUSION AND FUTURE WORKS

7.1 Discussion

In this thesis, four multi-view learning methods including JSSL, SAGP, MKSGP and HMMF are proposed. In comparison to JSSL and HMMF, SAGP and MKSGP introduce the GP prior into the model to non-linearly represent the data. Differently, HMMF is designed which is quite adaptive for the multi-view and multi-feature data. To make a comparison among these four methods, we conduct an experiment on the Biomedical dataset since it is multi-view and multi-feature dataset, which can be applied to all methods. The accuracy, sensitivity and specificity values obtained by these four methods on the Biomedical dataset when the number of training samples is 30, 40, and 50 are shown in Tab.7.1.As we can see, MKSGP gains the best performance on the classification accuracy. Compared with JSSL and SAGP, MKSGP not only utilizes the GPLVM structure, but also introduces the multi-kernel strategy, which are both beneficial for real-world data representation. Although HMMF is designed for multi-view and multi-feature data, its assumption follows the gaussian distribution, which is linear and inferior to that in MKSGP.

We also list the comparison of JSSL, SAGP and MKSGP on the Wiki Text-Image, AWA and NUS-WIDE-LITE datasets in Tab.7.2, Tab.7.3 and Tab.7.4, respectively. Note that the experimental setting for the MKSGP on the Wiki Text-Image dataset is the same to that in SAGP. It is easy to see that MKSGP achieves the best performance in most cases, followed by SAGP and JSSL.

Table 7.1. The accuracy, sensitivity and specificity values obtained by different methods on Biomedical dataset when the number of training samples is 30, 40, and 50, respectively. Best results are highlighted in bold.

	Number of Training Samples								
Num	num=30			num=40			num=50		
Methods	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
JSSL	79.8%	78.7%	81.0%	81.5%	83.1%	79.9%	82.8%	82.0%	83.6%
SAGP	79.0%	78.9%	79.2%	79.2%	81.3%	77.2%	82.5%	83.9%	81.2%
MKSGP	**80.7%**	81.1%	80.2%	**82.1%**	82.5%	81.8%	**83.6%**	84.8%	82.4%
HMMF	80.7%	80.0%	81.3%	82.0%	80.3%	83.7%	82.7%	80.7%	84.6%

Table 7.2. The overall and average accuracies on the Wiki Test-Image dataset obtained by JSSL, SAGP and MKSGP when the dimensionality of the latent variable changes from 1 to 10.

		Dimensionality									
Methods	Result	d_{out}=1	d_{out}=2	d_{out}=3	d_{out}=4	d_{out}=5	d_{out}=6	d_{out}=7	d_{out}=8	d_{out}=9	d_{out}=10
JSSL	Overall	68.0%									
	Average	64.9%									
SAGP(SE)	Overall	50.5%	58.9%	65.6%	67.7%	70.6%	70.0%	69.7%	71.0%	69.3%	65.1%
	Average	48.0%	54.9%	62.3%	63.8%	**67.2%**	66.5%	65.5%	67.1%	64.6%	61.7%
MKSGP	Overall	**52.5%**	**60.0%**	**67.0%**	**68.0%**	**71.0%**	**70.3%**	**70.7%**	**71.4%**	**71.1%**	**70.9%**
	Average	**48.9%**	**56.9%**	**62.5%**	**63.8%**	67.1%	**67.9%**	**67.6%**	**68.3%**	**67.5%**	**67.2%**

Table 7.3. The overall and average accuracies on the AWA dataset computed by JSSL, SAGP and MKSGP when the dimensionality of the latent variable changes from 40 to 130.

		Dimensionality									
Methods	Result	d_{out}=40	d_{out}=50	d_{out}=60	d_{out}=70	d_{out}=80	d_{out}=90	d_{out}=100	d_{out}=110	d_{out}=120	d_{out}=130
JSSL	Overall	83.8%									
	Average	81.7%									
SAGP(SE)	Overall	**82.4%**	84.3%	83.4%	82.7%	83.3%	83.3%	83.1%	83.3%	82.6%	82.9%
	Average	**80.0%**	82.3%	81.5%	80.8%	81.4%	81.4%	81.0%	81.5%	80.9%	81.4%
MKSGP	Overall	81.6%	**85.3%**	**85.4%**	**85.3%**	**85.4%**	**85.2%**	**85.2%**	**84.9%**	**85.0%**	**85.1%**
	Average	78.7%	**83.2%**	**83.4%**	**83.3%**	**83.3%**	**83.0%**	**83.0%**	**82.6%**	**82.8%**	**83.0%**

Table 7.4. The overall and average accuracies on the NUS-WIDE-LITE dataset computed by JSSL, SAGP and MKSGP when q changes from 1 to 30.

		Dimensionality						
Methods	Result	d_{out}=1	d_{out}=5	d_{out}=10	d_{out}=15	d_{out}=20	d_{out}=25	d_{out}=30
JSSL	Overall	47.2%						
	Average	48.9%						
SAGP(SE)	Overall	23.6%	52.6%	58.7%	56.2%	56.9%	53.1%	51.8%
	Average	25.2%	51.9%	56.6%	55.0%	55.7%	50.4%	49.8%
MKSGP	Overall	**26.2%**	**59.6%**	**62.2%**	**62.5%**	**63.0%**	**62.6%**	**58.6%**
	Average	**27.1%**	**54.9%**	**58.6%**	**59.2%**	**59.9%**	**59.6%**	**56.4%**

7.2 Conclusion

In this thesis, we focus on the multi-view learning and propose four novel methods for classification.

In chapter 3, considering the similar and specific parts existing across different views, the JSSL is proposed to effectively and sparsely divide the representation coefficients into the similar and diverse ones. In this case, a balance between similarity and distinctiveness among all features is achieved which leads to a more stable and accurate representation for classification tasks.

In chapter 4, to tackle the non-linearity existing in the real-world data, the second method SAGP is presented by introducing the auto-encoder and GPLVM to learn a shared variable on the manifold. Thanks for these two structures, we can not only represent the data in a non-linear and smooth way, but also get the latent variable corresponding to the testing sample simply.

In chapter 5, to address the limitations of SAGP in the covariance construction and classifier learning, we further extend SAGP to MKSGP by jointly taking the multi-kernel learning and the large margin prior into account. Compared with SAGP, MKSGP is more

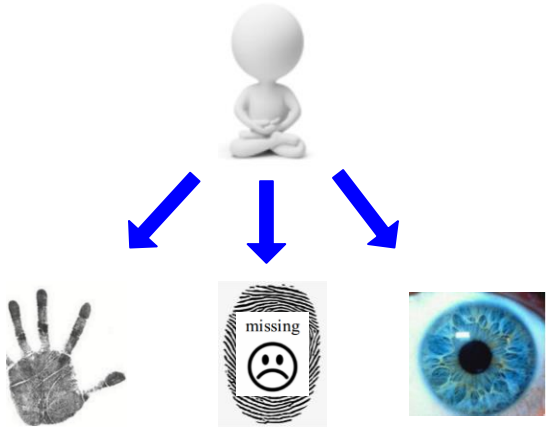


Figure 7.1. The examples of incomplete multi-view data.

powerful in data representation and more adaptive for classifier learning. Experimental results show that MKSGP achieves better performance.

In chapter 6, considering the multi-view and multi-feature data, we also propose a probabilistic model to hierarchically fuse multiple views and multiple features, fully exploiting the correlation among them. Particularly, a shared and latent variable is first fused for the observed features from a view or modality. These learned variables associated with different views are then assumed to be independently influenced by their ground-truth label. To optimize HMMF, EM algorithm is exploited to obtain the closed-form solution for each variable or parameter.

7.3 Future Works

Although four fusion approaches have been presented and achieved the remarkable performance in classification, there are still some problems on multi-view or multi-feature data that should be overcome.

The main importance is how to model the correlation when the multi-view data is incomplete. Incomplete multi-view data means one or more than one views may suffer from

missing data, as shown is Fig.7.1. In the real world applications, there are many situations in which complete datasets are not available due to the sensor failure or other reasons. Thus, many existing multi-view methods would inevitably degenerate or even fail. To solve the incomplete multi-view problem, a naive strategy is to remove samples suffering from missing information or generate the incomplete samples before processing [87]. However, these methods would result in the reduction of the number of the training samples or inaccurate estimation, which greatly degrade the performance of classification. Therefore, an efficient approach of well modeling the incomplete multi-view data is crucial. In our future work, we will focus on this problem to not only reveal the common and latent correlation across views, but also fully exploit the information among different views. In our plan, we attempt to divide the incomplete multi-view data into two parts: the overlapped and specific ones. For the overlapped part, the conventional multi-view based methods can be applied to reveal the relationship across different views. Generally speaking, the key issue is to exploit the view-specific parts to improve the performance. However, since these specific samples from different views are not paired, it is unable to find a common component in a latent space. In our future work, we plan to use the distribution to measure the correlation of these specific parts, which means that these parts from different views would be projected in a subspace by following the same distribution.

Additionally, from Tab.7.1 we can see that, HMMF is still inferior to MKSGP although it is specifically designed for the multi-view and multi-feature data. The main reason is that the assumption of HMMF only follows gaussian distributions, while MKSGP introduces the GP prior and multi-kernel strategy, which is more powerful of non-linearly representing the data. To overcome the limitation of the HMMF, we plan to extend it to a more powerful version. One reasonable strategy is to modify the gaussian distribution to the mixture of gaussian model, which would be more reasonable to represent the real-world data. However, this assumption would increase the difficulty of the optimization. To address this problem, it would be better to use the variational bayes to estimate the parameter in the extended HMMF. We will make a detail analysis for the extension version of HMMF.

Furthermore, since our goal is to apply the proposed methods to classification, it

is general to encounter the case with imbalanced data in different classes. Especially, the imbalanced data classification is less to be discussed in the multi-view learning. Therefore, it is significant to have a study on this field. To address this problem, a simple way is to downsample the data from a class with abundant samples. However, this strategy is too naive since the samples which are not selected would be ignored. Thus, we prefer to introducing a strategy named self-space learning [27] to our multi-view learning methods. In this way, we can encourage the classes with fewer samples to have larger weights, which can enhance the importance of these classes, being reasonable to solve the imbalanced problem.

Bibliography

[1] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

[2] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.

[4] Cédric Archambeau and Francis R Bach. Sparse probabilistic projections. In *Advances in neural information processing systems*, pages 73–80, 2009.

[5] Francis R Bach and Michael I Jordan. A probabilistic interpretation of canonical correlation analysis. 2005.

[6] Soheil Bahrampour, Nasser M Nasrabadi, Asok Ray, and William Kenneth Jenkins. Multimodal task-driven dictionary learning for image classification. *IEEE Transactions on Image Processing*, 25(1):24–38, 2016.

[7] Samy Bengio, Fernando Pereira, Yoram Singer, and Dennis Strelow. Group sparse coding. In *Advances in neural information processing systems*, pages 82–89, 2009.

[8] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[9] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[12] Alfredo Cuzzocrea, Francesco Folino, Massimo Guarascio, and Luigi Pontieri. A multi-view multi-dimensional ensemble learning approach to mining business process deviances. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3809–3816. IEEE, 2016.

[13] Tom Diethe, David Roi Hardoon, and John Shawe-Taylor. Constructing nonlinear discriminants from multiple data views. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 328–343. Springer, 2010.

[14] Carl Henrik Ek and PHTND Lawrence. *Shared Gaussian process latent variable models*. PhD thesis, PhD thesis, 2009.

[15] Stefanos Eleftheriadis, Ognjen Rudovic, and Maja Pantic. Discriminative shared gaussian processes for multiview and view-invariant facial expression recognition. *IEEE transactions on image processing*, 24(1):189–204, 2015.

[16] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[17] Lucas Franek and Xiaoyi Jiang. Ensemble clustering by means of clustering embedding in vector spaces. *Pattern Recognition*, 47(2):833–842, 2014.

[18] Kenji Fukumizu, Francis R Bach, and Arthur Gretton. Statistical consistency of kernel canonical correlation analysis. *Journal of Machine Learning Research*, 8(Feb):361–383, 2007.

[19] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Kernel sparse representation for image classification and face recognition. In *Computer Vision–ECCV 2010*, pages 1–14. Springer, 2010.

[20] Xinbo Gao, Xiumei Wang, Dacheng Tao, and Xuelong Li. Supervised gaussian process latent variable model for dimensionality reduction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(2):425–434, 2011.

[21] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Projective dictionary pair learning for pattern classification. In *Advances in neural information processing systems*, pages 793–801, 2014.

[22] Yuhong Guo. Convex subspace representation learning from multi-view data. In *AAAI*, volume 1, page 2, 2013.

[23] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[26] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[27] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

[28] Xinwei Jiang, Junbin Gao, Xia Hong, and Zhihua Cai. Gaussian processes autoencoder for dimensionality reduction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 62–73. Springer, 2014.

[29] Meina Kan, Shiguang Shan, Haihong Zhang, Shihong Lao, and Xilin Chen. Multi-view discriminant analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):188–194, 2016.

[30] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[32] Vipin Kumar and Sonajharia Minz. Multi-view ensemble learning: A supervised feature set partitioning for high dimensional data classification. In *Proceedings of the Third International Symposium on Women in Computing and Informatics*, pages 31–37. ACM, 2015.

[33] Pei Ling Lai and Colin Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000.

[34] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[35] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[36] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6(Nov):1783–1816, 2005.

[37] Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16(3):329–336, 2004.

[38] Neil D Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the gp-lvm through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, pages 513–520. ACM, 2006.

[39] Jinxing Li, Weiyong Hong, Bob Zhang, Mu Li, Lei Zhang, and David Zhang. A probabilistic hierarchical model for multi-view and multi-feature classification. *AAAI Conference on Artificial Intelligence*, 2018.

[40] Jinxing Li, Bob Zhang, Guangming Lu, and David Zhang. Generative multi-view and multi-feature learning for classification. *Information Fusion*, 2018.

[41] Jinxing Li, Bob Zhang, Hu Ren, and David Zhang. Visual classification with multi-kernel shared gaussian process latent variable model. *IEEE Transactions on Cybernetics*, pages 1–14, 2018.

[42] Jinxing Li, Bob Zhang, and David Zhang. Joint discriminative and collaborative representation for fatty liver disease diagnosis. *Expert Systems with Applications*, 89:31–40, 2017.

[43] Jinxing Li, Bob Zhang, and David Zhang. Shared autoencoder gaussian process latent variable model for visual classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[44] Jinxing Li, David Zhang, Yongcheng Li, Jian Wu, and Bob Zhang. Joint similar and specific learning for diabetes mellitus and impaired glucose regulation detection. *Information Sciences*, 2016.

[45] Sheng Li and Yun Fu. Learning robust and discriminative subspace with low-rank constraints. 2015.

[46] David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. In *International Conference on Machine Learning*, pages 1359–1367, 2014.

[47] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[48] Xiaoqiang Lu, Xuelong Li, and Lichao Mou. Semi-supervised multitask learning for scene recognition. *IEEE transactions on cybernetics*, 45(9):1967–1976, 2015.

[49] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2272–2279. IEEE, 2009.

[50] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.

[51] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[52] Mihalis A Nicolaou, Yannis Panagakis, Stefanos Zafeiriou, and Maja Pantic. Robust canonical correlation analysis: Audio-visual fusion for learning continuous interest. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1522–1526. IEEE, 2014.

[53] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint ?2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010.

[54] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[55] Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert RG Lanckriet, Roger Levy, and Nuno Vasconcelos. A new approach to cross-modal multimedia retrieval. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 251–260. ACM, 2010.

[56] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[57] Fernando Rodriguez and Guillermo Sapiro. Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries. Technical report, DTIC Document, 2008.

[58] Lorenzo Rosasco, Alessandro Verri, Matteo Santoro, Sofia Mosci, and Silvia Villa. Iterative projection methods for structured sparsity regularization. 2009.

[59] Jan Rupnik and John Shawe-Taylor. Multi-view canonical correlation analysis. In *Conference on Data Mining and Data Warehouses (SiKDD 2010)*, pages 1–4, 2010.

[60] Mathieu Salzmann and Raquel Urtasun. Implicitly constrained gaussian process regression for monocular non-rigid pose estimation. In *Advances in Neural Information Processing Systems*, pages 2065–2073, 2010.

[61] Abhishek Sharma, Abhishek Kumar, Hal Daume, and David W Jacobs. Generalized multiview analysis: A discriminative latent space. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2160–2167. IEEE, 2012.

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[63] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[64] Guoli Song, Shuhui Wang, Qingming Huang, and Qi Tian. Similarity gaussian process latent variable model for multi-modal data analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4050–4058, 2015.

[65] Daniel Spielman. Spectral graph theory. *Lecture Notes, Yale University*, pages 740–0776, 2009.

[66] Jianyong Sun and Simeon Keates. Canonical correlation analysis on data with censoring and error information. *IEEE transactions on neural networks and learning systems*, 24(12):1909–1919, 2013.

[67] Geer Teng, Changzheng He, Jin Xiao, Yue He, Bing Zhu, and Xiaoyi Jiang. Cluster ensemble framework based on the group method of data handling. *Applied Soft Computing*, 43:35–46, 2016.

[68] Bruce Thompson. Canonical correlation analysis. *Encyclopedia of statistics in behavioral science*, 2005.

[69] Raquel Urtasun and Trevor Darrell. Discriminative gaussian process latent variable model for classification. In *Proceedings of the 24th international conference on Machine learning*, pages 927–934. ACM, 2007.

[70] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.

[71] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39. IEEE, 2009.

[72] David Weenink. Canonical correlation analysis. In *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, volume 25, pages 81–99. Citeseer, 2003.

[73] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[74] Fei Wu, Xiao-Yuan Jing, Xinge You, Dong Yue, Ruimin Hu, and Jing-Yu Yang. Multi-view low-rank dictionary learning for image classification. *Pattern Recognition*, 50:143–154, 2016.

[75] Jin Xiao, Xiaoyi Jiang, Changzheng He, and Geer Teng. Churn prediction in customer relationship management via gmdh-based multiple classifiers ensemble. *IEEE Intelligent Systems*, 31(2):37–44, 2016.

[76] Yong Xu and Yuwu Lu. Adaptive weighted fusion: A novel fusion approach for image classification. *Neurocomputing*, 168:566–574, 2015.

[77] Meng Yang, Dengxin Dai, Linlin Shen, and Luc Van Gool. Latent dictionary learning for sparse representation based classification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4138–4145. IEEE, 2014.

[78] Meng Yang, Luc Van Gool, and Lei Zhang. Sparse variation dictionary learning for face recognition with a single training sample per person. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 689–696. IEEE, 2013.

[79] Meng Yang, Lei Zhang, Xiangchu Feng, and David Zhang. Sparse representation based fisher discrimination dictionary learning for image classification. *International Journal of Computer Vision*, 109(3):209–232, 2014.

[80] Meng Yang, Lei Zhang, Jian Yang, and Dejing Zhang. Regularized robust coding for face recognition. *Image Processing, IEEE Transactions on*, 22(5):1753–1766, 2013.

[81] Meng Yang, Lei Zhang, David Zhang, and Shenlong Wang. Relaxed collaborative representation for pattern classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2224–2231. IEEE, 2012.

[82] Xiao-Tong Yuan, Xiaobai Liu, and Shuicheng Yan. Visual classification with multitask joint sparse representation. *Image Processing, IEEE Transactions on*, 21(10):4349–4360, 2012.

[83] Yuan Yuan, Jianzhe Lin, and Qi Wang. Hyperspectral image classification via multi-task joint sparse representation and stepwise mrf optimization. *IEEE transactions on cybernetics*, 46(12):2966–2977, 2016.

[84] Yun-Hao Yuan and Quan-Sen Sun. Multiset canonical correlations using globality p-reserving projections with applications to feature extraction and recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1131–1146, 2014.

[85] Lefei Zhang, Qian Zhang, Liangpei Zhang, Dacheng Tao, Xin Huang, and Bo Du. Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. *Pattern Recognition*, 48(10):3102–3112, 2015.

[86] Lei Zhang and David Zhang. Visual understanding via multi-feature shared learning with global consistency. *IEEE Transactions on Multimedia*, 18(2):247–259, 2016.

[87] S-YLYJ Zhi and Hua Zhou. Partial multi-view clustering. In *AAAI Conference on artificial intelligence*. Citeseer, 2014.

[88] Wangmeng Zuo, Deyu Meng, Lei Zhang, Xiangchu Feng, and Dejing Zhang. A generalized iterated shrinkage algorithm for non-convex sparse coding. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 217–224. IEEE, 2013.