Shiliang Sun · Liang Mao · Ziang Dong · Lidan Wu

Multiview Machine Learning

Springer

Multiview Machine Learning

Shiliang Sun · Liang Mao ·
Ziang Dong · Lidan Wu

Multiview Machine Learning

Springer

Shiliang Sun
Department of Computer Science and Technology
East China Normal University
Shanghai, China

Liang Mao
Department of Computer Science and Technology
East China Normal University
Shanghai, China

Ziang Dong
Department of Computer Science and Technology
East China Normal University
Shanghai, China

Lidan Wu
Department of Computer Science and Technology
East China Normal University
Shanghai, China

ISBN 978-981-13-3028-5 ISBN 978-981-13-3029-2 (eBook)
https://doi.org/10.1007/978-981-13-3029-2

Library of Congress Control Number: 2018963292

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

During the past two decades, multiview learning as an emerging direction in machine learning became a prevailing research topic in artificial intelligence (AI). Its success and popularity were largely motivated by the fact that real-world applications generate various data as different views while people try to manipulate and integrate those data for performance improvements. In the data era, this situation will continue. We think the multiview learning research will be active for a long time, and further development and in-depth studies are needed to make it more effective and practical.

In 2013, a review paper of mine, entitled "A Survey of Multi-view Machine Learning" (*Neural Computing and Applications*, 2013), was published. It generates a good dissemination and promotion of multiview learning and has been well cited. Since then, much more research has been developed. This book aims to provide an in-depth and comprehensive introduction to multiview learning and hope to be helpful for AI researchers and practitioners.

I have been working in the machine learning area for more than 15 years. Most of my work introduced in this book was completed after I graduated from Tsinghua University and joined East China Normal University in 2007. And we also include many important and representative works from other researchers to make the book content complete and comprehensive. Due to space and time limits, we may not be able to include all relevant works.

I owe many thanks to the past and current members of my Pattern Recognition and Machine Learning Research Group, East China Normal University, for their hard work to make research done in time. The relationship between me and them is not just professors and students, but also comrades-in-arms.

Shanghai, China
September 2018

Shiliang Sun

Contents

Chapter 1
Introduction

1.1 Background

In many scientific data analysis tasks, data are often collected through different measuring methods, such as various feature extractors or sensors, as usually the single particular measuring method cannot comprehensively describe all the information of the data. In this case, the features of each data example can be naturally divided into groups, each of which can be considered as a view. For instance, for images and videos, color features and texture features can be regarded as two views. It is important to make good use of the information from different views. Multiview machine learning (referred to as multiview learning for short) is a branch of machine learning which studies and utilizes the information and relationship between views. Two representative works of multiview learning in the early days are canonical correlation analysis (CCA) (Hotelling 1936) and co-training (Blum and Mitchell 1998). Especially after co-training was proposed, the study of multiview learning is on the rise. A multiview learning workshop was held at the International Conference on Machine Learning in 2005, which further boosted the research of multiview learning. So far, multiview learning has made great progress in theory and practice. The related ideas have also been incorporated into several areas of machine learning and have been developed continually. A well-designed multiview learning strategy may bring benefits to the developments of these areas.

1.2 Definition of Multiview Machine Learning and Related Concepts

Multiview learning is an important branch of machine learning (Sun 2013). During learning, multiview learning explicitly uses multiple distinct representations of data, and models the relationship between them and/or the results from downstream computations. Here, these "representation" can either be the original features of the data or the features obtained through some computations. The abrupt approach to utilize these representations is to simply concatenate them into a single representation to perform learning. However, two drawbacks of this strategy are that over-fitting may

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_1

arise on relatively small training sets and the specific statistical property of each view is ignored. Furthermore, multiview learning can effectively improve the performance of learning on natural single-view data.

In general, multiview learning methods are divided into three major categories: co-training style algorithms, co-regularization style algorithms, and margin-consistency style algorithms (Zhao et al. 2017).

Co-training style algorithms ride on single-view learning algorithms. They are often used to solve semi-supervised problems (Blum and Mitchell 1998). Co-training style algorithms makes use of multiple views of data to iteratively learn multiple classifiers that can provide predicted labels for the unlabeled data for one another. For example, co-EM (Nigam and Ghani 2000), co-testing (Muslea et al. 2006), and robust co-training (Sun and Jin 2011) belong to this co-training style algorithm.

For co-regularization style algorithms, the disagreement between the discriminant or regression functions of two views is regarded as a regularization term in the objective function (Sun et al. 2017). The regularization terms constrain that the prediction results from multiple views should be close.

Besides the two conventional style algorithms, margin-consistency style algorithms are recently proposed to make use of the latent consistency of classification results from multiple views (Sun and Chao 2013; Chao and Sun 2016). They are realized under the framework of maximize entropy discrimination (MED). Different from the co-regularization style algorithms which make restrictions on the discriminant or regression functions from multiple views, margin-consistency style algorithms model the margin variables of multiple views to be close and constrain that the product of every output variable and discriminant function should be greater than every margin variable. Particularly, in the margin-consistency style algorithms, the values of the discriminant functions for multiple views may have a large difference.

Multiview learning is closely associated with data fusion and multimodal learning, but their motivations are somewhat different. Data fusion emphasizes the comprehensive utilization of multiple raw representations of data. Multimodal learning emphasizes that multiple raw representations of data come from different modalities (such as data obtained by different types of sensors) and are then combined for machine learning. Multiview learning not only emphasizes that data have many different representations (for example, images can be captured from different angles by the same type of sensor), but also emphasizes that building models and algorithms should consider the relationship between views.

1.3 Typical Application Fields in Artificial Intelligence

The potential of multiview learning is huge. Research in basic theory and algorithms can promote the development of many important applications of artificial intelligence. Multiview data have become increasingly available in real-world applications. For example, in multimedia understanding, multimedia segments can be described simultaneously by their video and audio signals. In web page classification, there are

often two views for describing a given web page: the text content of the web page itself and the anchor text of any web page linking to this page. In content-based web image retrieval, an object is described simultaneously through the visual features of the image and the text surrounding the image. Different views usually contain complementary information, and multiview learning can use this information to model data better than single-view learning methods. Therefore, multiview learning has become a promising research area with wide applicability. We describe four typical application areas as follows.

Cross-Media Intelligence. Information retrieval is a process of finding desired or interesting contents from a collection of information. Various multimedia information (such as images, texts, and videos, etc.) coexisting on the internet requires cross-media retrieval techniques. The key of cross-media retrieval techniques is modeling relationships between different media representations and reducing the semantic gaps to achieve the purpose of using a modal example to retrieve other modal examples with similar semantics. Generating text from images and videos, or the inverse is important to research, where the generating between text and videos is a multiview learning problem with sequential representations. Another research direction in cross-media intelligence technology is video classification. Since the representation of a video usually involves a variety of features (such as image frame, image stream, audio, and text, etc.), video classification is a typical multiview learning problem.

Intelligent Medical. The development of artificial intelligence has provided a very advantageous technical support for the intelligent development in the medical and health field. Intelligent medical plays an important role in computer-aided diagnosis, medical image analysis, and disease prediction. Sampling patient data often involves multiple sets of different docimasias and assays, and doctors make a diagnosis based on these results. When diagnosing images of pathological slices, pathologists often combine text (natural language) descriptions of the patient's condition from clinicians. In addition, there are also a large number of sequential data in the medical and health field, such as the patient's electronic health records, which constitutes multiview sequential data. Modeling the dynamic of people's health records to reveal the principles underlying the changes in patients' conditions over time can provide theoretical and algorithmic foundations for the development of precision medicine.

Automatic Driving. Automatic driving relies on the collaboration of artificial intelligence, computer vision, radar, GPS, and so on, so that the computer can automatically control the vehicle safely without human initiative in Fig. 1.1. When the automatic driving system perceives the environment, it often depends on the multiple types of equipment such as radar, camera, GPS, and so on. These equipment provide different representations of the environment that a vehicle faces.

Intelligent Education. Education takes an important strategic position in the development of a country. The application of artificial intelligence technology to analyze educational data can promote personalized education, campus safety, and a lifelong learning, etc. Another application of artificial intelligence technology is information recommendation, where news may be recommended to specific groups through a

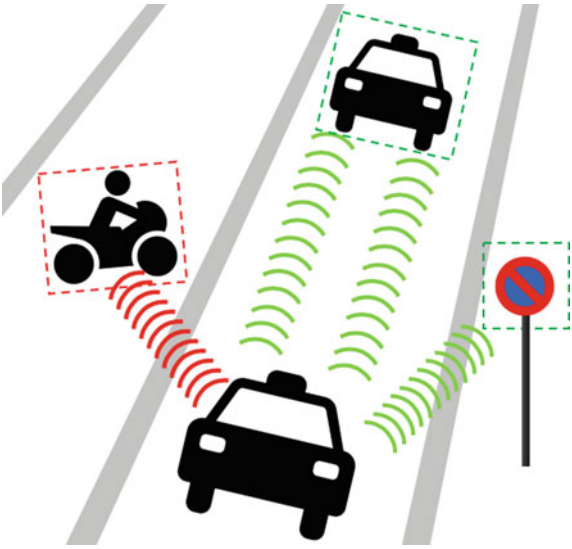


Fig. 1.1 Multiview learning has provided support for automatic driving

trained classifier according to the different learning stages of the involved students. In addition to the text (natural language), the news is often accompanied by rich images, which form two different representations of the news content.

1.4 Why Can Multiview Learning Be Useful

The complementarity from different representative views can effectively improve the performance of artificial intelligence systems. In the natural language processing problem, it is very beneficial for many sequence annotation tasks if you have access to the following context in addition to the proceeding one. In the case of special character classification, it would be very helpful that you know the next letter as you know the previous letter. For example, bi-directional long short-term memory (BiLSTM) not only considers the information contained in the sequence, but also consider the information of the reverse sequence as the second representation to complement the comprehensive description of the sequence. Similarly, animals with two eyes (two views) can capture depth information in three dimensional that cannot be captured by a single view as depicted in Fig. 1.2.

How to better utilize the relationships between views is another important aspect. In order to adapt the traditional single-view learning algorithms, one approach can be simply concatenating all multiple views into one single view. But there are two shortcomings: it may break the balance between the size and the dimensions of the data and cause over-fitting; it may ignore the obvious differences in the physical interpretations of different views. Multiview learning explicitly models the relationship between these different views and/or the results from downstream computation.

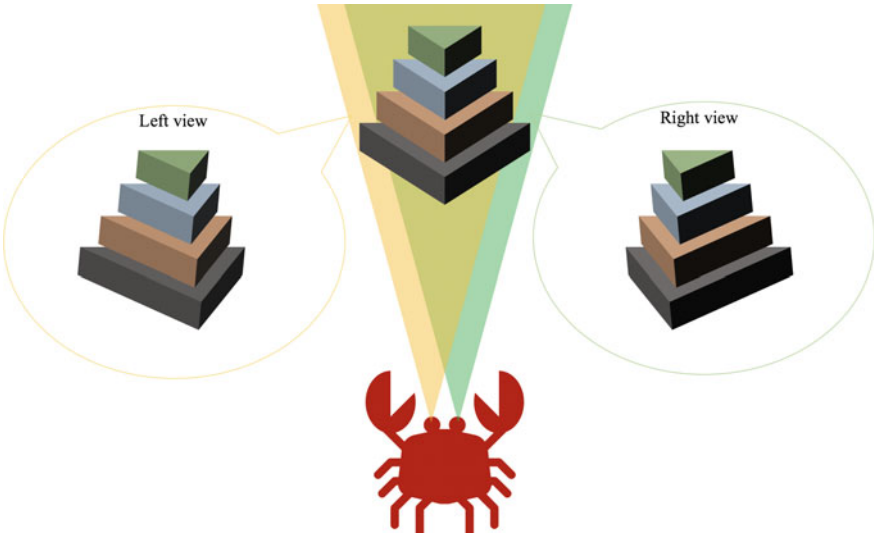


Fig. 1.2 Each eye captures its own view and the two views are sent on to the brain for processing

It better utilizes the structural information in the data, which leads to better modeling capability. Further, multiple views can act as complementary and collaborative normalization, effectively reducing the size of the hypothesis space.

1.5 Book Structure

The goal of this monograph is to review important developments in the field of multiview learning, especially models and algorithms, with the hope to provide inspirations for further research. In particular, we hope to inspire people to think about what can be done to make multiview learning more successful.

In this monograph, we provide a comprehensive overview of multiview learning methods. The remainder of this monograph proceeds according to the applications of multiview learning. The arrangement of chapters basically follows three principles: from early works to recent works, from a single task to multiple tasks, and from shallow models to deep models. Chapter 2 surveys multiview semi-supervised learning methods which combine the limited labeled data together with the unlabeled data for effective function learning. In Chap. 3, we introduce multiview subspace learning-based approaches. We illustrate different kinds of multiview supervised learning algorithms and multiview clustering in Chaps. 4 and 5. Multiview active learning is introduced in Chap. 6, whose aim is to alleviate the burden of labeling abundant examples by asking the user to label only the most informative ones. In Chap. 7, we introduce multiview transfer learning and multitask learning. A variety of multiview

deep learning methods have been proposed to learn robust multiview representation and deal with multiview data, which will be introduced in Chap. 8. Finally, we list some different approaches to the construction of multiple views in Chap. 9.

References

Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th annual conference on computational learning theory, ACM, pp 92–100

Chao G, Sun S (2016) Alternative multiview maximum entropy discrimination. IEEE Trans Neural Netw Learn Syst 27(7):1445–1456

Hotelling H (1936) Relations between two sets of variates. Biometrika 28(3/4):321–377

Muslea I, Minton S, Knoblock CA (2006) Active learning with multiple views. J Artif Intell Res 27(1):203–233

Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. In: Proceedings of the 9th international conference on information and knowledge management, ACM, pp 86–93

Sun S (2013) A survey of multi-view machine learning. Neural Comput Appl 23(7–8):2031–2038

Sun S, Chao G (2013) Multi-view maximum entropy discrimination. In: Proceedings of the 23rd international joint conference on artificial intelligence, pp 1706–1712

Sun S, Jin F (2011) Robust co-training. Int J Pattern Recognit Artif Intell 25(07):1113–1126

Sun S, Shawe-Taylor J, Mao L (2017) Pac-bayes analysis of multi-view learning. Inf Fusion 35:117–131

Zhao J, Xie X, Xu X, Sun S (2017) Multi-view learning overview: Recent progress and new challenges. Inf Fusion 38:43–54

Chapter 2
Multiview Semi-supervised Learning

Abstract Semi-supervised learning is concerned with such learning scenarios where only a small portion of training data are labeled. In multiview settings, unlabeled data can be used to regularize the prediction functions, and thus to reduce the search space. In this chapter, we introduce two categories of multiview semi-supervised learning methods. The first one contains the co-training style methods, where the prediction functions from different views are trained through their own objective, and each prediction function is improved by the others. The second one contains the co-regularization style methods, where a single objective function exists for the prediction functions from different views to be trained simultaneously.

2.1 Introduction

In many real-world machine learning problems, the acquisition of labeled training data is costly while unlabeled ones are cheap. To save cost, in some scenarios, only a small portion of training data is labeled and semi-supervised learning works in these cases. The key to a successful semi-supervised learning method is the ability of boosting the performance with the abundant unlabeled data. Since one of the earliest multiview learning schemes, co-training (Blum and Mitchell 1998), was purposed for semi-supervised learning, it is natural to take the multiview semi-supervised learning as the first step to the multiview learning world.

In multiview learning settings, each data point $\mathbf{x}_n = \{\mathbf{x}_n^v\}_{v\in[V]}$ is a sample of random variable $X = \{X^v\}_{v\in[V]}$ with sample space $\mathbb{R}^{D_1} \times \cdots \times \mathbb{R}^{D_V}$. A multiview dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,\ldots,V,\ l=1,\ldots,L}$ is composed of L labeled samples with target y_l, each of which is represented by V views. $\mathbf{x}_l^v$ indicates the lth labeled example from the vth view. In semi-supervised setting, we also have access to an unlabeled dataset $\mathscr{U} = \{\mathbf{x}_u^v\}_{v=1,\ldots,V,\ u=1,\ldots,U}$ of size U. For the target, we have $y_l \in \mathbb{R}$ for regression and $y_l \in \mathbb{N}$ for classification problems. We mainly focus ourselves on two-view binary classification problems, namely, $y_l \in \{0, 1\}$ and $V = 2$.

We classify the semi-supervised multiview learning methods of interest to us into two groups: the co-training style methods and the co-regularization style methods. For the co-training style methods, the prediction functions from different views

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_2

are trained through their own objective, and each prediction function is improved by the others. There are usually information flowing explicitly between views. For co-regularization style methods, there is a single objective function for the prediction functions from different views to be trained simultaneously. Usually, the object consists of the loss functions for each view and a so-called co-regularization term encoding a desired consensus shared by views. We will introduce some of these two groups of algorithms in sequence in this section, followed by a brief summary of other methods.

2.2 Co-training Style Methods

In the first class of multiview semi-supervised learning algorithms, we are going to introduce the co-training style methods, which origins from the earliest semi-supervised multiview learning algorithm, co-training (Blum and Mitchell 1998). We will begin with an introduction to co-training, followed by some discussions about its successors.

2.2.1 Co-training

As the earliest multiview learning algorithm for semi-supervised learning, the basic idea of co-training is quite straightforward. Rather than designing a model which can learn directly from unlabeled data, co-training seeks for a training procedure to incorporate the information from unlabeled data into the models tailored for supervised learning. Since these models can only learn from labeled data, co-training tries to label the unlabeled examples and add them into the labeled training set. This can be done easily when data have multiple views.

Suppose the samples are from a random variable $X = \{X^1, X^2\}$ with sample space $\mathbb{R}^{D_1} \times \mathbb{R}^{D_2}$. We assume the target y is generated by some underlying procedure (maybe stochastic) $y = g(X^1, X^2)$. Co-training seeks a function $f(X) = (f^1(X^1), f^2(X^2))$ from some function space $\mathscr{F} = \mathscr{F}_1 \times \mathscr{F}_2$ on $\mathbb{R}^{D_1} \times \mathbb{R}^{D_2}$, e.g., the space of linear functions or neural networks. The function f is expected to be a best approximation of g from $\mathscr{F}$ in the sense of minimizing some loss function.

To motivate, we first consider one property f should have, and how this property improves the classifiers with unlabeled data. This property we desire is called compatibility, that is, the classifiers should output the correct label for labeled data, and their outputs should also agree for unlabeled data. Namely, $f^1(\mathbf{x}_l^1) = f^2(\mathbf{x}_l^2) = y_l$, and $f^1(\mathbf{x}_u^1) = f^2(\mathbf{x}_u^2)$. The constraint on labeled data suggests that f should approximate g correctly. The agreement on unlabeled data reduces the function space $\mathscr{F}$. With the increasing amount of available unlabeled data, it constrains the feasible set to be a much smaller and simpler subset of $\mathscr{F}$, thus helping to improve the classifiers.

The key to the success of co-training is how to impose the agreement on unlabeled data, which is achieved by an iterative label exchange between two views. Denote $\mathbb{L}_0^1 = \{\mathbf{x}_l^1, y_l\}_{l\in[L]}$, $\mathbb{L}_0^2 = \{\mathbf{x}_l^2, y_l\}_{l\in[L]}$, $\mathbb{U}_0^1 = \{\mathbf{x}_u^1\}_{u\in[U]}$, and $\mathbb{U}_0^2 = \{\mathbf{x}_u^2\}_{u\in[U]}$. To begin with, we train two weak classifiers for each view using labeled data, namely, training f^1 on $\mathbb{L}_0^1$ and f^2 on $\mathbb{L}_0^2$. Then we label the unlabeled data with these classifiers. Because of the lack of labeled data, these classifiers usually perform poorly at beginning. Thus for each class, only one example with highest classification confidence is remained. We denote the examples remained from view 2 and view 1 at the tth iteration as $\mathbb{S}_t^1 = \{\mathbf{x}_i^1, f^2(\mathbf{x}_i^2)\}$ and $\mathbb{S}_t^2 = \{\mathbf{x}_j^2, f^1(\mathbf{x}_j^1)\}$, respectively. After that, the classifier from one view is then updated by retraining it on the labeled dataset augmented with the example labeled by the other view, that is, training f^1 on $\mathbb{L}_t^1 = \mathbb{L}_{t-1}^1 \bigcup \mathbb{S}_t^1$ and f^2 on $\mathbb{L}_t^2 = \mathbb{L}_{t-1}^2 \bigcup \mathbb{S}_t^2$. These examples are then removed from the unlabeled datasets, $\mathbb{U}_t^1 = U_{t-1}^1 - \mathbb{S}_t^1$ and $\mathbb{U}_t^2 = \mathbb{U}_{t-1}^2 - \mathbb{S}_t^2$. These classifiers are updated iteratively, until all of the unlabeled data are labeled, namely, $\mathbb{U}_t^1 = \mathbb{U}_t^2 = \varnothing$.

Since this iteration procedure keeps pushing the classifier toward a set of compatible functions, a classifier well approximating the true label generator can be obtained.

2.2.2 Co-EM

One drawback of the original co-training algorithm is that the classifiers are only improved by one sample during each iteration, which makes it converge slowly and thus hinders its application to a moderate-sized dataset. Different methods have been proposed to alleviate this problem, among which a simple and efficient one is co-EM (Nigam and Ghani 2000).

When dealing with unlabeled data, a natural idea is to treat the unobservable labels as missing data, and use the expectation-maximization (EM) technique to estimate both the model parameters and the missing labels. To apply EM, one needs to probabilistically model the data generation process. A commonly used probabilistic model for co-EM is naive Bayes.

To begin with, we first briefly introduce naive Bayes in the single view supervised setting, and then illustrate how to handle unlabeled data with EM. The extension to the multiview setting is quite straightforward and will be given later.

Consider a classification problem with input variable $X \in \mathbb{R}^{\mathbb{D}}$ and target variable $y \in \{0, 1\}$. Again we consider binary classification for clarity. Naive Bayes assumes all the features of X are independent given y namely,

$$p(X_1, X_2, \ldots, X_D|y) = \prod_{d=1}^{D} p(X_d|y),$$

where X_d is the dth feature of X.

According to Bayesian rules, the posterior of y is given by

$$p(y|X) \propto \prod_{i=1}^{d} p(X_i|y)p(y),$$

where $p(y)$ is the prior class distribution which is usually approximated by the empirical distribution. With the posterior, we can make prediction using the maximum a posteriori rule,

$$f(X) = \arg\max_{y} p(y|X).$$

A concrete implementation of Naive Bayes also needs to assume the parametric form of the conditional distribution $p(X_d|y)$, and the parameters are estimated by maximizing the joint likelihood $p(X, y)$.

Now, we come to the semi-supervised setting. In this case, we also have a bunch of unlabeled data, with the corresponding target variable treated as missing values. We use EM to estimate the parameters of naive Bayes. Denote the labeled and unlabeled data as $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$ and $\mathscr{U} = \{\mathbf{x}_u\}_{u=1,\ldots,U}$, respectively. EM is an iterative procedure which consists of two basic steps. E-step uses the current estimate of model parameters to predict the label for each $\mathbf{x}_u \in \mathbb{U}$. M-step reestimates the parameters by performing maximum likelihood over $\mathbb{D} = \mathbb{L} \bigcup \mathbb{U}$, with missing labels from E-step. These two steps are iterated until convergence. A first estimate of the parameters is given by maximum likelihood over the labeled data $\mathbb{L}$.

Empirically, EM with naive Bayes converges much faster than co-training, while co-training usually outperforms EM when there exists a natural view partition of data features. Combining the two methods gives an algorithm that exhibits both merits, co-EM. The idea is to use co-training to combine multiple views, and EM to update classifiers with unlabeled data. Since a probabilistic model is needed, naive Bayes is again used as the basic classifiers. In detail, we first train a Naive Bayes classifier for view 1 with labeled data and use it to predict the unlabeled data. Then these newly labeled data are used to train a naive Bayes classifier for view 2. After that, labeled data from view 2 are used to retrain the classifier for view 1. These steps are repeated until convergence.

The main advantage of co-EM over co-training is to label all the unlabeled data and update classifiers at each iteration, while co-training only learns from one sample each time. Thus co-EM exhibits both the abilities of learning from multiview data of co-training and fast convergence from EM.

2.2.3 Robust Co-training

As we introduced above, co-training propagates information between two views by labeling the unlabeled samples with the classifier from one view and adding them to the labeled sample set for the other view iteratively. Since the classifiers are retrained with the augmented sample sets in each iteration, the quality of the classifiers

highly depends on those additionally labeled samples, especially when the original labeled sample set is small (which is usually the case in practice). Even though only high-confidently labeled samples are propagated, there is no promise on the quality of their labels. Robust co-training (Sun and Jin 2011) was proposed to address this problem.

The idea of robust co-training is to perform a test step called label inspection before propagating the predicted label. To do this, we need some criterion to qualify the label assigned to an unlabeled sample. Since the only information of great confidence about the labels comes from the original labeled sample set, and it is a natural assumption that similar samples have the same label. One let an unlabeled sample pass the test if the label assigned to it is the same as the labeled sample which is most the similar to it. So, the problem reduces to measure the similarity between two samples basing on their distance. Although we can directly operate on the original feature space, there are two main drawbacks. First, since the original feature space does not necessarily reflect the intrinsic geometry of the sample space and a sample can be represented by multiple redundant views, two close samples may not be of high similarity. Second, the original feature is usually of high dimension thus calculating the distance is computationally inefficient and suffers the curse of dimensionality. Therefore, we use a more practical scheme that is to project the features of the samples to a space of lower dimension which better captures the intrinsic geometry of the samples, and to measure the similarity therein. Fortunately, there exists a widely used method that provides such a projection with the properties we desire, canonical correlation analysis (CCA).

CCA is a classical dimension reduction method for multiview data. It seeks two linear mappings which project the features from two views to a shared linear space such that the projections of the features are maximally correlated. Note that, CCA itself is a method of great importance and inspires a bunch of multiview algorithms. Here, we will briefly introduce CCA and a detailed discussion will be given in Chap. 3.

Let $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ be two data matrix whose rows contain all the samples from view 1 and view 2, respectively. Let $\boldsymbol{\Sigma}^1$ and $\boldsymbol{\Sigma}^2$ be the covariance matrix of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, respectively, and $\boldsymbol{\Sigma}^{12}$ is the covariance matrix between $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$. The objective function of CCA is

$$\max_{\mathbf{w}^1,\mathbf{w}^2} \mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}^{12} \mathbf{w}^2$$
$$s.t. \begin{cases} \mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}^1 \mathbf{w}^1 = 1, \\ \mathbf{w}^{2\mathsf{T}} \boldsymbol{\Sigma}^2 \mathbf{w}^2 = 1, \end{cases}$$

where $\mathbf{w}^1$ and $\mathbf{w}^2$ are the parameters for the linear mappings.

The solution is obtained by solving the following generalized eigenvalue decomposition problem:

$$\boldsymbol{\Sigma}^{12}(\boldsymbol{\Sigma}^2)^{-1}(\boldsymbol{\Sigma}^{12})^{\mathsf{T}}\mathbf{w}^1 = \lambda \boldsymbol{\Sigma}^1 \mathbf{w}^1,$$

and

$$\mathbf{w}^2 = \lambda^{-\frac{1}{2}}(\boldsymbol{\Sigma}^2)^{-1}(\boldsymbol{\Sigma}^{12})^\intercal \mathbf{w}^1.$$

A K dimensional projections can be given by

$$(P^1(\mathbf{x}^1), P^2(\mathbf{x}^2)) = ([P_1^1(\mathbf{x}^1), \ldots, P_K^1(\mathbf{x}^1)]^\intercal, [P_1^2(\mathbf{x}^2), \ldots, P_K^2(\mathbf{x}^2)]^\intercal),$$

where $P_k^v(\cdot) = \mathbf{w}_k^{v\intercal} \cdot$ with $\mathbf{w}_k^v$ the eigenvector corresponding to the kth largest eigenvalue λ_k.

With the projections, the similarity between the sample to be inspected $\mathbf{x}_u$ and the lth labeled sample $\mathbf{x}_l$ is given by Zhou et al. (2007)

$$sim_l = \sum_{k=1}^{K} \lambda_k sim_{lk},$$

with

$$sim_{lk} = \exp\{-(P_k^1(\mathbf{x}_u^1) - P_k^1(\mathbf{x}_l^1))^2\} + \exp\{-(P_k^2(\mathbf{x}_u^2) - P_k^2(\mathbf{x}_l^2))^2\}.$$

Since only when the label assigned to an unlabeled sample is the same as the labeled sample of the most similarity will be used to teach the classifier from the other view, robust co-training well improves the classifiers learned in the following iterations. The original co-training lacks such a label inspection scheme.

2.3 Co-regularization Style Methods

The methods we introduced so far utilizes an explicit iterative procedure to propagate information between views. In this section, we will discuss the methods of learning classifiers simultaneously with one single objective function. We call them the co-regularization style methods, among which the most representative one is the co-regularization algorithm.

2.3.1 Co-regularization

Co-regularization (Sindhwani et al. 2005) performs multiview learning under the framework of regularization. In the regularization framework, a function $f^* \in \mathscr{F}$ is learned by minimizing the following regularized empirical risk function:

$$f_* = \arg\min_{f \in \in \mathscr{F}} \frac{1}{L} \sum_{l=1}^{L} l(\mathbf{x}_l, y_l, f(\mathbf{x}_l)) + \lambda ||f||^2_{\mathscr{F}}.$$

Here, $(\mathbf{x}_l, y_l)$ is an example from a labeled dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$ of size L, $l(\cdot)$ is some loss function, $\mathscr{F}$ is a reproducing kernel Hilbert space (RKHS) of functions, and $||\cdot||_{\mathscr{F}}$ is the norm in $\mathscr{F}$ as a regularization term. λ is a trade-off parameter leveraging fitting and regularization. Note that, the regularization term constrains the function space to reduce the complexity of the model and thus avoid over-fitting. Recall that the co-training algorithm also constrains the function by assuming the compatibility of the classifier. Thus it is a natural idea to use the unlabeled data to construct a regularization term.

Recall that we hope the classifiers from different views predict correctly on labeled samples and agree on unlabeled samples. Co-regularization solves the following minimization problem to obtain the solution:

$$\begin{aligned}(f^1_*, f^2_*) = \arg\min_{f^1 \in \mathscr{F}_1, f^2 \in \mathscr{F}_2} & \gamma \sum_{l=1}^{L} l(\mathbf{x}^1_l, y_l, f(\mathbf{x}^1_l)) + (1-\gamma) \sum_{l=1}^{L} l(\mathbf{x}^2_l, y_l, f(\mathbf{x}^2_l)) \\ & + \lambda_1 ||f^1||^2_{\mathscr{F}} + \lambda_2 ||f^2||^2_{\mathscr{F}} + \frac{\lambda_c}{L+U} \sum_{t=1}^{L+U} (f^1(\mathbf{x}^1_n) - f^2(\mathbf{x}^2_n))^2 \quad (2.1)\end{aligned}$$

The last term is a regularization term penalizing the risk with the disagreement between the classifiers on the whole training set. Here $\gamma \in [0, 1]$ is a parameter to balance the views, and λ_1, λ_2, λ_c are parameters leveraging regularization on the classifiers.

Once the loss function l is convex, (2.1) is convex and has a unique minimum. A representer theorem suggests that the minimizer (f^1_*, f^2_*) has the following form:

$$(f^1_*(\mathbf{x}^1), f^2_*(\mathbf{x}^2)) = \left(\sum_{t=1}^{L+U} \alpha_t k^1(\mathbf{x}^1, \mathbf{x}^1_t), \sum_{t=1}^{L+U} \beta_t k^2(\mathbf{x}^2, \mathbf{x}^2_t) \right),$$

where $k^1(\cdot,\cdot)$ and $k^2(\cdot,\cdot)$ are the kernel functions of the RKHS $\mathscr{F}_1$ and $\mathscr{F}_2$, respectively. The expansion coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can be obtained by solving the following linear system:

$$\begin{aligned}\left(\frac{\gamma}{L} diag(K^1) K^1 + \lambda_1 I + \frac{\lambda_c}{L+U} K^1 \right) \boldsymbol{\alpha} - \frac{\lambda_c}{L+U} K^2 \boldsymbol{\beta} &= \frac{\gamma}{L} \mathbf{y}, \\ \left(\frac{1-\gamma}{L} diag(K^2) K^2 + \lambda_1 I + \frac{\lambda_c}{L+U} K^2 \right) \boldsymbol{\beta} - \frac{\lambda_c}{L+U} K^1 \boldsymbol{\alpha} &= \frac{1-\gamma}{L} \mathbf{y}.\end{aligned}$$

Here, $\boldsymbol{K}^v$ is the kernel matrices with $\boldsymbol{K}^v_{ij} = k^v(\mathbf{x}^v_i . \mathbf{x}^v_j)$, $i, j \in [L+U]$, and $diag(\boldsymbol{K}^v)$ is a diagonal matrix consists of the diagonal elements of $\boldsymbol{K}^v$. $\mathbf{y}$ is a vector of the labels, where

$$\mathbf{y}_t = \begin{cases} y_t, & 0 \le t \le L \\ 0, & L \le t \le L+U \end{cases}.$$

One may also exploit other information to co-regularize the classifiers. One example is the co-Laplacian. As the name suggests, it constructs a similarity graph for each view, and incorporates the graph Laplacians into the co-regularization term. In this case, the similarity between the input variables is also taken into consideration.

2.3.2 Bayesian Co-training

In this section, we discuss Bayesian co-training (Yu et al. 2011), which performs semi-supervised learning in the nonparametric Bayesian framework. In particular, Bayesian co-training constructs an undirected graphic model for co-training. Note that although it is named Bayesian co-training, the model is learned by maximizing a single marginal likelihood which resembles the co-regularization target function. Thus, we take it as a co-regularization style method.

A commonly used tool for modeling the uncertainty of a function is the Gaussian process (GP). We first briefly introduce the GP model for single-view supervised learning.

A GP defines a nonparametric prior over functions. We say a function $f \sim GP(m, k)$, if for any finite sample set $\{\mathbf{x}_n\}_{n\in[N]}$, the outputs of the function $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N))^\intercal$ have a multivariate Gaussian distribution $N(\mathbf{m}, \boldsymbol{K})$. Here, $\mathbf{m}$ is a vector of the outputs of a mean function m over $\{\mathbf{x}_n\}_{n\in[N]}$, and $\boldsymbol{K}$ is a kernel matrix whose element $\boldsymbol{K}_{ij}$ is given by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. A GP is parameterized by the mean function m and the kernel function k. We set $m = 0$ and use the RBF kernel ($k(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\eta}\}$, $\eta > 0$) for illustration.

A GP model models the conditional distribution of the target variable y given input variable X, $p(y|X)$. It assumes there exists a latent function f with GP prior $GP(0, k)$, and $p(y|X) = \int p(y|f, X)p(f)df$. The model can also be described as an undirected model. Suppose we have a labeled dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$. Let $\mathbf{y} = [y_1, \ldots, y_L]^\intercal$ and $\mathbf{f} = [f_1, \ldots, f_L]^\intercal$ with $f_l = f(\mathbf{x}_l)$. The joint distribution $p(\mathbf{y}, \mathbf{f}) = \frac{1}{Z}\psi(\mathbf{f}) \prod_{l=1}^{L} \psi(y_l, f_l)$, where

$$\begin{aligned} \psi(\mathbf{f}) &= \exp\{\frac{1}{2}\mathbf{f}^\intercal \mathbf{K}^{-1}\mathbf{f}\}, \\ \psi(y_l, f_l) &= \begin{cases} \exp\{-\frac{1}{2\sigma^2}||y_l - f_l||^2\}, & for\ regression \\ \lambda(y_l f_l), & for\ classification \end{cases}. \end{aligned} \tag{2.2}$$

Here $\lambda(\cdot)$ is the sigmoid function and $\mathbf{Z}$ are the normalization constant.

Now consider the multiview learning setting, with a multiview dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,\ldots,V,\ l=1,\ldots,L}$. Bayesian co-training assumes that there exists one latent function $f^v \sim GP(0, k^v)$ for each view, which is parameterized by its own kernel parameter η^v. It also introduces a consensus latent function f^c, ensuring the consensus between the latent functions of different views. The relationship between f^c and f^v is modeled by the following potential function:

$$\psi(f^c, f^v) = \exp\{-\frac{1}{2\sigma_v^2}||f^c - f^v||^2\}.$$

Let $\mathbf{f}^v = [f_1^v, \ldots, f_L^v]^\intercal$ and $\mathbf{f}^c = [f_1^c, \ldots, f_L^c]^\intercal$. The joint distribution of $\mathbf{y}$, $\mathbf{f}^c$ and $\mathbf{f}^v$ is

$$p(\mathbf{y}, \mathbf{f}^c, \{\mathbf{f}^v\}) = \frac{1}{Z}\prod_{l=1}^{L}\psi(y_l, \mathbf{f}_l^c)\prod_{v=1}^{V}\psi(\mathbf{f}^v)\psi(\mathbf{f}^c, \mathbf{f}^v), \tag{2.3}$$

where $\psi(y_l, f_l^c)$ and $\psi(\mathbf{f}^v)$ are given by (2.2).

In the semi-supervised learning setting, we also have access to an unlabeled dataset $\mathscr{U} = \{\mathbf{x}_u^v\}_{v=1,\ldots,V,\ u=1,\ldots,U}$. It is easy to incorporate the unlabeled samples into a Bayesian framework. The idea is to consider the unobservable labels as latent variables and marginalize them out. In this case, let $\mathbf{f}^v = [f_1^v, \ldots, f_{L+U}^v]^\intercal$ and $\mathbf{f}^c = [f_1^c, \ldots, f_{L+U}^c]^\intercal$. The joint distribution again follows (2.3). Note that the product of the output potentials contains only the labeled data samples.

The parameters of Bayesian co-training include the view-wise kernel parameters $\{\eta_v\}$ and the parameters in the potential functions. These parameters are learned by maximizing the marginal likelihood $p(\mathbf{y})$, which is obtained by integrating out the latent functions $\mathbf{f}^c$ and $\{\mathbf{f}^v\}$. Unless the output potential $\psi(y_l, f_l^c) = \exp\{-\frac{1}{2\sigma^2}||y_l - f_l^c||^2\}$, the integral is intractable. Different approximations can be applied to handle this problem, which is out of the scope of this book. Since it is empirically shown that satisfying performance of the classification task was obtained under the regression setting, we will focus on the tractable case from now on.

Before calculating the marginal likelihood, we first calculate the marginal distribution of the latent functions to get some insight of the model. Integrating out $\mathbf{f}^c$ gives

$$p(\mathbf{y}, \{\mathbf{f}^v\}) = \frac{1}{\mathbf{Z}}\exp\{-\frac{1}{2}\sum_v\frac{||\mathbf{y} - \mathbf{f}^v||^2}{\sigma_v^2 + \sigma^2} - \frac{1}{2}\sum_v \mathbf{f}^{v\intercal}\mathbf{K}^{v-1}\mathbf{f}^v - \frac{1}{2}\sum_{v<u}\frac{||\mathbf{f}^v - \mathbf{f}^u||^2}{\sigma_v^2 + \sigma_u^2}\}. \tag{2.4}$$

Note that, the negative log of (2.4) recovers the target function of co-regularization with the least square loss.

We can also integrate out $\{\mathbf{f}^v\}$ from (2.3), resulting in

$$p(\mathbf{y}, \mathbf{f}^c) = \frac{1}{\mathbf{Z}}\exp\{-\frac{1}{2}\frac{||\mathbf{y} - \mathbf{f}^c||^2}{\sigma^2} - \frac{1}{2}\mathbf{f}^{c\intercal}\mathbf{K}^{c-1}\mathbf{f}^c\}, \tag{2.5}$$

where $\mathbf{K}^c = \left[\sum_v \left(\mathbf{K}^v + \sigma_v^2 \mathbf{I}\right)^{-1}\right]^{-1}$. This marginal distribution is indeed the same as that of a single-view GP model with one latent function $\mathbf{f}^c \sim GP(0, \mathbf{K}^c)$. The log marginal likelihood is given by

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y}, \mathbf{f}^c) d\mathbf{f}^c = -\frac{1}{2}\mathbf{y}^\intercal (\mathbf{K}^c + \sigma^2)^{-1}\mathbf{y} - \frac{1}{2}\log |\mathbf{K}^c + \sigma^2| - \frac{|L|}{2}\log 2\pi. \quad (2.6)$$

As mentioned before, the parameters of the model can be obtained by maximizing the log marginal likelihood (2.6).

To make prediction, we calculate the posterior of the consensus latent function $\mathbf{f}_*^c = \{f^c(\mathbf{x}_*)\}$ where $\mathbf{x}_*$ comes from a test sample set.

$$p(\mathbf{f}_*^c | \mathbb{L} \bigcup \mathbb{U}) = N(\boldsymbol{\mu}, \Sigma)$$

with

$$\boldsymbol{\mu} = \mathbf{K}_*^c (\mathbf{K}^c + \sigma^2 \mathbf{I})^{-1}\mathbf{y},$$
$$\Sigma = \mathbf{K}_{**}^c - \mathbf{K}_*^c (\mathbf{K}^c + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_*^{c\intercal},$$

where $\mathbf{K}_*^c$ contains the kernel function evaluated between training samples and test samples, and $\mathbf{K}_{**}^c$ contains the kernel function evaluated on test samples. Usually, the mean is used to make prediction and the covariance measures the uncertainty of the predictions.

It is easy to incorporate unlabeled samples in a Bayesian framework. The idea is to consider the unobservable labels as latent variables and marginalize them out.

2.3.3 *Multiview Laplacian SVM*

SVM can be formulated under the regularization framework. We can easily extend SVM to the multiview setting by fitting it to the co-regularization framework. Specifically, we can encode our assumption on the consensus between the classifiers from different views into a co-regularization term to regularize the classifiers. Indeed, different types of regularization can be performed on the function space, and multiple regularization can be easily integrated together thanks to the power of the regularization framework. In this subsection, we will introduce one such example for multiview semi-supervised learning, multiview Laplacian SVM (Sun 2011), which integrates three types of regularization, i.e., the function norm regularization, the multiview regularization and the manifold regularization.

Given a training set $\mathscr{L} \bigcup \mathscr{U}$ with $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$ and $\mathscr{U} = \{\mathbf{x}_u\}_{u=1,\ldots,U}$, the manifold regularization first constructs a data adjacency matrix $\boldsymbol{W}$. There are several graph construction method in literature, and here, we introduce the one used in the multiview Laplacian SVM paper (Sun 2011). The matrix $\boldsymbol{W}$ is of size $(L + U) \times (L + U)$, where

$$\boldsymbol{W}_{ij} = \exp\{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^2}\},$$

with a hyperparameter σ.

The manifold regularization for a function f is defined as

$$M(f) = \frac{1}{2}\sum_{i,j} \boldsymbol{W}_{ij}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2.$$

Let $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{L+U})]$. The regularization term can be rewritten as

$$M(f) = \frac{1}{2}\mathbf{f}^\intercal \boldsymbol{L}\mathbf{f},$$

where $\boldsymbol{L} = \boldsymbol{V} - \boldsymbol{W}$ with $\boldsymbol{V}$ a diagonal matrix whose diagonal element $\boldsymbol{V}_{ii} = \sum_j \boldsymbol{W}_{ij}$. $\boldsymbol{L}$ here is called the graph Laplacian of $\boldsymbol{W}$.

For a multiview dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,\ldots,V,\ l=1,\ldots,L}$ and $\mathscr{U} = \{\mathbf{x}_u^v\}_{v=1,\ldots,V,\ u=1,\ldots,U}$, we can use the co-regularization term introduced in (2.3.1) to regularize a pair of function f^1 and f^2. In particular, we have the following co-regularization term:

$$C(f^1, f^2) = \frac{1}{L+U}\sum_{n=1}^{L+U}(f^1(\mathbf{x}_n^1) - f^2(\mathbf{x}_n^2))^2. \tag{2.7}$$

For any function f, we can regularize it with its functional norm $||f||$. In particular, for a linear function $f(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x} + b$, we can regularize it with $||w||^2$.

Combining these three regularization terms, we obtain the objective function for multiview Laplacian SVM

$$\begin{aligned}\min_{f^1\in\mathscr{F}_1, f^2\in\mathscr{F}_2}\ & \frac{1}{2L}\sum_{l=1}^{L}\left[(1 - y_l f^1(\mathbf{x}_l^1))_+ + (1 - y_l f^2(\mathbf{x}_l^2))_+\right] + \gamma_1(||f^1||^2 + ||f^2||^2)\\ & + \frac{\gamma_2}{L+U}((\mathbf{f}^1)^\intercal \boldsymbol{L}^1\mathbf{f}^1 + (\mathbf{f}^2)^\intercal \boldsymbol{L}^2\mathbf{f}^2) + \frac{\gamma_3}{L+U}\sum_{t=1}^{L+U}(f^1(\mathbf{x}_t^1) - f^2(\mathbf{x}_t^2))^2,\end{aligned} \tag{2.8}$$

where $(\cdot)_+$ is the positive part of $\cdot$, L^1 and L^2 are the graph Laplacian of view 1 and view 2, respectively, and $\{\gamma_i\}$ are the trade-off parameters.

According to the representer theorem, the solution to (2.8) has the following form:

$$f^1(\cdot) = \sum_{n=1}^{L+U}\alpha_n^1 k_1(\mathbf{x}_n^1, \cdot),\ f^2(\cdot) = \sum_{n=1}^{L+U}\alpha_n^2 k_1(\mathbf{x}_n^2, \cdot),$$

where k_1 and k_2 are the kernels of the RKHS $\mathscr{F}_1$ and $\mathscr{F}_2$, respectively. Thus we have

$$||f^1|| = \boldsymbol{\alpha}^{\mathbf{1}\intercal}\boldsymbol{K}^1\boldsymbol{\alpha}^{\mathbf{1}},\ ||f^1|| = \boldsymbol{\alpha}^{\mathbf{2}\intercal}\boldsymbol{K}^2\boldsymbol{\alpha}^{\mathbf{2}}, \tag{2.9}$$

where $\boldsymbol{K}^1$ and $\boldsymbol{K}^2$ are the Gram matrices from view 1 and view 2, respectively, and $\boldsymbol{\alpha}^v$ is a vector that contains α_t^v. We also have

$$\mathbf{f}^1 = K_1\boldsymbol{\alpha}^1, \mathbf{f}^2 = K_2\boldsymbol{\alpha}^2. \tag{2.10}$$

With the above notions, the objective function of multiview Laplacian SVM can be rewritten as

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}^1,\boldsymbol{\alpha}^2} \gamma_1(\boldsymbol{\alpha}^{1\intercal}K_1\boldsymbol{\alpha}^1 + \boldsymbol{\alpha}^{2\intercal}K_2\boldsymbol{\alpha}^2) + \gamma_2(\boldsymbol{\alpha}^{1\intercal}K_1L_1K_1\boldsymbol{\alpha}^1 + \boldsymbol{\alpha}^{2\intercal}K_2L_2K_2\boldsymbol{\alpha}^2) \\
&\quad + \gamma_3(K_1\boldsymbol{\alpha}^1 - K_2\boldsymbol{\alpha}^2)^\intercal(K_1\boldsymbol{\alpha}^1 - K_2\boldsymbol{\alpha}^2) \\
&s.t. \begin{cases} y_i \sum\limits_{n=1}^{L+U} \alpha_t^1 k_1(\mathbf{x}_n^1, \mathbf{x}_i^1) \geq 1, \\ y_i \sum\limits_{n=1}^{L+U} \alpha_t^2 k_1(\mathbf{x}_n^2, \mathbf{x}_i^2) \geq 1. \end{cases}
\end{aligned}$$

This problem can be solved use the Lagrangian multiplier method.

2.3.4 *Multiview Laplacian Twin SVM*

SVM is a classical and effective classifier algorithm. However, restricted by the simple form of its discriminant function, SVM may fail to handle some particular simple dataset, even with the kernel tricks. To address this problem, twin SVM (TSVM) (Khemchandani et al. 2007) was proposed. Rather than using a single hyperplane to divide the data, TSVM approximates each class of data with one hyperplane, and predicts the label of a test sample using its distance to these hyperplanes. This idea can also be extended to the multiview setting. In this case, two hyperplanes are learned for each view (binary classification), and the distances from a sample to these hyperplanes for different views are forced to be close. Also, within the regularization framework, different regularizations can be performed to fit TSVM to various learning settings. Specifically, manifold regularization can be used to extend TSVM to semi-supervised learning problems. Taking all of these into consideration, we arrive at a multiview semi-supervised learning method, multiview Laplacian twin SVM (MvLapTSVM) (Xie and Sun 2014).

We first introduce TSVM briefly in the single-view supervised learning setting. Consider a labeled dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$. We construct two data matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ whose rows are examples from the positive class and the negative class, respectively. TSVM seeks two hyperplanes

$$\begin{aligned}
\mathbf{w}_1^\intercal\mathbf{x} + b_1 = 0, \\
\mathbf{w}_2^\intercal\mathbf{x} + b_2 = 0,
\end{aligned}$$

each of which is close to the samples form one class and is far form the other samples. To simplify the notation, we can augment the covariate $\mathbf{x}$ with an additional feature of value 1, and absorb the bias b_i into $\mathbf{w}_i$. From now on, we assume all the data matrices are constructed by augmented examples. Notice that $\mathbf{w}_1^\intercal \mathbf{x}$ is the (rescaled) distance from $\mathbf{x}$ to hyperplane 1, and $A\mathbf{x}$ is the summation of the distances corresponding to all positive class samples. We can obtain two hyperplanes by solving the following two optimization problem separately:

$$\begin{aligned} \min_{\mathbf{w}_1} \quad & (\boldsymbol{A}\mathbf{w}_1)^\intercal(\boldsymbol{A}\mathbf{w}_1) \\ s.t. \quad & -\boldsymbol{B}\mathbf{w}_2 \geq \mathbf{1}, \end{aligned}$$

$$\begin{aligned} \min_{\mathbf{w}_2} \quad & (\boldsymbol{B}\mathbf{w}_2)^\intercal(\boldsymbol{B}\mathbf{w}_2) \\ s.t. \quad & -\boldsymbol{A}\mathbf{w}_2 \geq \mathbf{1}. \end{aligned}$$

Now suppose we also have an unlabeled sample set $\mathscr{U} = \{\mathbf{x}_u\}_{u=1,\ldots,U}$. To handle semi-supervised learning problems, we can integrate manifold regularization into TSVM, leading to the following objective function for Laplacian TSVM (Qi et al. 2012)

$$\begin{aligned} \min_{\mathbf{w}_1} \frac{1}{2} \quad & (\boldsymbol{A}\mathbf{w}_1)^\intercal(\boldsymbol{A}\mathbf{w}_1) + \frac{\gamma_1}{2}||\mathbf{w}_1||^2 + \frac{\gamma_2}{2}(M\mathbf{w}_1)^\intercal \boldsymbol{L}(\boldsymbol{X}\mathbf{w}_1) \\ s.t. \quad & -\boldsymbol{B}\mathbf{w}_2 \geq \mathbf{1}, \end{aligned}$$

$$\begin{aligned} \min_{\mathbf{w}_2} \frac{1}{2} \quad & (\boldsymbol{B}\mathbf{w}_2)^\intercal(\boldsymbol{B}\mathbf{w}_2) + \frac{\gamma_1}{2}||\mathbf{w}_2||^2 + \frac{\gamma_2}{2}(M\mathbf{w}_2)^\intercal \boldsymbol{L}(\boldsymbol{X}\mathbf{w}_2) \\ s.t. \quad & -A\mathbf{w}_2 \geq \mathbf{1}. \end{aligned}$$

Here, $\boldsymbol{X}$ is the data matrix contains all the labeled and unlabeled data, $\boldsymbol{L}$ is the graph Laplacian, and $\{\gamma_i\}$ are the trade-off parameters as usual. Note that we also use function norm regularization terms.

Now come to the multiview case, with dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,2,\ l=1,\ldots,L}$ and $\mathscr{U} = \{\mathbf{x}_u^v\}_{v=1,2,\ u=1,\ldots,U}$. In order to extend Laplacian TSVM to the multiview setting, we need to pose some consensus constraints on the hyperplanes from different views. Since the label of a test sample is assigned according to its distance to the hyperplanes, one way to exploit consensus is to enforce the difference between a sample's distances to the hyperplanes of the same class from different views smaller than some threshold, that is,

$$|(\mathbf{w}_1^1)^\intercal \mathbf{x}_l^1 - (\mathbf{w}_1^2)^\intercal \mathbf{x}_l^2| < \eta,$$

where $\mathbf{w}_i^v$ is the parameter of the hyperplane for the ith class from the vth view, and $\eta \geq 0$ is a variable to be optimized, controlling the degree of view consensus.

Let $\boldsymbol{A}^v$ and $\boldsymbol{B}^v$ be the data matrix of positive and negative samples from view v, respectively. The two objective functions of MvLapTSVM are

$$\begin{aligned}\min_{\mathbf{w}_1^1,\mathbf{w}_1^2,\eta_1} & \frac{1}{2}(\boldsymbol{A}^1\mathbf{w}_1^1)^\intercal(\boldsymbol{A}^1\mathbf{w}_1^1)+\frac{1}{2}(\boldsymbol{A}^2\mathbf{w}_1^2)^\intercal(\boldsymbol{A}^2\mathbf{w}_1^2)+\frac{\gamma_1}{2}(||\mathbf{w}_1^1||^2+||\mathbf{w}_1^2||^2)\\ &+\frac{\gamma_2}{2}\left(\left(\boldsymbol{X}^1\mathbf{w}_1^1\right)^\intercal\boldsymbol{L}^1\left(\boldsymbol{X}^1\mathbf{w}_1^1\right)+\left(\boldsymbol{X}^2\mathbf{w}_1^2\right)^\intercal\boldsymbol{L}^2\left(\boldsymbol{X}^2\mathbf{w}_1^2\right)\right)\\ &+\gamma_3\mathbf{1}^\intercal\boldsymbol{\eta}_1\\ s.t. & \begin{cases}|\boldsymbol{A}^1(\mathbf{w}_1^1)-\boldsymbol{A}^2(\mathbf{w}_1^2)|<\boldsymbol{\eta}_1,\\ -\boldsymbol{B}^1\mathbf{w}_1^1\geq\mathbf{1},\\ -\boldsymbol{B}^2\mathbf{w}_1^2\geq\mathbf{1},\\ \boldsymbol{\eta}_1\geq 0,\end{cases}\end{aligned}$$

$$\begin{aligned}\min_{\mathbf{w}_2^1,\mathbf{w}_2^2,\eta_2} & \frac{1}{2}(\boldsymbol{B}^1\mathbf{w}_2^1)^\intercal(\boldsymbol{B}^1\mathbf{w}_2^1)+\frac{1}{2}(\boldsymbol{B}^2\mathbf{w}_2^2)^\intercal(\boldsymbol{B}^2\mathbf{w}_2^2)+\frac{\gamma_1}{2}(||\mathbf{w}_2^1||^2+||\mathbf{w}_2^2||^2)\\ &+\frac{\gamma_2}{2}\left(\left(\boldsymbol{X}^2\mathbf{w}_2^1\right)^\intercal\boldsymbol{L}^1\left(\boldsymbol{X}^1\mathbf{w}_2^1\right)+\left(\boldsymbol{X}^2\mathbf{w}_2^2\right)^\intercal\boldsymbol{L}^2\left(\boldsymbol{X}^2\mathbf{w}_2^2\right)\right)\\ &+\gamma_3\mathbf{1}^\intercal\boldsymbol{\eta}_2\\ s.t. & \begin{cases}|\boldsymbol{B}^1(\mathbf{w}_1^1)-\boldsymbol{B}^2(\mathbf{w}_1^2)|<\boldsymbol{\eta}_1,\\ -\boldsymbol{A}^1\mathbf{w}_1^1\geq\mathbf{1},\\ -\boldsymbol{A}^2\mathbf{w}_1^2\geq\mathbf{1},\\ \boldsymbol{\eta}_1\geq 0.\end{cases}\end{aligned}$$

Here, $\boldsymbol{X}^v$ is a data matrix contains all labeled and unlabeled samples from view v, and $\boldsymbol{L}^v$ is the graph Laplacian of view v. These two optimization problems can be solved separately using the Lagrangian multiplier method. To make prediction, we use the following rule:

$$y=\begin{cases}1, & if\ \ |(\mathbf{w}_1^1)^\intercal\mathbf{x}^1|+|(\mathbf{w}_1^2)^\intercal\mathbf{x}^2|<|(\mathbf{w}_2^1)^\intercal\mathbf{x}^1|+|(\mathbf{w}_2^2)^\intercal\mathbf{x}^2|,\\ -1, & else.\end{cases}$$

Note that the kernel trick can be easily applied to MvLapSVM, just like the original SVM.

2.4 Other Methods

Co-training Under Weak Dependence (Blum and Mansour 2017): The original co-training is developed under the assumption of view independence, that is, different views of the input variable are assumed to be independent given the response variable. To make co-training suitable for practical problems, it is appealing to develop a

polynomial-time algorithm for co-training under a relaxed assumption, e.g., the weak dependence assumption. Under weak dependence, a multiview example is generated in a partly random fashion. Specifically, the two views are assumed to be conditional independence with a probability p, and be arbitrarily correlated with probability $1 - p$. This setting is solvable from an information theory view. The algorithm seeks two linear discriminant functions depending on the assumption that the true classifiers assign the same label to two views of an example at some margin. Under this assumption, the product of predictions from two views should be equal to or greater than the square of the margin. This inequality can be used to solve the outer product of the parameters of the two discriminant functions. Then the product is used to compute the discriminant functions.

Sparse Multiview Support Vector Machines (Sun and Shawe-Taylor 2010): Sparse multiview support vector machine (SMvSVM) is an instantiation of a general sparse semi-supervised learning framework, which makes use of Fenchel–Legendre conjugates to rewrite a convex insensitive loss involving a regularization term with unlabeled data. Specifically, SMvSVM seeks two discriminant functions from two RKHSs, one for each view. The functions are learned through minimizing an objective function, which is a combination of two independent Tikhonov regularization problems with hinge loss depending on labeled data, as well as a data-dependent convex insensitive loss which penalizes the disagreement between views. This convex loss is rewritten by using Fenchel–Legendre conjugates that make the problem much easier to solve. According to a representer theorem, each of the solutions can be represented as a linear combination of the kernel of the corresponding RKHS evaluated at the training examples. Thus, the optimization problem can be transferred to a finite-dimensional problem. Using the Lagrangian dual, one can obtain the dual problem of the objective function, which has much less optimization variables in typical semi-supervised learning settings and simpler constraints.

Multiple-View Multiple-learner Semi-supervised Learning (Sun and Zhang 2011): Multiple-view multiple-learner (MVML) is a framework for multiview semi-supervised learning, which combines the power of multiview learning and ensemble learning. The basic idea of MVML is to learn one ensemble of classifiers from each view. By updating these ensembles with a co-training-like algorithm, these ensembles can efficiently exploit the unlabeled data as well as the multiview nature of them. Finally, some combination criterion is used to integrate the learned ensembles to produce a prediction.

Semi-supervised Multiview Maximum Entropy Discrimination (Chao and Sun 2019): Multiview maximum entropy discrimination (MvMED) combines the large margin principle and Bayesian learning under multiview learning settings. Specifically, MvMED seeks the posterior distributions of the parameters of two discriminant functions which classify two views of an example at a margin, and force the posterior distributions of the margin variables from different views to be identical. In order to exploit the unlabeled data, semi-supervised multiview maximum entropy discrimination (SMvMED) utilizes the idea of Laplacian regularization. Since the

predictions from each view are random variables, this regularization is performed by taking expectation with respect to the posteriors of the parameters. The optimization problem is solved by optimizing its Lagrangian dual.

References

Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th annual conference on computational learning theory, ACM, pp 92–100

Blum A, Mansour Y (2017) Efficient co-training of linear separators under weak dependence. In: Proceedings of the 30th annual conference on learning theory, pp 302–318

Chao G, Sun S (2019) Semi-supervised multi-view maximum entropy discrimination with expectation laplacian regularization. Inf Fusion 45:296–306

Khemchandani R, Chandra S et al (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910

Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. In: Proceedings of the 9th international conference on information and knowledge management, ACM, pp 86–93

Qi Z, Tian Y, Shi Y (2012) Laplacian twin support vector machine for semi-supervised classification. Neural Netw 35(11):46–53

Sindhwani V, Niyogi P, Belkin M (2005) A co-regularization approach to semi-supervised learning with multiple views. Proc ICML Work Learn Mult Views ACM 2005:74–79

Sun S (2011) Multi-view laplacian support vector machines. In: Proceedings of the 7th international conference on advanced data mining and applications, Springer, pp 209–222

Sun S, Jin F (2011) Robust co-training. Int J Pattern Recognit Artif Intell 25(7):1113–1126

Sun S, Shawe-Taylor J (2010) Sparse semi-supervised learning using conjugate functions. J Mach Learn Res 11(9):2423–2455

Sun S, Zhang Q (2011) Multiple-view multiple-learner semi-supervised learning. Neural Process Lett 34(3):229–240

Xie X, Sun S (2014) Multi-view laplacian twin support vector machines. Appl Intell 41(4):1059–1068

Yu S, Krishnapuram B, Rosales R, Rao RB (2011) Bayesian co-training. J Mach Learn Res 12(9):2649–2680

Zhou ZH, Zhan DC, Yang Q (2007) Semi-supervised learning with very few labeled training examples. In: Proceedings of the 22nd AAAI national conference on artificial intelligence, AAAI, vol 1, pp 675–680

Chapter 3
Multiview Subspace Learning

Abstract In multiview settings, observations from different views are assumed to share the same subspace. The abundance of views can be utilized to better explore the subspace. In this chapter, we consider two different kinds of multiview subspace learning problems. The first one contains the general unsupervised multiview subspace learning problems. We focus on canonical correlation analysis as well as some of its extensions. The second one contains the supervised multiview subspace learning problems, i.e., there exists available label information. In this case, representations more suitable for the on-hand task can be obtained by utilizing the label information. We also briefly introduce some other methods at the end of this chapter.

3.1 Introduction

A common assumption underlying many machine learning algorithms is the manifold assumption, i.e., the data observed lie near a lower dimensional manifold. Recovering such a manifold benefits downstream tasks since operating on a lower dimensional representation is more computationally efficient and the target function on the manifold can be expected to be in a simpler function class. What is more, the recovered manifold usually provides an easier visualization, thus giving a better data interpretation. In the multiview learning setting, since a data sample is represented by multiple distinct feature sets (views), the abundance of views can be utilized to better explore the manifold. In this chapter, we will focus on introducing multiview subspace learning which is based on the assumption that the underlying data manifold is a linear subspace. Since seeking a more general representation is still an open problem and is usually combined with deep learning, an introduction to such methods will be given in Chap. 8.

In multiview learning settings, each data point $\mathbf{x}_n = \{\mathbf{x}_n\}_{v\in[V]}$ is a sample of random variable $X = \{X^v\}_{v\in[V]}$ with sample space $\mathbb{R}^{D_1} \times \cdots \times \mathbb{R}^{D_V}$. A multiview dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,\ldots,V,\ l=1,\ldots,L}$ is composed of L labeled samples with target y_l, each of which is represented by V views. $\mathbf{x}_l^v$ indicates the l-th labeled sample from the v-th view.

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_3

The organization of this chapter is as follows. To begin with, we consider the general unsupervised subspace learning problem. The most representative solution to this problem is canonical correlation analysis (CCA). We will mostly focus on CCA as well as some of its extensions. After that, we consider the supervised subspace learning problem, i.e., there exists available label information. In this case, representations more suitable for the on-hand task can be obtained by utilizing the label information. We will show how different models handle labeled multiview data to learn a subspace. Finally, we end this chapter with a brief summary of other methods.

3.2 Canonical Correlation Analysis and Related Methods

Canonical correlation analysis (CCA) was first proposed in 1933 (Hotelling 1933) and is a widely used technique in the statistic community to measure the linear relationship between two multidimensional variables. It has got popular in the machine learning literature since the beginning of this century (Hardoon et al. 2004). Although we have already given a brief introduction to CCA in Chap. 2, here we start from scratch and focus on more details. We note that CCA is vital to multiview learning for several reasons. First, CCA is the earliest multiview subspace learning method and still popular in both the industry and academic community. Second, a bunch of successful multiview learning algorithms is inspired by CCA, especially those subspace learning algorithms. Finally, CCA can be used as a standard tool to handle multiview data without particular knowledge about multiview learning, that is, we can simply apply any single-view learning algorithms to the representations learned by CCA.

The original CCA and most of its extensions are proposed for two multidimensional variables, thus we focus on two views data first. Extensions to more-than-two-view cases will be introduced later.

3.2.1 Canonical Correlation Analysis

Consider a multiview variable $X = (X^1, X^2) \in \mathbb{R}^{D_1} \times \mathbb{R}^{D_2}$ with covariance $\boldsymbol{\Sigma}_{11}$, $\boldsymbol{\Sigma}_{22}$ and cross-variance $\boldsymbol{\Sigma}_{12}$. Subspace learning assumes there exists a latent space where data lie. Let the representations of the variable in the latent space be $Z = (Z^1, Z^2) \in \mathbb{R}^{D_z} \times \mathbb{R}^{D_z}$, where $D_z \leq min(D_1, D_2)$. CCA further assumes that (1) each coordinate of the representations can be obtained by a linear mapping, i.e., $Z^v = \boldsymbol{W}^{v\mathsf{T}} \boldsymbol{X}^v$ and $Z_i^v = \mathbf{w}_i^{v\mathsf{T}} X^v$, with $\mathbf{w}_i^v$ the i-th column of $\mathbf{W}^v$; (2) the representations of each view are uncorrelated, i.e., $\mathbf{w}_i^{1\mathsf{T}} \boldsymbol{\Sigma}_{11} \mathbf{w}_j^1 = 0$ and $\mathbf{w}_i^{2\mathsf{T}} \boldsymbol{\Sigma}_{22} \mathbf{w}_j^2 = 0$; (3) each coordinate of the representations from different views of the same sample in the latent space are maximally correlated,

$$
\begin{aligned}
(\mathbf{w}_i^1, \mathbf{w}_i^2) &= \arg\max_{\mathbf{w}^1, \mathbf{w}^2} corr(Z^1, Z^2) \\
&= \arg\max_{\mathbf{w}^1, \mathbf{w}^2} \frac{\mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}_{12} \mathbf{w}^2}{\sqrt{\mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}_{11} \mathbf{w}^1 \mathbf{w}^{2\mathsf{T}} \boldsymbol{\Sigma}_{22} \mathbf{w}^2}}.
\end{aligned}
\tag{3.1}
$$

Note that (3.1) is invariant to scaling of $\mathbf{w}_1$ and $\mathbf{w}_2$ and can be reformulated as

$$
\begin{aligned}
(\mathbf{w}_i^1, \mathbf{w}_i^2) = \arg\max_{\mathbf{w}^1, \mathbf{w}^2} \ & \mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}_{12} \mathbf{w}^2 \\
s.t. \quad & \begin{cases} \mathbf{w}^{1\mathsf{T}} \boldsymbol{\Sigma}_{11} \mathbf{w}^1 = 1, \\ \mathbf{w}^{2\mathsf{T}} \boldsymbol{\Sigma}_{22} \mathbf{w}^2 = 1. \end{cases}
\end{aligned}
\tag{3.2}
$$

This constrained optimization problem can be solved by using the Lagrangian multiplier method. The solution can be obtained by solving the following generalized eigenvalue problem:

$$
\boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \mathbf{w}_i^1 = \lambda \boldsymbol{\Sigma}_{11} \mathbf{w}_i^1, \tag{3.3}
$$

and

$$
\boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12} \mathbf{w}_i^2 = \lambda \boldsymbol{\Sigma}_{22} \mathbf{w}_i^2. \tag{3.4}
$$

Any eigenvector of (3.3) and (3.4) can be a solution to (3.2), and can be used as (the parameters of) the linear mappings from the original feature space to the coordinate in the latent space. For the sake of retaining the information of the original representation as much as possible, the eigenvectors corresponding to the top d_z eigenvalues are selected. Note that these eigenvectors automatically satisfy the uncorrelated constrains (Borga 2001), i.e., $\mathbf{w}_i^{v\mathsf{T}} \boldsymbol{\Sigma}_{vv} \mathbf{w}_j^v = 0$.

In practice, all covariance matrices $\boldsymbol{\Sigma}_{11}$, $\boldsymbol{\Sigma}_{22}$, and $\boldsymbol{\Sigma}_{12}$ are replaced by their empirical estimates $\hat{\boldsymbol{\Sigma}}_{\mathbf{11}}$, $\hat{\boldsymbol{\Sigma}}_{\mathbf{22}}$ and $\hat{\boldsymbol{\Sigma}}_{\mathbf{12}}$ given a dataset $\mathscr{U} = \{\mathbf{x}_u^v\}_{v=1,2,\ u=1,\ldots,U}$. Some small diagonal matrix should be added to $\hat{\boldsymbol{\Sigma}}_{\mathbf{11}}$, $\hat{\boldsymbol{\Sigma}}_{\mathbf{22}}$ and $\hat{\boldsymbol{\Sigma}}_{\mathbf{12}}$ to avoid singularity.

We can also formulate the objective function of CCA as an optimization problem over $\mathbf{W}^1$ and $\mathbf{W}^2$ directly,

$$
\begin{aligned}
(\boldsymbol{W}^1, \boldsymbol{W}^2) = \arg\max_{\boldsymbol{W}^1, \boldsymbol{W}^2} \ & tr(\boldsymbol{W}^{1\mathsf{T}} \boldsymbol{\Sigma}_{12} \boldsymbol{W}^2) \\
s.t. \quad & \begin{cases} \boldsymbol{W}^{1\mathsf{T}} \boldsymbol{\Sigma}_{11} \boldsymbol{W}^1 = \mathbf{I}, \\ \boldsymbol{W}^{2\mathsf{T}} \boldsymbol{\Sigma}_{22} \boldsymbol{W}^2 = \mathbf{I}. \end{cases}
\end{aligned}
\tag{3.5}
$$

Let $\boldsymbol{U}_{D_z}$ and $\boldsymbol{V}_{D_z}$ be the matrices whose columns are the top D_v left- and right-singular vectors of $\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}$. The solution to (3.5) is

$$(\boldsymbol{W}^1, \boldsymbol{W}^2) = (\boldsymbol{\Sigma}_{11}^{-\frac{1}{2}}\mathbf{U}_{D_z}, \boldsymbol{\Sigma}_{22}^{-\frac{1}{2}}\boldsymbol{V}_{D_z}).$$

3.2.2 Kernel Canonical Correlation Analysis

One drawback of CCA is the assumption that the representations in the latent space are obtained by linear mappings from the original feature space. This linearity assumption restricts the representation power of the latent space. To address this problem, several nonlinear extensions to CCA were proposed. In this subsection, we introduce one of the most widely used ones, kernel CCA (KCCA) (Lai and Fyfe 2000). We will give two derivations of KCCA from different perspectives. Although these derivations are leading to the same solution, viewing KCCA from different perspectives provides more insights into the subspace learning problem. The first one is from the feature space perspective, which is a most straightforward adaption of the kernel method to CCA. The second one is from the function space perspective. Although a bit tricky, it provides a more general way to improve CCA.

To begin with, we first rewrite the objective function (3.2) of CCA. If we assume the data are centralized, we have $\boldsymbol{\Sigma}_{11} = X^1(X^1)^\intercal$, $\boldsymbol{\Sigma}_{22} = X^2(X^2)^\intercal$ and $\Sigma_{12} = X^1(X^2)^\intercal$ in the population setting, and the empirical estimates $\hat{\boldsymbol{\Sigma}}_{\mathbf{11}} = \boldsymbol{X}^1(\boldsymbol{X}^1)^\intercal$, $\hat{\boldsymbol{\Sigma}}_{\mathbf{22}} = \boldsymbol{X}^2(\boldsymbol{X}^2)^\intercal$ and $\hat{\boldsymbol{\Sigma}}_{\mathbf{12}} = \boldsymbol{X}^1(\boldsymbol{X}^2)^\intercal$ with data matrices $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, in practice. The parameters of the linear mappings $\mathbf{w}_i^v$ can be rewritten as $\mathbf{w}_i^v = (\boldsymbol{X}^v)^\intercal\mathbf{m}_i^v$. Then, the objective function of CCA can be reformulated as

$$\begin{aligned}(\mathbf{m}_i^1, \mathbf{m}_i^2) = \underset{\mathbf{m}_i^1, \mathbf{m}_i^2}{\arg\max}\ & \mathbf{m}_i^{1\intercal}\boldsymbol{X}^1\boldsymbol{X}^{1\intercal}\boldsymbol{X}^{\mathbf{2}}\boldsymbol{X}^{2\intercal}\mathbf{m}_i^2 \\ s.t.\ & \begin{cases}\mathbf{m}_i^{1\intercal}\boldsymbol{X}^1(\boldsymbol{X}^1)^\intercal\mathbf{m}_i^1 = 1, \\ \mathbf{m}_i^{2\intercal}\boldsymbol{X}^2(\boldsymbol{X}^2)^\intercal\mathbf{m}_i^2 = 1.\end{cases}\end{aligned} \tag{3.6}$$

Note that (3.6) depends on data only through the inner product between them. Thus we can easily apply the kernel trick, i.e., replacing the inner product with the evaluation of a kernel function. In particular, we replace $\boldsymbol{X}^1\boldsymbol{X}^{1\intercal}$ and $\boldsymbol{X}^2\boldsymbol{X}^{2\intercal}$ with two kernel matrices $\boldsymbol{K}^1$ and $\boldsymbol{K}^2$, whose entity $\boldsymbol{K}_{ij}^v$ is given by the kernel function $k^v(\cdot, \cdot)$ evaluated at the i-th and the j-th samples. We can have different kernel functions for two views.

In practice, regularization should be performed to avoid over-fitting. It is usually adapted from the idea of the norm regularization, i.e., to regularize the mappings with their function norms. In the linear mapping case, the regularization term is simply the squared 2-norm of $\mathbf{w}_i^v$. Note that $||\mathbf{w}_i^v||^2 = (\mathbf{m}_i^v)^\intercal\boldsymbol{X}^v\boldsymbol{X}^{v\intercal}\mathbf{m}_i^v = (\mathbf{m}_i^v)^\intercal\boldsymbol{K}^v\mathbf{m}_i^v$ The objective function (3.6) becomes

$$(\mathbf{m}_i^1, \mathbf{m}_i^2) = \arg\max_{\mathbf{m}_1, \mathbf{m}_2} \mathbf{m}_1{}^\top \boldsymbol{K}^1 \boldsymbol{K}^2 \mathbf{m}_2$$
$$s.t. \begin{cases} \mathbf{m}_1{}^\top \boldsymbol{K}^1 \mathbf{m}^1 + \gamma (\mathbf{m}_i^1)^\top \boldsymbol{K}^1 \mathbf{m}_i^1 = 1, \\ \mathbf{m}_2{}^\top \boldsymbol{K}^2 \mathbf{m}_2 + \gamma (\mathbf{m}_i^2)^\top \boldsymbol{K}^2 \mathbf{m}_i^2 = 1, \end{cases} \tag{3.7}$$

where $\gamma > 0$ is the regularization parameter. The solution to (3.7) can be obtained by calculating the top d_z eigenvectors of

$$(\boldsymbol{K}^1 + \gamma \mathbf{I})^{-1} \boldsymbol{K}^2 (\boldsymbol{K}^2 + \gamma \mathbf{I})^{-1} \boldsymbol{K}^1.$$

The idea behind this adaption of the kernel trick is to use some implicitly defined feature mappings to map the original representations of data to some high-dimensional feature space, whose inner product can be calculated through some kernel functions. Since the algorithm depends on the data only through the inner product, we can perform it without the explicit evaluation of the feature mappings but by directly defining the kernel functions. For more details about the kernel trick, see, e.g., (Hofmann et al. 2008). Basing on the new higher dimensional feature space, we can expect to obtain better representations in the subspace learned by the linear CCA. This derivation of KCCA is from the feature space perspective, i.e., we apply the kernel trick to obtain a new feature space and apply the standard CCA. Although straightforward, it does not provide many insights on how to further improve CCA.

Next, we give another derivation of KCCA from the function space perspective. As we mentioned, one downside of CCA is its restriction on the mapping from the original representations to the latent space representations. Thus, a natural idea to improve CCA is to consider more flexible function classes.

For KCCA, it replaces the linear function class with the RKHS of functions on the original feature space. Namely, we seek a group of functions (f_i^1, f_i^2) from the RKHS $\mathscr{H}_1$ and $\mathscr{H}_2$ with kernel function k^1 and k^2, respectively. Then the objective function can be rewritten as

$$(f_i^1, f_i^2) = \arg\max_{f_1 \in \mathscr{H}_2, f_2 \in \mathscr{H}_2} cov(f_1(X^1), f_2(X^2))$$
$$s.t. \begin{cases} var(f^1(X^1)) = 1, \\ var(f^2(X^2)) = 1. \end{cases} \tag{3.8}$$

Since f_i^1 and f_i^2 are in RKHS, the solutions to (3.8) can be represented as linear combinations of the kernel functions evaluated at data, i.e., $f_i^v(\cdot) = \sum_{j=1}^{|U|} k^v(\cdot, \mathbf{x}_\mathbf{j}^\mathbf{v})$. With the usual notation for the kernel matrices and proper regularization, the objective function (3.8) can be rewritten to the same as (3.7) and we can obtain the same solutions. This derivation is from the function space perspective, i.e., we perform CCA on the original feature space with maps from more flexible function spaces. Since more flexible function classes can be used, this perspective suggests a general way to improve CCA.

3.2.3 Probabilistic Canonical Correlation Analysis

Many popular machine learning algorithms have their corresponding probabilistic interpretations, and there is also no exception to CCA. Specifically, one can reformulate CCA in terms of the probabilistic graphical models. Having such a probabilistic interpretation is of multiple merits. For example, established probably theory provides tools for a deeper understanding of the algorithms and theoretical analysis of them. One can easily replace the components of the graphical model with other probabilistic distributions to better model the data at hand. The graphical model can also be incorporated into a larger hierarchical model, to handle more complex problems. In this section, we introduce a probabilistic interpretation of CCA (Bach and Jordan 2005). In particular, we show there exists a probabilistic model, of which the maximum likelihood estimates of the parameters are exactly the same as the solution to CCA.

The probabilistic model we consider is a generative model, i.e., it models the data generating process. To begin with, recall the assumption that there is a latent space underlying the observed data. Thus, we consider the following data generating process. First, we generate a latent variable Z taking values in the latent space $\mathbb{Z}$. Then the two views X^1 and X^2 of the observable variable are generated independently conditioning on Z. We assume the latent variable Z follows a d_Z-dimensional Gaussian prior $N(0, \mathbf{I})$. Since the map to different coordinates of the representation learned by CCA is forced to be uncorrelated, the diagonal prior covariance assumption is not limited. Given the latent variable Z, the observed variables, X^1 and X^2, are assumed to be linear functions of it, with independent Gaussian noises, namely, $X^1 \sim N(\boldsymbol{W}^1 Z + \boldsymbol{\mu}^1, \boldsymbol{\Psi}^1)$ and $X^2 \sim N(\boldsymbol{W}^2 Z + \boldsymbol{\mu}^2, \boldsymbol{\Psi}^2)$. Here $\boldsymbol{W}^v$, $\boldsymbol{\mu}^v$ and $\boldsymbol{\Psi}^v$ are model parameters to be optimized.

Formally, the model we consider can be formulated as

$$\begin{aligned} Z &\sim N(0, \mathbf{I}) \\ X^1|Z &\sim N(\boldsymbol{W}^1 Z + \boldsymbol{\mu}^1, \boldsymbol{\Psi}^1) \\ X^2|Z &\sim N(\boldsymbol{W}^2 Z + \boldsymbol{\mu}^2, \boldsymbol{\Psi}^2). \end{aligned}$$

The maximum likelihood estimates of parameters $\mathbf{W}_v$, $\boldsymbol{\mu}_v$ and Ψ_v are given by

$$\begin{aligned} \hat{\boldsymbol{W}}^1 &= \tilde{\boldsymbol{\Sigma}}^{11} \boldsymbol{U}^1 \boldsymbol{M}^1 \\ \hat{\boldsymbol{W}}^2 &= \tilde{\boldsymbol{\Sigma}}^{22} \boldsymbol{U}^2 \boldsymbol{M}^2 \\ \hat{\boldsymbol{\Psi}}^1 &= \tilde{\boldsymbol{\Sigma}}^{11} - \hat{\boldsymbol{W}}^{1\mathsf{T}} \hat{\boldsymbol{W}}^1 \\ \hat{\boldsymbol{\Psi}}^2 &= \tilde{\boldsymbol{\Sigma}}^{22} - \hat{\boldsymbol{W}}^{2\mathsf{T}} \hat{\boldsymbol{W}}^2 \\ \hat{\mu}^1 &= \tilde{\mu}^1 \\ \hat{\mu}^2 &= \tilde{\mu}^2, \end{aligned}$$

where $\tilde{\boldsymbol{\mu}}^v$ and $\tilde{\boldsymbol{\Sigma}}^{vv}$ are the sample estimates of data mean and covariance, respectively, $\boldsymbol{U}^v \in \mathbb{R}^{D_v \times D_z}$ whose columns are the first D_z canonical directions, i.e., the solution

to CCA (3.6), and $\boldsymbol{M}^1, \boldsymbol{M}^2 \in \mathbb{R}^{D_z \times D_z}$ are arbitrary matrices whose spectral norms are less than one and such that $\boldsymbol{M}^1 \boldsymbol{M}^{2\mathsf{T}} = \boldsymbol{P}$ with $\boldsymbol{P}$ the diagonal matrix having the first D_Z canonical correlations as diagonal elements.

The posterior mean of Z given X^1 and X^2 are

$$E[Z|X^1] = \boldsymbol{M}^{1\mathsf{T}} \boldsymbol{U}^{1\mathsf{T}} (X^1 - \boldsymbol{\mu}^1)$$
$$E[Z|X^2] = \boldsymbol{M}^{2\mathsf{T}} \boldsymbol{U}^{2\mathsf{T}} (X^2 - \mu^2).$$

By taking the posterior expectation of Z given X^1 and X^2, we project the observed variables to the same subspaces obtained by CCA. In other words, we recover the solutions of CCA by calculating the maximum likelihood estimation of the parameters of a probabilistic model. Thus, the model can be taken as a probabilistic interpretation of CCA.

3.2.4 *Bayesian Canonical Correlation Analysis*

So far, we have given a probabilistic interpretation of CCA, which is a probabilistic model whose maximum likelihood estimations of parameters are equivalent to the solution of CCA. However, it is that known MLE suffers over-fitting severely, and one way to alleviate this is the Bayesian methods. Rather than simply applying the Bayesian methods to the probabilistic CCA (PCCA) model, there indeed exist several different Bayesian treatments of CCA with their extensions. We will not exhaustively introduce all these methods, but focus on a particular one of great practical interest, the inter-battery factor analysis (IBFA) (Klami et al. 2013).

PCCA assumes there is a latent variable $Z \in \mathbb{R}^{D_z}$ determines the generation of the observed variable $X^1 \in \mathbb{R}^{D_1}$ and $X^{\in} \mathbb{R}^{D_2}$ from both views. IBFA further assumes the existence of two more latent variables $Z^1, Z^2 \in \mathbb{R}^{D_z}$, one for a view. The latent variable Z captures the variations common to both view, while Z^1 and Z^2 model each view's own variations. Thus, IBFA has more explanatory power than the PCCA model we introduced.

Formally, IBFA is formulated as

$$\begin{aligned} Z &\sim N(0, \boldsymbol{I}), \\ Z^1 &\sim N(0, \boldsymbol{I}), \\ Z^2 &\sim N(0, \boldsymbol{I}), \\ X^1|Z &\sim N(\boldsymbol{A}^1 Z + \boldsymbol{B}^1 Z^1, \boldsymbol{\Psi}^1), \\ X^2|Z &\sim N(\boldsymbol{A}^2 Z + \boldsymbol{B}^2 Z^2, \boldsymbol{\Psi}^2). \end{aligned}$$

Here, $\boldsymbol{A}^v, \boldsymbol{B}^v \in R^{D_v \times D_z}$ and $\boldsymbol{\Psi}^v \in R^{D_v \times D_v}$ are also latent variables since we consider Bayesian treatment of CCA here. The specification of the corresponding priors will be given later. Like PCCA, the observed variables $\boldsymbol{X}^v$ are assumed be linear functions

of latent effects with additive Gaussian noise $N(\mathbf{0}, \boldsymbol{\Psi}^v)$. Usually, diagonal noise in particular isotropic noise is assumed.

By integrating out the view-specific variations Z^1 and Z^2, we can easily recover the PCCA model introduced in the last section

$$\begin{aligned} Z &\sim N(0, \mathbf{I}), \\ X^1|Z &\sim N(\boldsymbol{A}^1 Z, \boldsymbol{B}^1(\boldsymbol{B}^1)^\intercal \boldsymbol{\Psi}^1), \\ X^2|Z &\sim N(\boldsymbol{A}^2 Z, \boldsymbol{B}^2(\boldsymbol{B}^2)^\intercal \boldsymbol{\Psi}^2), \end{aligned}$$

with a reparameterization of the covariance matrices. In particular, the view-specific variations are implicitly modeled by the noise of each view.

Now, we come to the Bayesian inference of IBFA. Although we can equip the latent variables with proper priors and do inference directly over the proposed model, it suffers from the problem of undefinability. In particular, the model has three sets of latent variables but the interchange of components within these sets does not affect the likelihood. Hence, careful model selections should be addressed for every solution to IBFA. One solution to this problem is to replace the three sets with one single set, and impose further constraints on the structure of the solution.

To begin with, we reformulate the model. Let $X = [X^1; X^2]$, $Y = [Z; Z^1; Z^2]$,

$$\mathbf{W} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{0} & \mathbf{B}_2 \end{bmatrix}, \tag{3.9}$$

and

$$\Psi = \begin{bmatrix} \Psi_1 & \mathbf{0} \\ \mathbf{0} & \Psi_2 \end{bmatrix}, \tag{3.10}$$

we have

$$\begin{aligned} Y &\sim N(0, \mathbf{I}), \\ X|Y &\sim N(\mathbf{W}Y.\Psi). \end{aligned}$$

This model is equivalent to the IBFA model. The special structure of $\mathbf{W}$ and Ψ forces the model to capture the view-specific variations. By the introduction of proper prior to $\mathbf{W}$, the model can accomplish automate component allocation within the latent variable sets.

The prior used is called group-wise automatic relevance determination (ARD), which is defined as

$$p(\mathbf{W}) = \prod_{v=1}^{2} ARD(W^v|\alpha_0, \beta_0), \tag{3.11}$$

where $\mathbf{W}^1$ is a matrix composed of the first d_1 rows of $\mathbf{W}$ while $\mathbf{W}^2$ composed of the rest rows. The ARD prior is defined as

$$ARD(W^v|\alpha_0, \beta_0) = \prod_{i=1}^{d_v} N(\mathbf{w}_i^v|\mathbf{0}, (\mathbf{a}_i^v)^{-1}\mathbf{I})Gamma(\mathbf{a}_i^v|\alpha_0, \beta_0).$$

For simplicity, we assume isotopic noise here, i.e., $\Psi_v = \eta_v^{-1}\mathbf{I}$, with Gamma prior for the precision variables..

We want to calculate the posterior distribution $p(\mathbf{W}, \eta_v, \mathbf{a}^v, \mathbf{Y}|\mathbf{X})$, which can be approximated by variational inference with the following factorized variational distribution:

$$q(\mathbf{W}, \eta_v, \mathbf{a}^v, \mathbf{Y}) = \prod_{n=1}^{N} q(\mathbf{y}_n) \prod_{v=1}^{2} (q(\eta_v)q(\mathbf{a}_v) \prod_{i=1}^{d_1+d_2} q(\mathbf{w}_i),$$

with $\mathbf{w}_i$ the i-th row of $\mathbf{W}$. In particular, we maximize the following evidence lower bound with respect to each of the factor $q(\cdot)$ alternatively

$$L(q) = \int q(\mathbf{W}, \eta_v, \mathbf{a}^v, \mathbf{Y}) \log \frac{p(\mathbf{W}, \eta_v, \mathbf{a}^v, \mathbf{Y}, \mathbf{X})}{q(\mathbf{W}, \eta_v, \mathbf{a}^v, \mathbf{Y})}.$$

Since all the priors used are conjugate, we do not need to specify the form of the factors, and they can be automatically determined.

With the obtained posterior, we can easily calculate the desired projections as well as the latent representations.

3.3 Multiview Subspace Learning with Supervision

In the last section, we introduced several CCA-based multiview subspace learning methods. Since all these methods are unsupervised, they can be applied to general multiview learning problems. However, when there exists available label information, although a sequential application of one of these methods and a single'-view supervised learning algorithm is reasonable, better performance can be expected if we operate directly on a subspace learned with label information. Thus, in this section, we will introduce to supervised multiview subspace learning methods. Note that we only focus on the supervised methods with an explicitly learned subspace. For a more general introduction to multiview supervised learning, see Chap. 4.

3.3.1 Multiview Linear Discriminant Analysis

A general procedure to develop new multiview learning algorithms is to start with a successful single-view algorithm and extend it to multiview settings. This procedure applies to subspace learning as well. As a simple but effective supervised single-view subspace learning algorithm, linear discriminant analysis (LDA) a.k.a. Fisher

discriminant analysis (FDA) is suitable as the starting point of our first supervised multiview subspace learning algorithm.

The basic idea of LDA is to project the data onto a low-dimensional subspace, in which the between-class distance of the data is maximized while the within-class distance is minimized. Given a labeled dataset $L = \{\mathbf{x}_{nt}\}$, $t \in [T]$, where $\mathbf{x}_{nt} \in R^d$ is the l-th sample from class t. Define three scatter matrices, that is, the within-class scatter matrix

$$S_W = \frac{1}{N} \sum_{t=1}^{T} \sum_{n=1}^{N_t} (\mathbf{x}_{nt} - \mathbf{m}_t)(\mathbf{x}_{nt} - \mathbf{m}_t)^\intercal,$$

the between-class scatter matrix

$$S_B = \frac{1}{N} \sum_{t=1}^{T} N_t (\mathbf{m}_t - \mathbf{m})(\mathbf{m}_t - \mathbf{m})^\intercal,$$

and the total scatter matrix

$$S_T = \frac{1}{N} \sum_{t=1}^{T} \sum_{n=1}^{N_t} (\mathbf{x}_{nt} - \mathbf{m})(\mathbf{x}_{nt} - \mathbf{m})^\intercal,$$

where N_t is the number of samples from class t, $N = \sum_{t=1}^{T} N_t$, $\mathbf{m}_t$ is the mean of samples from class t, and $\mathbf{m}$ is the mean of all the samples. LDA seeks a linear projection $\mathbf{W} \in R^{d_z \times d}$ from the observed space R^d to the low-dimensional subspace R^{d_z}. Each dimension of the subspace representation should maximize the between-class distance of the data while minimize the within-class distance. Thus, the linear mapping parameterized by $\mathbf{w}$ from the original feature space to any dimension of the subspace should satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\intercal S_B \mathbf{w}}{\mathbf{w}^\intercal S_W \mathbf{w}}, \tag{3.12}$$

or alternatively

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\intercal S_B \mathbf{w}}{\mathbf{w}^\intercal S_T \mathbf{w}}. \tag{3.13}$$

The solution to (3.12) (3.13) is the eigenvector corresponding to the largest eigenvalue of $S_W^{-1} S_B$ ($S_T^{-1} S_B$). Since a d_z-dimensional subspace is desired, the eigenvectors corresponding to the largest d_Z eigenvalues are used to formulate the projection $\mathbf{W}$.

In the multiview case, consider a multiview dataset $L = \{\mathbf{x}_{tvn}\}_{t \in [T], v \in [2], n \in [N_{tv}]}$, where $\mathbf{x}_{tvn} \in R^{d_v}$ is the features from the v-th view of the n-th sample form the t-th class, N_{tv} is the number of samples from the v-th view of the t-th class. We need two linear projections $\mathbf{W}_1 \in R^{d_z \times d_1}$, $\mathbf{W}_2 \in R^{d_z \times d_2}$ to project two views onto a d_z-

dimensional subspace. Besides discriminative information preserved by LDA, the subspace is desired to encode one's beliefs on the multiview nature of the data. One way to obtain such projections is to combine LDA and CCA. In particular, multiview linear discriminant analysis (MLDA) (Yang and Sun 2014) optimizes the following objective function:

$$\arg\max_{\mathbf{w}_1,\mathbf{w}_2} \frac{\mathbf{w}_1^\intercal S_{B_1}\mathbf{w}_1}{\mathbf{w}_1^\intercal S_{T_1}\mathbf{w}_1} + \frac{\mathbf{w}_2^\intercal S_{B_2}\mathbf{w}_2}{\mathbf{w}_2^\intercal S_{T_2}\mathbf{w}_2} + 2\gamma \frac{\mathbf{w}_1^\intercal C_{12}\mathbf{w}_2}{\sqrt{(\mathbf{w}_1^\intercal C_{11}\mathbf{w}_1)(\mathbf{w}_2^\intercal C_{22}\mathbf{w}_2)}}, \tag{3.14}$$

where γ is a trade-off parameter balancing the relative significance of LDA and CCA, S_{B_v} and S_{T_v} are the between-class scatter matrix and the total scatter matrix of data from view v, respectively, and C_{uv} is the covariance matrix defined as

$$C_{uv} = \frac{1}{N}\sum_{t=1}^{T}\sum_{n=1}^{N_t}(\mathbf{x}_{tun} - \mathbf{m}_u)(\mathbf{x}_{tvn} - \mathbf{m}_v)^\intercal,$$

where $\mathbf{m}_v$ is the mean of all samples form view v.

Since (3.14) is invariance to rescaling of $\mathbf{m}_v$, it can be reformulated as

$$\begin{aligned} &\arg\max_{\mathbf{w}_1,\mathbf{w}_2} \mathbf{w}_1^\intercal S_{B_1}\mathbf{w}_1 + \mathbf{w}_2^\intercal S_{B_2}\mathbf{w}_2 + 2\gamma \mathbf{w}_1{}^\intercal C_{12}\mathbf{w}_2 \\ &\qquad s.t. \begin{cases} \mathbf{w}_1{}^\intercal C_{11}\mathbf{w}_1 = 1, \\ \mathbf{w}_2{}^\intercal C_{22}\mathbf{w}_2 = 1. \end{cases} \end{aligned} \tag{3.15}$$

Using the Lagrangian multiplier method, (3.15) can be transformed to a generalized multi-variable eigenvalue problem,

$$\begin{bmatrix} S_{B_1} & \gamma C_{12} \\ \gamma C_{12} & S_{B_2} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} = \begin{bmatrix} S_{T_1} & \\ & S_{T_2} \end{bmatrix} \begin{bmatrix} \lambda_1\mathbf{w}_1 \\ \lambda_2\mathbf{w}_2 \end{bmatrix} \tag{3.16}$$

which can be solved by iteratively finding the eigenvalue–eigenvector pairs (Rupnik 2010; Horst 1961).

MLDA is a natural combination of LDA and CCA, which seeks a subspace preserving both the within-view discriminative information and the cross-view correlation. It benefits from the nature of both LDA and CCA. As classical subspace learning algorithms, LDA and CCA both exhibit many extensions, which can be easily incorporated into MLDA. In the next section, we will give such an example.

3.3.2 Multiview Uncorrelated Linear Discriminant Analysis

Although LDA can learn a discriminative subspace, there are restrictions on the relationship between learned features. In particular, LDA may learn statistical related

features and thus a redundant subspace. Since uncorrelated features with minimum redundancy are desired in many applications, uncorrelated linear discriminant analysis (ULDA) (Jin et al. 2001) was proposed to alleviate this problem.

Suppose a d_z-dimensional subspace is desired. ULDA seeks d_z linear maps $\mathbf{w}_1, \ldots, \mathbf{w}_{d_z}$. Besides maximizing the criterion for LDA (3.12), these vectors should also be S_T orthogonal, that is, $\mathbf{w}_i^\intercal S_T \mathbf{w}_j = 0, i \neq j$. Using the definition of S_T, it is easy to show the vectors $\mathbf{w}_i$ are statistically uncorrelated.

The solution to ULDA can be obtained by successively solving a sequence of generalized eigenvalue problems. Specifically, the i-th linear map $\mathbf{w}_i$ is the eigenvector corresponding to the maximum eigenvalue of the following generalized eigenvalue problem:

$$P_i S_B \mathbf{w}_i = \lambda_i S_W \mathbf{w}_i$$

where

$$\begin{aligned} P_1 &= I \\ P_i &= I - S_T D_i^\intercal \left(D_i S_T S_W^{-1} S_T D_i^\intercal \right)^{-1} D_i S_T S_W^{-1}, \ i > 1, \\ D_i &= [\mathbf{w}_1, \ldots, w_{i-1}]^\intercal, \ i > 1. \end{aligned}$$

As an extension of LDA to multiview setting, MLDA may also learn a redundant subspace. To alleviate this problem, we can incorporate the S_T orthogonal restrictions to MLDA, leading to a new multiview learning algorithm, multiview uncorrelated linear discriminant analysis (MULDA) (Yang and Sun 2014). MULDA seeks d_z pairs of linear maps $(\mathbf{w}_{11}, \mathbf{w}_{21}), \ldots, (\mathbf{w}_{1d_z}, \mathbf{w}_{21d_z})$, which maximize (3.14) and satisfy

$$\mathbf{w}_{1i} \mathtt{S}_{\mathtt{T}_1} \mathbf{w}_{1\mathtt{j}} = \mathbf{w}_{2\mathtt{i}} \mathtt{S}_{\mathtt{T}_2} \mathbf{w}_{2\mathtt{j}} = 0, \mathtt{i} \neq \mathtt{j}.$$

The objective function of MULDA can be formulated as

$$\begin{aligned} &\arg\max_{\mathbf{w}_{1i}, \mathbf{w}_{2i}} \mathbf{w}_{1i}^\intercal S_{B_1} \mathbf{w}_{1i} + \mathbf{w}_{2i}^\intercal S_{B_2} \mathbf{w}_{2i} + 2\gamma \mathbf{w}_{1i}{}^\intercal C_{12} \mathbf{w}_{2i} \\ s.t. &\begin{cases} \mathbf{w}_{1i}^\intercal S_{T_1} \mathbf{w}_{1i} + \sigma \mathbf{w}_{2i}^\intercal S_{T_2} \mathbf{w}_{21i} = 1, \\ \mathbf{w}_{1i} \mathtt{S}_{\mathtt{T}_1} \mathbf{w}_{1\mathtt{j}} = \mathbf{w}_{2\mathtt{i}} \mathtt{S}_{\mathtt{T}_2} \mathbf{w}_{2\mathtt{j}} = 0, \mathtt{j} = 1, \ldots, \mathtt{i} - 1. \end{cases} \end{aligned} \tag{3.17}$$

Like ULDA, the solution to MULDA can also be obtained by successively solving a sequence of generalized eigenvalue problems, where the i-th linear map is the solution to

$$\begin{bmatrix} P_1 & \\ & P_2 \end{bmatrix} \begin{bmatrix} S_{B_1} & \gamma C_{12} \\ \gamma C_{12} & S_{B_2} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1i} \\ \mathbf{w}_{2i} \end{bmatrix} = \lambda \begin{bmatrix} S_{T_1} & \\ & S_{T_2} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1i} \\ \mathbf{w}_{2i} \end{bmatrix}, \tag{3.18}$$

where

$$
\begin{aligned}
P_1 &= I - S_{T_1} D_i^{\mathsf{T}} \left(D_1 S_{T_1} D_1^{\mathsf{T}}\right)^{-1} D_1, \\
P_2 &= I - S_{T_2} D_i^{\mathsf{T}} \left(D_2 S_{T_2} D_2^{\mathsf{T}}\right)^{-1} D_2, \\
D_1 &= [\mathbf{w}_{11}, \ldots, w_{1(i-1)}]^{\mathsf{T}}, \\
D_2 &= [\mathbf{w}_{21}, \ldots, w_{2(i-1)}]^{\mathsf{T}}.
\end{aligned}
$$

Note that we omit the dependence of the notions on i for simplicity.

After obtaining d_z pairs of linear maps $(\mathbf{w}_{11}, \mathbf{w}_{21}), \ldots, (\mathbf{w}_{1d_z}, \mathbf{w}_{21d_z})$, two strategies can be used to calculate the lower dimensional representations in the subspace,

$$
Z = \begin{bmatrix} W_1 & \\ & W_2 \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix},
$$

or

$$
Z = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix},
$$

where $W_1 = [\mathbf{w}_{11}, \ldots, w_{1d_z}]$ and $W_2 = [\mathbf{w}_{21}, \ldots, w_{2d_z}]$.

MULDA is a combination of ULDA and CCA and an extension to MLDA. Like MLDA, it considers both intra-view class structure and inter-view correction. Additionally, the feature vectors extracted are mutually uncorrelated in the subspace, thus removing the redundancy in the original features while achieving maximum correlation between different views and discrimination in each view.

3.3.3 Hierarchical Multiview Fisher Discriminant Analysis

So far, we have illustrated how to construct supervised multiview subspace learning methods by combining supervised single-view subspace learning with unsupervised multiview subspace learning. However, if we focus on the supervised learning task more than the learned subspace, it is more natural to combine supervised single-view subspace learning with supervised multiview learning. In this section, we will introduce such a method called multiview Fisher discriminant analysis (MFDA) (Chen and Sun 2009), which is again a multiview extension of LDA (FDA) but combined with co-regularization rather than CCA.

MFDA seeks a pair of linear discriminant functions (f_1, f_2), parameterized by two projections $\mathbf{w}_1$, $\mathbf{w}_2$ and two biases b_1, b_2, where $f_1(\cdot) = \mathbf{w}_1^{\mathsf{T}} \cdot + b_1$ and $f_2(\cdot) = \mathbf{w}_2^{\mathsf{T}} \cdot + b_2$, such that (f_1, f_2) are solutions to a co-regularization problem with the projections satisfying constraints induced from FDA. The objective function of MFDA is

$$\min_{\mathbf{w}_1,\mathbf{w}_2,b_1,b_2} \sum_i \left[(y_i - f_{1i})^2 + \alpha(y_i - f_{2i})^2\right] + \beta \sum_i (f_{1i} - f_{2i})^2$$
$$s.t. \begin{cases} k\mathbf{w}_1^\intercal S_{W_1}\mathbf{w}_1 + \mathbf{w}_1^\intercal S_{B_1}\mathbf{w}_1 \leq 0, \\ k\mathbf{w}_2^\intercal S_{W_2}\mathbf{w}_2 + \mathbf{w}_2^\intercal S_{B_2}\mathbf{w}_2 \leq 0, \end{cases} \tag{3.19}$$

where f_{vi} is the output of the discriminant function of the i-th input from view v. It is easy to see that the objective function of MFDA is indeed that of co-regularization with mean squared error loss, with additional constraints which are simple reformulations of the objective function of LDA.

One drawback of MFDA is that it can only handle binary classification problems. One simple strategy to apply MFDA to multi-class cases is using the one-vs-rest scheme. It is usually difficult to find a one-dimensional subspace separating one class from others directly. To address this problem, (Chen and Sun 2009) proposed hierarchical MFDA (HMFDA) which combines hierarchical metric learning (Sun and Chen 2011) with MFDA.

HMFDA consists of four steps. First, two clusters are constructed according to some cluster criterion. Then MFDA is used to learn optimal transforms that separate these two clusters. After that, for any cluster including more than one class, the first two steps are applied to it again. Finally, test points are classified hierarchically.

3.4 Other Methods

Tensor Canonical Correlation Analysis (Luo et al. 2015): The original CCA and most of its extensions are restricted to two-view data. A naive extension to more-than-two-view settings is to consider the view-wise covariance of the data and then maximize the summation of the correlation coefficients. A more principled way is to utilize the covariance tensor of the multiview data, which captures the relationship between all the views simultaneously. Tensor CCA (TCCA) is such a multiview subspace learning method, which can handle an arbitrary number of views. TCCA seeks a subspace by maximizing the canonical correlation of the views. The maximization is equivalent to finding the best rank-1 approximation of the covariance tensor of the data, which can be efficiently solved by the alternating least squares algorithm.

Multiview Intact Space Learning (Xu et al. 2015): A common assumption of multiview learning is that each view of the multiview data is sufficient for learning, which may not be the case in practice. Multiview intact space learning (MISL) works under the assumption that the views can be insufficient. By utilizing the complementary information from the views, MISL seeks a latent intact representation of the data. MISL assumes that each view is generated by a linear transformation of a common latent space representation. The representation as well as the transformations are obtained by minimizing the difference between the transformed representation and the observations. The difference is measured by Cauchy loss, which makes the algorithm more robust to outliers.

Multiview Regularized Fisher Discriminant Analysis (Diethe et al. 2008): Multiview regularized Fisher discriminant analysis (MrFDA) is a multiview extension of regularized kernel Fisher discriminant analysis (rKFDA). This extension effectively utilizes the fact that the discriminant functions from different views should predict the same label to an example and the similarity between the formulations of the term penalizing the examples' within-class distance in rKFDA and the term ensures the projection vectors of the same view be uncorrelated in KCCA. The objective function of MFDA is convex thus can be efficiently optimized. As a combination of two kernel methods, i.e., rKFDA and KCCA, MFDA can also be kernelized easily.

References

Bach FR, Jordan MI (2005) A probabilistic interpretation of canonical correlation analysis, pp 1–11. http://statistics.berkeley.edu/sites/default/files/tech-reports/688

Borga M (2001) Canonical correlation: a tutorial, pp 1–12. http://people.imt.liu.se/magnus/cca

Chen Q, Sun S (2009) Hierarchical multi-view fisher discriminant analysis. In: Proceedings of the 16th international conference on neural information processing: Part II, pp 289–298

Diethe T, Hardoon DR, Shawe-Taylor J (2008) Multiview fisher discriminant analysis. In: NIPS workshop on learning from multiple sources

Hardoon DR, Szedmak SR, Shawe-taylor JR (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Comput 16(12):2639–2664

Hotelling H (1933) Analysis of a complex of statistical variables into principal components. J Educ Psychol 24(6):498–520

Horst P (1961) Relations amongm sets of measures. Psychometrika 26(2):129–149

Hofmann T, Schölkopf B, Smola AJ (2008) Kernel methods in machine learning. Ann Stat 1171–1220

Jin Z, Yang JY, Hu ZS, Lou Z (2001) Face recognition based on the uncorrelated discriminant transformation. Pattern Recognit 34(7):1405–1416

Klami A, Virtanen S, Kaski S (2013) Bayesian canonical correlation analysis. J Mach Learn Res 14(1):965–1003

Lai PL, Fyfe C (2000) Kernel and nonlinear canonical correlation analysis. Int J Neural Syst 10(5):365–377

Luo Y, Tao D, Ramamohanarao K, Xu C, Wen Y (2015) Tensor canonical correlation analysis for multi-view dimension reduction. IEEE Trans Knowl Data Eng 27(11):3111–3124

Rupnik J (2010) Multi-view canonical correlation analysis. Technical report

Sun S, Chen Q (2011) Hierarchical distance metric learning for large margin nearest neighborhood classification. Int J Pattern Recognit Artif Intell 25(7):1073–1087

Xu C, Tao D, Xu C (2015) Multi-view intact space learning. IEEE Trans Pattern Anal Mach Intell 37(12):2531–2544

Yang M, Sun S (2014) Multi-view uncorrelated linear discriminant analysis with applications to handwritten digit recognition. In: Proceedings of 2014 international joint conference on neural networks, pp 4175–4181

Chapter 4
Multiview Supervised Learning

Abstract Multiview supervised learning algorithm can exploit the multiview nature of the data by the consensus of the views, that is, to seek predictors from different views that agree on the same example. In this chapter, we introduce three categories of multiview supervised learning methods. The first one contains the multiview large margin-based classifiers, which regularize the classifiers from different views with their agreements on classification margins to enforce view consensus. The second one contains multiple kernel learning, where the feature mappings underlying multiple kernels will map the views to new feature spaces where the classifiers are more likely to agree on the views. The third one contains some Gaussian process related models, in which case the predict functions themselves are taken as random variables. We also briefly introduce some other methods at the end of this chapter.

4.1 Introduction

Supervised learning is one of the most classical learning scenarios. For multiview supervised learning, multiple distinct feature sets (views) share a common target variable. A multiview supervised learning algorithm can exploit the multiview nature of the data by the consensus of the views, that is, to seek predictors from different views that agree on the same example.

In this chapter, we will introduce several multiview supervised learning algorithms, which model the multiview data and encode their beliefs on view consensus through different ways. We begin with several large margin-based classifiers, which regularize the classifiers with their agreements on classification margins to enforce view consensus. After that, we consider multiple kernel learning, where the feature mappings underlying multiple kernels will map the views to new feature spaces where the classifiers are more likely to agree on the views. Finally, we introduce some Gaussian process related models, in which case the predict functions themselves are taken as random variables. Thus the view consensus can be measured through some similarity measure between the distributions of the predict functions.

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_4

4.2 Multiview Large Margin Classifiers

The large margin principle is a widely used principle to construct classifiers, among which SVM is one of the most famous algorithms. In this section, we will show how the classification margins can be a natural quantity to encode our beliefs on view consensus. We will begin our introduction with SVM-2K which is one of the earliest algorithms that exploits the large margin principle in multiview settings. After that, we will introduce multiview maximum entropy discrimination (MVMED), which is a successful combination of the large margin principle and probabilistic models.

4.2.1 SVM-2K

As a classical classification algorithm, SVM has many different multiview extensions (Sun et al. 2017), and some of which have already been introduced before. However, the one we introduce here, SVM-2K (Farquhar et al. 2006), is the earliest one of these algorithms, and the first one that exploitees the large margin principle in multiview settings. Thus it is worth devoting a subsection to SVM-2K, to open a door to multiview supervised learning.

Recall that given a labeled training set $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$, $y_l \in \{-1, 1\}$, SVM seeks a discriminant function $f(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + b$ which classifies all the training samples correctly at a margin. SVM can be formulated as the following optimization problem for the linearly separable case

$$\begin{aligned} &\arg\min_{\mathbf{w}} && \frac{1}{2}||\mathbf{w}||^2 \\ &s.t. && y_l(\mathbf{w}^\intercal \mathbf{x}_l + b) \geq 1,\ l = 1, \ldots, L. \end{aligned} \tag{4.1}$$

In the case where the training set is not linearly separable, one may map the input variable to a high-dimensional feature space using some feature mapping $\phi(\mathbf{x})$, and introduce a set of slack variables $\{\xi_l\}_{l=1,\ldots,L}$. Note that we never explicitly need the feature map $\phi(\mathbf{x})$ but instead a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >$, since the solution to SVM only depends on the input variable through the inner product between them. In this case, the optimization problem of SVM (4.1) becomes

$$\begin{aligned} &\arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + c\sum_{l=1}^{L}\xi_l \\ &s.t. \begin{cases} y_l(\mathbf{w}^\intercal \phi(\mathbf{x}_l) + b) \geq 1 - \xi_l, \\ \xi_l \geq 0,\ l = 1, \ldots, L, \end{cases} \end{aligned} \tag{4.2}$$

where $c > 0$ is some trade-off parameter.

In multiview cases, two discriminant functions $f^1(\mathbf{x}^1) = (\mathbf{w}^1)^\intercal \mathbf{x}^1 + b^1$ and $f^2(\mathbf{x}^2) = (\mathbf{w}^1)^\intercal \mathbf{x}^2 + b^2$ should be learned, one for each view. Besides the classification margin constraints of SVM, additional constraints are needed to impose the view consensus on these discriminant functions. SVM-2K introduces a constraint of similarity between the outputs from different views of the same input, which is measured by an ε-insensitive 1-norm with slack variables,

$$\left|(\mathbf{w}^1)^\intercal \phi(\mathbf{x}_l^1) + b^1 - (\mathbf{w}^2)^\intercal \phi(\mathbf{x}_l^2) - b^2\right| \geq \eta_l + \varepsilon.$$

Combining this with the objective function of SVM, we reach the following optimization problem for SVM-2K:

$$\begin{aligned} \underset{\mathbf{w}^1, \mathbf{w}^2, b^1, b^2}{\arg\min} \quad & \frac{1}{2}\left\|\mathbf{w}^1\right\|_2^2 + \frac{1}{2}\|\mathbf{w}\|_2^2 + c_1 \sum_{l=1}^{L} \xi_l^1 + c_2 \sum_{l=1}^{L} \xi_l^2 + d \sum_{l=1}^{L} \eta_l \\ s.t. \quad & \begin{cases} \left|(\mathbf{w}^1)^\intercal \phi^1(\mathbf{x}_l^1) + b^1 - (\mathbf{w}^2)^\intercal \phi^2(\mathbf{x}_l^2) - b_2\right| \leq \eta_l + \varepsilon \\ y_l \left((\mathbf{w}^1)^\intercal \phi^1(\mathbf{x}_l^1) + b^1\right) \geq 1 - \xi_l^1, \\ y_l \left((\mathbf{w}^2)^\intercal \phi^2(\mathbf{x}_l^2) + b^2\right) \geq 1 - \xi_l^2, \\ \xi_l^1 \geq 0, \xi_l^2 \geq 0, \eta_l \geq 0 \; l \in [L]. \end{cases} \end{aligned}$$

As usual, this problem can be solved by using the Lagrangian multiplier method, leading to the following dual problem

$$\begin{aligned} \underset{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2}{\arg\max} \quad & \mathbf{1}^\intercal \boldsymbol{\lambda}^1 + \mathbf{1}^\intercal \boldsymbol{\lambda}^2 - \frac{1}{2} \sum_{i,j=1}^{|L|} \left(g_i^1 g_j^1 \boldsymbol{K}_{ij}^1 + g_i^2 g_j^2 \boldsymbol{K}_{ij}^2\right) \\ s.t. \quad & \begin{cases} g_i^1 = \lambda_i^1 y_i - \beta^+ + \beta^-, \\ g_i^2 = \lambda_i^2 y_i + \beta^+ - \beta^-, \\ \sum_{i=1}^{L} g_i^1 = \sum_{i=1}^{L} g_i^2 = 0, \\ 0 \leq \boldsymbol{\lambda}^1 \leq c^1, 0 \leq \boldsymbol{\lambda}^2 \leq c^2, \\ \beta^+ \geq 0, \beta^2 \geq 0, \\ \beta^+ + \beta^- \leq d. \end{cases} \end{aligned}$$

The final decision rule is given by

$$f(\mathbf{x}^1, \mathbf{x}^2) = sign\left(0.5\left((\mathbf{w}^1)^\intercal \mathbf{x}^1 + b^1 + (\mathbf{w}^2)^\intercal \mathbf{x}^2 + b^2\right)\right).$$

4.2.2 Multiview Maximum Entropy Discriminant

For a supervised learning task, generative learning models the joint distribution over the input X and the output y, while discriminative learning directly models the conditional distribution of y given X. Discriminative learning can outperform generative learning on classification, but the latter takes the distribution of X into consideration and thus benefits some tasks.

Maximum entropy discrimination (MED) (Jaakkola et al. 2000) is a combination of generative and discriminative learning, which learns a discriminative classifier as well as considers uncertainties over model parameters. Consider a dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$, $y_l \in \{-1, 1\}$. MED wants to learn a discriminant function $L(x|\Theta)$ parameterized by a set of parameters Θ, which classifies the examples correctly at a desired classification margin, namely, $y_l L(x_l|\Theta) \geq \gamma_l$. Rather than finding a single set of parameters Θ, MED considers learning a distribution $p(\Theta)$ over Θ. Taking into account the case that training examples cannot be separated with the specified margin, MED also treats the margin $\boldsymbol{\gamma} = \{\gamma_l\}_{l\in[L]}$ as random variables. Incorporating our prior knowledge about parameter values into a prior distribution $p_0(\Theta, \boldsymbol{\gamma})$, we can find the posterior distribution closest to it by minimizing $\mathrm{KL}(p(\Theta, \boldsymbol{\gamma})||p_0(\Theta, \boldsymbol{\gamma}))$, the Kullback–Leiber divergence between them, under the classification constraints. Formally, MED was formulated as follows:

$$\begin{aligned} &\arg\min_{p(\Theta,\boldsymbol{\gamma})} \ \mathrm{KL}(p(\Theta, \boldsymbol{\gamma}) \,||\, p_0(\Theta, \boldsymbol{\gamma})) \\ &\quad s.t. \ \begin{cases} \int p(\Theta, \boldsymbol{\gamma})[y_l L(x_l|\Theta) - \gamma_l] d\Theta d\boldsymbol{\gamma} \geq 0, \\ l \in [L]. \end{cases} \end{aligned} \tag{4.3}$$

Consider a multiview learning setting with dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,\ldots,V,\ l=1,\ldots,L}$. For each view, we would like to find a discriminant function $L^v(x^v|\Theta^v)$ parameterized by a set of parameters Θ_v, satisfying $y_l L^v(x_l^v|\Theta^v) \geq \gamma_l^v$, $l \in [L]$, $v \in [2]$. In order to combine information from multiple views, we regularize the discriminant functions from different views to classify the training examples at the same confidence. MVMED (Sun and Chao 2013) uses a scheme named margin consistency, that is, to enforce the margins from different views to be identical, namely, $\gamma_l \ddot{=} \gamma_l^1 = \gamma_l^2$, $l \in [L]$. Taking into account the case that some training examples may not be classified with specified margins, we also treat the margin $\Theta = \{\Theta^v\}_{v\in[2]}$ as random variables, assigned with prior $p_0(\boldsymbol{\gamma})$ as in the MED framework. The following optimization is solved for the posterior $p(\Theta, \boldsymbol{\gamma})$:

$$\begin{aligned} &\arg\min_{p(\Theta,\boldsymbol{\gamma})} \ \mathrm{KL}(p(\Theta, \boldsymbol{\gamma}) \,||\, p_0(\Theta, \boldsymbol{\gamma})) \\ &\quad \text{s.t.} \ \begin{cases} \int p(\Theta, \boldsymbol{\gamma})[y_l L^1(x_l^1|\Theta^1) - \gamma_l] d\Theta^1 d\boldsymbol{\gamma} \geq 0, \\ \int p(\Theta, \boldsymbol{\gamma})[y_l L^2(x_l^2|\Theta^2) - \gamma_l] d\Theta^2 d\boldsymbol{\gamma} \geq 0, \\ l \in [L]. \end{cases} \end{aligned} \tag{4.4}$$

Problem (4.4) can be solved with the Lagrangian multiplier method. The Lagrangian of (4.4) is

$$L = \int p(\Theta, \boldsymbol{\gamma}) \log \frac{p(\Theta, \boldsymbol{\gamma})}{p_0(\Theta, \boldsymbol{\gamma})} d\Theta d\boldsymbol{\gamma} - \sum_{v=1}^{2} \sum_{l=1}^{L} \int p(\Theta, \boldsymbol{\gamma}) \lambda_l^v [y_l L/6v(x_l^v|\Theta/6v) - \gamma_l] \, d\Theta d\boldsymbol{\gamma},$$

where $\boldsymbol{\lambda}^v = \{\lambda_l^v\}_{l=1,\ldots,L}$ are Lagrange multipliers for view v.

In order to find the solution, we require

$$\begin{aligned} \frac{\partial L}{\partial p(\Theta, \boldsymbol{\gamma})} = & \quad \log \tfrac{p(\Theta,\boldsymbol{\gamma})}{p_0(\Theta,\boldsymbol{\gamma})} + 1 \\ & - \sum_{v=1}^{2} \sum_{l=1}^{L} \lambda_l^v [y_l L^v(x_l^v|\Theta^v) - \gamma_l] \\ = & \quad 0, \end{aligned}$$

which results in the following solution:

$$p(\Theta, \boldsymbol{\gamma}) = \frac{1}{Z(\boldsymbol{\lambda})} p_0(\Theta, \boldsymbol{\gamma}) \exp\left\{ \sum_{v=1}^{2} \sum_{l=1}^{L} \lambda_l^v [y_l L_v(x_l^v|\Theta^v) - \gamma_l] \right\}, \tag{4.5}$$

where $Z(\boldsymbol{\lambda})$ is the normalization constant and $\boldsymbol{\lambda} = \{\lambda_l^v\}_{v=1,2,l=1,\ldots,L}$ define a set of nonnegative Lagrange multipliers, one for each classification constraint. $\boldsymbol{\lambda}$ is set by finding the unique maximum of the following jointly concave objective function:

$$J(\boldsymbol{\lambda}) = -\log Z(\boldsymbol{\lambda}). \tag{4.6}$$

After $\boldsymbol{\lambda}$ is obtained, we can recover $p(\Theta) = \int p(\Theta, \boldsymbol{\gamma}) d\boldsymbol{\gamma}$, and use the following formula as the decision rule for classifying a new example $(x^1, x^2, \ldots, x^2)$

$$\hat{y} = \text{sign}\left(\frac{1}{2} \sum_{v=1}^{2} \int p(\Theta) \, L^v(x^v|\Theta^v) \, d\Theta \right).$$

With the jointly learned $p(\Theta)$, we can also make predictions on each view's own by

$$\hat{y} = \text{sign}\left(\int p(\Theta) \, L^v(x^v|\Theta^v) \, d\Theta \right).$$

In practice, we need to instantiate MVMED by determining the concrete formulation of the prior distribution $p_0(\Theta, \boldsymbol{\gamma})$, as well as the discriminant functions $L_v(x^v|\Theta_v)$. A common configuration is the linear discriminant function

$$L_v(x^v|\Theta_v) = (\theta^v)^\mathsf{T} x^v + b^v, v \in [2], \tag{4.7}$$

and

$$\begin{aligned} p_0(\Theta, \boldsymbol{\gamma}) &= [\prod_{v=1}^{2} p_0(\Theta^v)] p_0(\boldsymbol{\gamma}) \\ &= [\prod_{v=1}^{2} p_0(\boldsymbol{\theta}^v) p_0(b_v)] p_0(\boldsymbol{\gamma}), \end{aligned} \tag{4.8}$$

where $p_0(\boldsymbol{\theta}^v)$ is a Gaussian distribution with mean $\mathbf{0}$ and standard deviation I, $p_0(b^v)$ is set to the non-informative Gaussian distribution, and $p_0(\boldsymbol{\gamma})$ is assumed to be fully factorized, namely,

$$p_0(\boldsymbol{\gamma}) = \prod_{l \in [L]} p_0(\gamma_l), \tag{4.9}$$

with $p_0(\gamma_l) = \frac{c}{\sqrt{2\pi}} \exp\{-\frac{c^2}{2}(1-\gamma_l)^2\}$, a Gaussian prior with mean 1 that encourages large margins.

In this case, we have the following dual optimization problem:

$$\begin{aligned} \arg\max_{\boldsymbol{\lambda}} \quad & \sum_{v=1}^{2} \sum_{l=1}^{L} \lambda_l^v - \sum_{l=1}^{L} \frac{1}{2c^2} \left(\sum_{v=1}^{2} \lambda_l^v \right)^2 \\ & -\frac{1}{2} \sum_{v=1}^{2} \sum_{l,\tau=1}^{L} \lambda_l^v \lambda_\tau^v y_l y_\tau (x_l^v)^\mathsf{T} x_\tau^v \\ s.t. \quad & \begin{cases} \boldsymbol{\lambda}_v \succeq 0, \\ \sum_{l=1}^{L} \lambda_l^v y_l = 0, \\ v \in [2] \end{cases} \end{aligned} \tag{4.10}$$

where $\boldsymbol{\lambda}^v \succeq 0$ means every element of $\boldsymbol{\lambda}^v$ is nonnegative. The Lagrange multipliers $\boldsymbol{\lambda}$ is set by solving the convex optimization problem (4.10), whose nonzero values indicate the support vectors as reflected by (4.5). We can also replace the inner product $x_l^{v\mathsf{T}} x_\tau^v$ with some kernel function $k^v(x_l^v, x_\tau^v)$ to obtain a nonlinear classifier. Substituting (4.7), (4.8) and (4.9) into (4.5), we can recover the solution distribution $p(\Theta, \boldsymbol{\gamma})$. The prediction rule using view v can be given as

$$\begin{aligned}\hat{y} &= \text{sign}\left(\int p(\Theta)\, L^v(x^v|\Theta^v)\, d\Theta\right) \\ &= \text{sign}\left(\int p(\boldsymbol{\theta}^v, b^v)\, ((\boldsymbol{\theta}^v)^\intercal x^v + b_v)\, d\boldsymbol{\theta}^v db^v\right) \\ &= \text{sign}\left((\hat{\boldsymbol{\theta}}^v)^\intercal x^v + \hat{b}^v\right),\end{aligned}$$

where $\hat{\boldsymbol{\theta}}^v$ and $\hat{b}^v$ are the expected values related to the discriminant function. $\hat{\boldsymbol{\theta}}^v$ is obtained as follows:

$$\begin{aligned}\hat{\boldsymbol{\theta}}^v &= \int p(\Theta^v, \boldsymbol{\gamma})\boldsymbol{\theta}^v d\Theta^v d\boldsymbol{\gamma} = \int p(\boldsymbol{\theta}^v)\boldsymbol{\theta}^v d\boldsymbol{\theta}^v \\ &= \sum_{l=1}^{L} \lambda_l^v y_l x_l^v,\end{aligned}$$

and $\hat{b}^v$ can be given by the Karush–Kuhn–Tucker (KKT) conditions using support vectors.

4.2.3 Soft Margin-Consistency-Based Multiview Maximum Entropy Discrimination

MVMED exploits the multiple views by using margin consistency, that is, to enforce the margins from different views to be identical. Although MVMED has provided state-of-the-art multiview learning performance, this margin-consistency requirement may be too strong to fulfill in many cases. For example, all positive margins can lead to the same label prediction in binary classifications. To address this problem, soft margin-consistency-based multiview maximum entropy discrimination (SMVMED) (Mao and Sun 2016) is proposed. SMVMED achieves "soft" margin consistency by minimizing the sum of two KL divergences $\text{KL}(p(\boldsymbol{\gamma}) \,||\, q(\boldsymbol{\gamma}))$ and $\text{KL}(q(\boldsymbol{\gamma}) \,||\, p(\boldsymbol{\gamma}))$, where $p(\boldsymbol{\gamma})$ and $q(\boldsymbol{\gamma})$ are the posteriors of two-view margins, respectively. SMVMED also introduces a trade-off between maximum margin and margin consistency, and makes itself more scalable.

Consider a multiview dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$. SMVMED aims to learn two discriminant functions $L^1(x_l^1|\Theta^1)$ and $L^2(x_l^2|\Theta^2)$ for two views, respectively, where Θ^1 and Θ^2 are parameters of these two functions. SMVMED assumes that there are two independent distributions $p(\Theta^1)$ and $p(\boldsymbol{\gamma})$, with their joint distribution $p(\Theta^1, \boldsymbol{\gamma}) = p(\Theta^1)p(\boldsymbol{\gamma})$, where $\boldsymbol{\gamma} = \{\gamma_l\}$, $l = 1, \ldots, L$, is the margin parameter. Here $p(\Theta^1)$ is the posterior of Θ^1 and $p(\boldsymbol{\gamma})$ is the posterior of margins from view 1. Then the same settings are applied to view 2, that is, $q(\Theta^2, \boldsymbol{\gamma}) = q(\Theta^2)q(\boldsymbol{\gamma})$, where $q(\boldsymbol{\gamma})$ is the posterior of margins from view 2. Formally, SMVMED can be formulated as follows:

$$\begin{aligned}\underset{p(\Theta^1,\boldsymbol{\gamma}),\, q(\Theta^2,\boldsymbol{\gamma})}{\arg\min}\quad & \mathrm{KL}(p(\Theta^1)\,||\,p_0(\Theta^1)) + \mathrm{KL}(q(\Theta^2)||q_0(\Theta^2)) \\ & +(1-\alpha)\mathrm{KL}(p(\boldsymbol{\gamma})\,||\,p_0(\boldsymbol{\gamma})) + (1-\alpha)\mathrm{KL}(q(\boldsymbol{\gamma})||q_0(\boldsymbol{\gamma})) \\ & +\alpha\mathrm{KL}(p(\boldsymbol{\gamma})\,||\,q(\boldsymbol{\gamma})) + \alpha\mathrm{KL}(q(\boldsymbol{\gamma})\,||\,p(\boldsymbol{\gamma})) \\ \text{s.t.}\quad & \begin{cases} \int p(\Theta^1,\boldsymbol{\gamma})\,[y_l L_1(x_l^1|\Theta^1) - \gamma_l]\,d\Theta_1 d\boldsymbol{\gamma} \geq 0 \\ \int q(\Theta^2,\boldsymbol{\gamma})\,[y_l L_2(x_l^2|\Theta^2) - \gamma_l]\,d\Theta_2 d\boldsymbol{\gamma} \geq 0 \\ 1 \leq l \leq L. \end{cases}\end{aligned} \tag{4.11}$$

Since the margin priors are chosen to favor large margins, the parameter α above plays the trade-off role of leveraging large margin and soft margin consistency.

Since it is tricky to find the solutions making the partial derivatives of the Lagrangian of (4.11) with respect to $p(\Theta^1,\boldsymbol{\gamma})$ and $p(\Theta^1,\boldsymbol{\gamma})$ be zero, SMVMED proposes an iterative scheme for finding a solution to (4.11). In the mth iteration, SMVMED successively updates $p^{(m)}(\Theta^1,\boldsymbol{\gamma})$ and $q^{(m)}(\Theta^2,\boldsymbol{\gamma})$ by solving the following two problems:

$$\begin{aligned} & p^{(m)}(\Theta^1,\boldsymbol{\gamma}) \\ & = \underset{p^{(m)}(\Theta^1,\boldsymbol{\gamma})}{\arg\min}\quad \mathrm{KL}(p^{(m)}(\Theta^1)\,||\,p_0(\Theta^1)) \\ & \qquad + (1-\alpha)\mathrm{KL}(p^{(m)}(\boldsymbol{\gamma})\,||\,p_0(\boldsymbol{\gamma})) \\ & \qquad + \alpha\mathrm{KL}(p^{(m)}(\boldsymbol{\gamma})\,||\,q^{(m-1)}(\boldsymbol{\gamma})) \\ & \qquad s.t.\quad \begin{cases} \int p^{(m)}(\Theta^1,\boldsymbol{\gamma})\,[y_l L^1(x_l^1|\Theta^1) - \gamma_l]\,d\Theta^1 d\boldsymbol{\gamma} \geq 0 \\ 1 \leq l \leq L, \end{cases}\end{aligned} \tag{4.12}$$

and

$$\begin{aligned} & q^{(m)}(\Theta^2,\boldsymbol{\gamma}) \\ & = \underset{q^{(m)}(\Theta^2,\boldsymbol{\gamma})}{\arg\min}\quad \mathrm{KL}(q^{(m)}(\Theta^2)\,||\,q_0(\Theta^2)) \\ & \qquad + (1-\alpha)\mathrm{KL}(q^{(m)}(\boldsymbol{\gamma})\,||\,q_0(\boldsymbol{\gamma})) \\ & \qquad + \alpha\mathrm{KL}(q^{(m)}(\boldsymbol{\gamma})\,||\,p^{(m)}(\boldsymbol{\gamma})) \\ & \qquad s.t.\quad \begin{cases} \int q^{(m)}(\Theta^2,\boldsymbol{\gamma})\,[y_l L^2(x_l^2|\Theta^2) - \gamma_l]\,d\Theta^2 d\boldsymbol{\gamma} \geq 0 \\ 1 \leq l \leq L. \end{cases}\end{aligned} \tag{4.13}$$

The Lagrangian of (4.12) can be written as

$$
\begin{aligned}
L &= \int p^{(m)}(\Theta^1) \log \tfrac{p^{(m)}(\Theta^1)}{p_0(\Theta^1)} d\Theta^1 \\
&\quad +(1-\alpha) \int p^{(m)}(\boldsymbol{\gamma}) \log \tfrac{p^{(m)}(\boldsymbol{\gamma})}{p_0(\boldsymbol{\gamma})} d\boldsymbol{\gamma} \\
&\quad +\alpha \int p^{(m)}(\boldsymbol{\gamma}) \log \tfrac{p^{(m)}(\boldsymbol{\gamma})}{q^{(m-1)}(\boldsymbol{\gamma})} d\boldsymbol{\gamma} \\
&\quad -\sum_{l=1}^{L} \int p^{(m)}(\Theta^1, \boldsymbol{\gamma}) \lambda_{1,t}^{(m)} [y_l L^1(x_l^1|\Theta^1) - \gamma_l] \, d\Theta^1 d\boldsymbol{\gamma} \\
&= \int p^{(m)}(\Theta^1, \boldsymbol{\gamma}) \log \tfrac{p^{(m)}(\Theta^1, \boldsymbol{\gamma})}{p_0(\Theta^1)[p_0(\boldsymbol{\gamma})]^{1-\alpha}[q^{(m-1)}(\boldsymbol{\gamma})]^{\alpha}} \\
&\quad -\sum_{l=1}^{L} \int p^{(m)}(\Theta^1, \boldsymbol{\gamma}) \lambda_{1,t}^{(m)} [y_l L^1(x_l^1|\Theta^1) - \gamma_l] \, d\Theta_1 d\boldsymbol{\gamma},
\end{aligned}
\tag{4.14}
$$

where $\boldsymbol{\lambda}_1^{(m)} = \{\lambda_{1,t}^{(m)}\}$ is a set of nonnegative Lagrange multipliers, one for each classification constraint. After taking the partial derivative of (4.14) with respect to $p^{(m)}(\Theta_1, \boldsymbol{\gamma})$ and setting it to zero, we will obtain the solution to (4.12) which has the following form

$$
\begin{aligned}
p^{(m)}(\Theta^1, \boldsymbol{\gamma}) = &\tfrac{1}{Z_1^{(m)}(\boldsymbol{\lambda}_1^{(m)})} p_0(\Theta^1)[p_0(\boldsymbol{\gamma})]^{1-\alpha}[q^{(m-1)}(\boldsymbol{\gamma})]^{\alpha} \\
&\exp\left\{\sum_{l=1}^{L} \lambda_{1,l}^{(m)} [y_l L^1(x_l^1|\Theta^1) - \gamma_l]\right\},
\end{aligned}
$$

where $Z_1^{(m)}(\boldsymbol{\lambda}^{(m)})$ is the normalization constant. $\boldsymbol{\lambda}_1^{(m)}$ is set by finding the unique maximum of the following concave objective function:

$$
J_1^{(m)}(\boldsymbol{\lambda}_1^{(m)}) = -\log Z_1^{(m)}(\boldsymbol{\lambda}_1^{(m)}).
$$

Similarly, the solution to (4.13) is

$$
\begin{aligned}
q^{(m)}(\Theta^2, \boldsymbol{\gamma}) = &\tfrac{1}{Z_2^{(m)}(\boldsymbol{\lambda}_2^{(m)})} q_0(\Theta^2)[q_0(\boldsymbol{\gamma})]^{1-\alpha}[p^{(m)}(\boldsymbol{\gamma})]^{\alpha} \\
&\exp\left\{\sum_{l=1}^{L} \lambda_{2,l}^{(m)} [y_l L^2(x_l^2|\Theta^2) - \gamma_l]\right\},
\end{aligned}
$$

where $\boldsymbol{\lambda}_2^{(m)} = \{\lambda_{2,l}^{(m)}\}$ is another set of Lagrange multipliers. $\boldsymbol{\lambda}_2^{(m)}$ is set by finding the maximum of the following objective function:

$$
J_2^{(m)}(\boldsymbol{\lambda}_2^{(m)}) = -\log Z_2^{(m)}(\boldsymbol{\lambda}_2^{(m)}).
$$

The procedure is continued until convergence. After that, the decision rule for view v is given by

$$
\hat{y}^v = \operatorname{sign}\left(\int p(\Theta^v) \, L^v(x^v|\Theta^v) \, d\Theta^v\right).
$$

Before employing this iterative scheme, we first need to choose some initial value for $q^{(0)}(\Theta^2, \boldsymbol{\gamma})$. It is a proper choice to initialize $q^{(0)}(\Theta^2, \boldsymbol{\gamma})$ with $q_0(\Theta^2, \boldsymbol{\gamma})$, which makes (4.12) a standard MED problem.

4.3 Multiple Kernel Learning

The use of kernel tricks is equivalent to mapping the features of data to some feature space with a feature function, and operating on this space. It is important to choose a proper kernel function with its parameters, which is not a trivial problem and usually handled by a computationally inefficient cross-validation procedure. To address this problem, multiple kernel learning (MKL) (Gönen and Alpaydın 2011) was proposed, which uses a combination of multiple kernels instead of a single one,

$$k_\eta(x_i, x_j) = f_\eta\left(k_v((x_i^v, x_j^v))_{v\in[V]}\right), \tag{4.15}$$

where f_η is a combination function parameterized by η, and k_v are V different kernels. When handling multiview data, x_i^v is corresponding to the v-th view of the i-th sample. In other words, MKL uses different kernels for different views. When handling single-view data, we can either apply different feature extractors to the same sample and feed the outputs to different kernels, or simply feed the same feature to these kernels. Nevertheless, because of the equivalence of kernel methods and feature extractions, MKL can be viewed as a special case of multiview learning. Thus we devote a section to MKL here.

4.3.1 Kernel Combination

The basic idea of MKL is to replace a single kernel with a combination of multiple kernels. An immediate question is that under what conditions this combination is again a valid kernel. Joachims et al. (2001) lists a number of closure properties satisfied by kernels. For completeness, we also list them here.

For kernels k_1 and k_2, real number $\lambda \in [0, 1]$ and $a > 0$, the following functions are kernels:

$$\begin{aligned}
k(x_i, x_j) &= \lambda k_1(x_i, x_j) + (1-\lambda)k_2(x_i, x_j),\\
k(x_i, x_j) &= ak_1(x_i, x_j),\\
k(x_i, x_j) &= k_1(x_i, x_j)k_2(x_i, x_j).
\end{aligned}$$

Also, for function $f : \mathbb{R} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^m$ with a kernel k_3 over $\mathbb{R}^m \times \mathbb{R}^m$, the followings are kernels:

$$\begin{aligned}
k(x_i, x_j) &= f(x_i)f(x_j),\\
k(x_i, x_j) &= k_3(\phi(x_i), \phi(x_j)).
\end{aligned}$$

With these simple rules, we can already build some combinations of kernels. Since each of such combinations is again a kernel, these rules can be used to construct more complicated composition kernels with those simple combinations as building blocks (Li and Sun 2010).

From these rules, we can see that the construction of a combination kernel is of great flexibility, and thus some principles are preferred to instruct a construction, as well as the learning of the parameters therein (if any). Gönen and Alpaydın (2011) made a detailed introduction to a bunch of MKL algorithms as well as a classification of their principles. We will not go into that depth but focus on some methods of practice interests, in the respect of multiview learning.

4.3.2 Linear Combination of Kernels and Support Kernel Machine

The first combination method we consider is a simple linear combination. For a classification problem, given V kernel matrices $\{\boldsymbol{K}^v\}$, we consider a linear combination of the kernels $K = \sum_{v=1}^{V} \eta^v \boldsymbol{K}^v$ with positive combination coefficients $\{\eta^v\}$. The purpose is to learn the coefficients of the best linear combination under a trace constraint $trK = \sum_{v=1}^{V} \eta^v tr\boldsymbol{K}^v = c$ with c a positive constant, which yields the following optimization problem (Lanckriet et al. 2004),

$$\begin{aligned} \arg\min_{z,\mathbf{a}} \quad & z - \mathbf{1}^\intercal \mathbf{a} \\ s.t. \quad & \begin{cases} \mathbf{a}^\intercal \mathbf{y} = 0, \\ 0 \le \mathbf{a} \le C, \\ \mathbf{a}^\intercal \boldsymbol{D}_y \boldsymbol{K}_v \boldsymbol{D}_y \mathbf{a} \le tr\boldsymbol{K}^v z/c, v \in [V], \end{cases} \end{aligned} \tag{4.16}$$

where $\mathbf{y}$ is the classification labels, $\boldsymbol{D}_y$ is a diagonal matrix with diagonal $\mathbf{y}$, and C a positive constant.

Support kernel machine (SKM) (Bach et al. 2004) is a block-based variant of SVM. One important property of SKM is that its dual optimization formula is exactly the objective function of the linear combination of kernels, i.e, the problem (4.16).

Consider a multiview classification problem with dataset $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$, SKM seeks a linear classifier $y = sign(\sum_{v=1}^{V} (\mathbf{w}^v)^\intercal \mathbf{x}^v + b)$ classifying the dataset at a margin. The objective function of SKM is given by

$$\begin{aligned} \arg\min_{\{\mathbf{w}^v\}_{v=1,\ldots,V}} \quad & \frac{1}{2}\left(\sum_{v=1}^{V} d^v \left\|\mathbf{w}^v\right\|_2\right) + c\sum_{l=1}^{L} \xi_l \\ s.t. \quad & \begin{cases} y_l(\sum_{v=1}^{V} (\mathbf{w}^v)^\intercal \mathbf{x}_l^v + b) \ge 1 - \xi_l, \\ \xi_l \ge 0, \ l \in [L]. \end{cases} \end{aligned} \tag{4.17}$$

This problem can be solved by using the method of Lagrangian multipliers method, leading to the following dual problem:

$$\begin{aligned} \underset{\boldsymbol{\lambda},\gamma}{\arg\min} \quad & -\boldsymbol{\lambda}^{\intercal}\mathbf{1} + \frac{1}{2}\gamma^2 \\ s.t. \quad & \begin{cases} \boldsymbol{\lambda}^{\intercal}\mathbf{y}, \\ \mathbf{0} \le \boldsymbol{\lambda} \le C, \\ \left\| \sum_l \lambda_l y_l \mathbf{x}_l^v \right\|_2^2 \le d^v \gamma, v \in [V] \end{cases} \end{aligned}$$

As in SVM, kernel methods can be applied to SKM to handling nonlinear classification problems. This can be done by simply replacing the inner products in the formulation with their corresponding kernel terms, leading to the following formulation:

$$\begin{aligned} \underset{\boldsymbol{\lambda},\gamma}{\arg\min} \quad & -\boldsymbol{\lambda}^{\intercal}\mathbf{1} + \frac{1}{2}\gamma^2 \\ s.t. \quad & \begin{cases} \boldsymbol{\lambda}^{\intercal}\mathbf{y}, \\ \mathbf{0} \le \boldsymbol{\lambda} \le C, \\ \left(\boldsymbol{\lambda}^{\intercal}\boldsymbol{D}_y \boldsymbol{K}^v \boldsymbol{D}_y \boldsymbol{\lambda}\right)^{1/2} \le d^v \gamma, v \in [V] \end{cases} \end{aligned} \tag{4.18}$$

Now, let $d^V = \sqrt{tr\boldsymbol{K}^v/c}$, we can see the equivalence of (4.16) and (4.18). Although some care need to be taken in theory, it is sufficient for us to believe that SKM can solve classification by seeking a best linear combination of multiple kernels. What's more, SKM bridges the unfamiliar MKL with the familiar SVM-like algorithms, and thus facilitates the understanding and further developments of MKL.

4.3.3 SimpleMKL

SKM learns a linear classifier $y = sign(\sum_{v=1}^{V}(\mathbf{w}^v)^{\intercal}\mathbf{x}^v + b)$, or equivalently, a group of linear function $\{f^v(\mathbf{x}^v) = (\mathbf{w}^v)^{\intercal}\mathbf{x}^v\}$ with a single offset b. With the introduction of kernels, it is easy to show that each solution of SKM f^v indeed belongs to an RKHS with kernel function k^v which determines the corresponding kernel matrix $\boldsymbol{K}^v$. Under this considerations, the primal objective function of SKM can be written as

$$\begin{aligned} \underset{\{f^v\}_{v=1,\ldots,V},b,\boldsymbol{\xi}}{\arg\min} \quad & \frac{1}{2}\left(\sum_{v=1}^{V}\left\|f^v\right\|_{\mathscr{H}_m}^2\right)+c\sum_{l=1}^{L}\xi_l \\ s.t. \quad & \begin{cases} y_l(\sum_{v=1}^{V} f^v(\mathbf{x}_l^v)+b)\geq 1-\xi_l, \\ \xi_l\geq 0,\ \ l=1,\ldots,L. \end{cases} \end{aligned} \tag{4.19}$$

Here $\|\cdot\|_{\mathscr{H}_m}$ is the norm induced by the kernel $\sum_{v=1}^{V} d^v k^v(\cdot,\cdot)$, where $\{d^v\}$ are the coefficients appear in the objective of SKM (4.17). One problem with respect to this formulation is that it is not smooth since $\|f^v\|_{\mathscr{H}_m}$ is not differentiable at $f^v = 0$. To address this problem, Rakotomamonjy et al. (2008) proposed to replace the summation of the squared norm in (4.19) with a mixed-norm regularization term $\sum_{v=1}^{V}\frac{1}{d^v}\|f^v\|_{\mathscr{H}_m}$, $d^v\geq 0$, $\sum_{v=1}^{V} d^v = 1$ resulting in the following objective function

$$\begin{aligned} \underset{\{f^v\}_{v=1,\ldots,V},b,\boldsymbol{\xi}}{\arg\min} \quad & \frac{1}{2}\left(\sum_{v=1}^{V}\frac{1}{d^v}\left\|f^v\right\|_{\mathscr{H}_m}^2\right)+c\sum_{l=1}^{L}\xi_l \\ s.t. \quad & \begin{cases} y_l(\sum_{v=1}^{V} f^v(\mathbf{x}_l^v)+b)\geq 1-\xi_l, \\ \xi_l\geq 0,\ \ l=1,\ldots,L \\ \sum_{v=1}^{V} d^v = 1, \\ d^v\geq 0,\ \ v=1,\ldots,V \end{cases} \end{aligned} \tag{4.20}$$

Note that the mixed-norm regularization term is an upper bound of the original term, namely,

$$\sum_{v=1}^{V}\frac{1}{m^v}\left\|f^v\right\|_{\mathscr{H}_m}^2 \geq \left(\sum_{v=1}^{V}\left\|f^v\right\|_{\mathscr{H}_m}\right)^2, \tag{4.21}$$

which can be shown by the Cauchy–Schwartz inequality. In other words, the newly proposed objective upper bounds the original one. With this modification, the non-smooth objective function is turned into a smooth one. The dual problem can again be deduced by using the method of Lagrangian multipliers method, which has the following form,

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\arg\min} \quad & \boldsymbol{\lambda}^\top\mathbf{1}-a \\ s.t. \quad & \begin{cases} \boldsymbol{\lambda}^\top\mathbf{y}=0, \\ 0\leq\boldsymbol{\lambda}\leq c, \\ \frac{1}{2}(\boldsymbol{\lambda}\mathbf{y})^\top\boldsymbol{K}^v(\boldsymbol{\lambda}\mathbf{y})\leq a,\ \ v=1\ldots,V. \end{cases} \end{aligned} \tag{4.22}$$

This problem is hard to solve due to the last constraint. If move this constraint into the objective function, the objective function will be non-differentiable. To efficiently solve (4.20) and (4.22), (Rakotomamonjy et al. 2008) proposed an iterating algorithm which alternatively solves an SVM-like problem and updates the coefficients m^v, until the duality gap

$$G = J(m^*) - \sum_i \lambda_i^* + \frac{1}{2} \max_v \sum_{i,j} \lambda_i^* \lambda_j^* y_i y_j k^v(\mathbf{x}_i^v, \mathbf{x}_j^v) \tag{4.23}$$

vanishes.

4.4 Multiview Probabilistic Models

In this section, we will introduce some multiview probabilistic models. In particular, we will focus on Gaussian process based multiview models. Since in Gaussian process models the predict functions themselves are modeled as random variables, the view consensus can be measured through some measure between distributions of random variables.

4.4.1 Multiview Regularized Gaussian Processes

Gaussian Process (GP) is a powerful tool in various areas of machine learning. It is natural to seek its extension to multiview learning scenario. Indeed, we have already introduced one GP-based undirected graphic model, Bayesian co-training, which utilizes the conditional independence assumption of multiview learning. In this section, we will introduce another GP-based multiview model named multiview regularized Gaussian process (MRGP) (Liu and Sun 2017a).

Unlike Bayesian co-training, MRGP combines multiple views by regularizing the marginal likelihood with the consensus of the latent functions from different views. Specifically, MRGP models the classifier of each view as a Gaussian process model. Then the hyperparameters of the GPs are learned by maximizing a regularized marginal likelihood, which consists of a weighted average of the marginal likelihood of each view and a symmetric KL-divergence between the posteriors of the latent functions from the two views as a regularization term. This is a reasonable objective function for optimization of the hyperparameters, as it considers the fitness of each view as well as the consensus between the views. It also allows efficient and elegant inference and optimization.

As usual we have a multiview dataset $\mathscr{L} = \{(\mathbf{x}_l^v, y_l)\}_{v=1,2,\ l=1,\ldots,L}$. MRGP models each view as a GP model, that is, $p(\mathbf{f}^v|\mathbf{X}^v) = N(0, \mathbf{K}^v)$ and $p(\mathbf{y}|\mathbf{X}^v) = \int p(\mathbf{y}|\mathbf{f}^v, \mathbf{X}^v) p(\mathbf{f}^v|\mathbf{X}^v) df$, where $\mathbf{f}^v$ is the latent function for view v, $\mathbf{X}^v$ is the set contains all

the input variables form view v, $\mathbf{K}^v$ is view v's kernel matrix, and $p(\mathbf{y}|\mathbf{f}^v, \mathbf{X}^v) = N(\mathbf{f}^v, \sigma_v^2\mathbf{I})$ is the likelihood function.

For a traditional GP model, the hyperparameters (kernel parameters and likelihood parameters) are learned by maximizing the log marginal likelihood, which is given by

$$\log p(\mathbf{y}|\mathbf{X}^v) = -\frac{1}{2}\mathbf{t}^\intercal(\mathbf{K}^v + \sigma_v^2)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}^v + \sigma_v^2\mathbf{I}| - \frac{L}{2}\log 2\pi \quad (4.24)$$

for view v.

In order to utilize the information from different views, MRGP learns the hyperparameters of GPs from the two views simultaneously. Specifically, the objective function is given by

$$L = -\left[a\log p(\mathbf{y}|\mathbf{X}^1) + (1-a)\log p(\mathbf{y}|\mathbf{X}^v)\right] + \frac{b}{2}\left[KL(p_1||p_2) + KL(p_2||p_1)\right], \quad (4.25)$$

where a is a hyperparameter balancing two views, b is a hyperparameter controlling the strength of regularization, $\log p(\mathbf{y}|\mathbf{X}^v)$ is the log marginal likelihood given by (4.24) and p_v is the posterior of the latent function from view v, which is a Gaussian distribution with mean $\boldsymbol{\mu}_v = \mathbf{K}^v(\mathbf{K}^v + \sigma_v^2\mathbf{I})^{-1}\mathbf{y}$ and covariance $\boldsymbol{\Sigma}_v = \mathbf{K}^v - \mathbf{K}^v(\mathbf{K}^v + \sigma_v^2\mathbf{I})(-1)\mathbf{K}^v$. Note that the KL-divergence between two Gaussian distributions has analytical forms. Thus the KL-divergence terms in (4.25) are given by

$$KL(p_1||p_2) = \frac{1}{2}\left[\log|\boldsymbol{\Sigma}_2| - \log|\boldsymbol{\Sigma}_1|\right] + tr(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\intercal\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \mu_1) - L,$$

and similar for $KL(p_2||p_1)$.

One issue with MRGP is that it regularizes the marginal likelihood with the KL-divergence between joint distributions of all the latent responses, imposing the assumption that the classifiers from different views agree on all the training points, which is usually not the case in practice. To address this problem, MRGP introduces a so-called posterior consistency selection procedure. Specifically, after training two classifiers with MRGP, a consistent set is constructed, which is a subset of the training set, containing the training points whose predictions given by the two classifiers agree with their true labels. Then MRGP is run again, with the regularization term only dependent on the selected consistent set. Experimental results show the improvement carried by the additional posterior consistency selection procedure.

4.4.2 Sparse Multiview Gaussian Processes

One drawback of GP model is that it scales badly to large-scale datasets, which restricts its applicability to real-world problems. The main computational burden comes from calculating the inverse of an L-by-L matrix arising in the marginal

likelihood, with L the size of the training set. As a multiview extension of GP models, MRGP exhibits this shortcoming. A common solution to this problem is to use the so-called sparse GP models, where a low-rank decomposition of the kernel matrix is used to approximate the original full rank kernel matrix. Thus the inverse can be calculated more efficiently. The key to this kind of methods is how to decompose the kernel matrix such that as much as information of the original kernel matrix can be retained. Although a bunch of research has been carried on in the traditional single-view learning scenario, little has been done in multiview cases. In this section, we will introduce two multiview low-rank matrix approximation methods, and show their combination with MRGP (Liu and Sun 2017b).

In single-view cases, a simple approximation to the full-sample GP predictor is to keep the GP predictor on a subset of the training set named sparse set. To obtain a better approximation, the sparse set should retain sufficient information contained in the original training set. Based on the information theory, the information retained can be quantified by the differential entropy score, and each time the training point maximizing the quantity

$$\delta H_{in_i} = -\frac{1}{2}\log|\Sigma_{in_i}| + \frac{1}{2}\log|\Sigma_{i-1}|$$

is added to the sparse set, where Σ_{in_i} is the posterior covariance after choosing the n_i-th point at the selection, and Σ_{i-1} is the posterior covariance after the $(i-1)$-th choice (Herbrich et al. 2003).

In multiview cases, each view is modeled by a GP model. Let $\mathscr{T}$ denote the sparse set while $\mathscr{I} = \mathscr{L} - \mathscr{T}$ is the candidate set. Initially, $\mathscr{T}$ is a null set and $\mathscr{I}$ contains all the training points. At the i-th selection, the entropy reduction with the n_i point for view v is given by

$$\delta H^v_{in_i} = -\frac{1}{2}\log|\Sigma^v_{in_i}| + \frac{1}{2}\log|\Sigma^v_{i-1}|,$$

where $\Sigma^v_{in_i}$ is the posterior covariance for view v after choosing the n_i-th point at the selection, and Σ^v_{i-1} is the posterior covariance for view v after the $(i-1)$-th choice. The overall reduction associated with the n_i-th point is given by

$$\delta H_{in_i} = max_v(\delta H^v_{in_i}).$$

At the i-th selection, the $n^*_i = \arg\max_{n_i}(\{\delta H_{in_i}\}_{n_i \in I})$-th data point is removed from $\mathscr{I}$ and added to $\mathscr{T}$.

Because of the maximum of information and its sparsity nature, this scheme is named the maximum information vector machine (mIVM). The mIVM explores a sparse representation of multiview data, which leverages the information from the input data and the corresponding output labels.

The second scheme is called alternative manifold-preserving (aMP), which utilizes the graph of the data to seek a sparse representation. Given a training set, one

can construct a graph $G(\mathscr{V}, \mathscr{E}, \boldsymbol{W})$, where vertex set $\mathscr{V} = \{v_i\}$ consists of the data points and the weight $\boldsymbol{W}_{ij}$ measures the similarity between the i-th and j-th points. Manifold-preserving seeks a sparse graph G', which is a subgraph of G, having a high connectivity with G, that is, a candidate that maximizes the quantity

$$\frac{1}{L-s}\sum_{i=s+1}^{L}(\max_{1\le j\le s}\boldsymbol{W}_{ij}),$$

where s is the desired size of the sparse set (Sun et al. 2014).

In multiview cases, one graph $G^v(\mathscr{V}^v, \mathscr{E}^v, \boldsymbol{W}^v)$ is constructed for each view. Denote $d_i^v = \sum_{i\sim j}$ the degree of v_i^v, the i-th vertex from graph G^v, where $i \sim j$ means that there is an edge connecting the vertex v_i^v and v_j^v. The goal is to seek a sparse set $\mathscr{T}$ of size s. To begin with, aMP randomly chooses a view v. Then the vertex v_t^v with the largest degree in graph G^v is selected, and the t-th data point is added to $\mathscr{T}$. Since the t-th vertice of all the graphs represent the same data point, they cannot be selected again. Thus the t-th vertice of all the graphs with the edges connected to them should be removed from all the graphs. After that, another view is chosen and a similar selection procedure is carried on. This alternative selection procedure repeats until s data points are added into the index set $\mathscr{T}$.

As general multiview low-rank approximation schemes, they can be combined with any multiview GP models in principle. In particular, they can be combined with MRGP introduced above to get sparse multiview Gaussian process models, where the acceleration of training is achieved at a minor cost of accuracy.

4.5 Other Methods

Large Margin Multiview Information Bottleneck (LMIB) (Xu et al. 2014): The information bottleneck method is a technique from information theory, which aims to find the best trade-off between accuracy and compression when summarizing a random variable, given a joint distribution between it and a relevant variable. When applying it to multiview subspace learning, one take the subspace representation of a multiview variable as the random variable to be summarized and the input variables from multiple views as the relevant variable. The subspace representation is learned with a classifier of it, through minimizing the mutual information between it and observations while enforcing the classifier correctly classifies the represent at a margin. LMIB successfully combines the information bottleneck theory with the large margin principle in multiview learning settings, to learn a compact subspace as well as a classifier on it.

Alternative Multiview Maximum Entropy Discrimination (AMVMED) (Chao and Sun 2016): Multiview maximum entropy discrimination (MVMED) effectively combines Bayesian learning and the large margin principle in multiview settings,

which learns a joint posterior distribution of the parameters of the large margin classifiers from two views and a group of margin variables simultaneously. MVMED implicitly assumes that the two views are of the equivalent importance, which may not be the case in practice. To address this problem, AMVMED proposes to learn two joint posterior distributions of the parameters and the margin variable, one for a view. An additional hyperparameter is incorporated to trade-off the importance between the views. In order to perform the margin consistency, AMVMED enforces the marginal distributions of the margin variables form two views to be identical. The problem is solved by a two-step procedure. The first step is to solve the problem without considering the equal posteriors of the two-view margins, and the second step is to make the posteriors of the two-view margins the same. AMVMED is more flexible than MVMED, and includes two single-view Maximum Entropy Discriminations as its special cases, which is of particular interesting when the data from one of the view is corrupted.

Multiview Privileged Support Vector Machines (**PSVM-2V**) (Tang et al. 2018): Learning using privileged information (LUPI) is a framework to handle the setting where not only the input/output pairs of the task are available, but also some additional privileged information. In multiview settings, multiple views can provide privileged information to complement each other. Thus the techniques used in LUPI can be incorporated into multiview learning. PSVM-2V is such an example utilizing this idea, which can be view as a modification of SVM-2K. Specifically, the privileged information from one view is used to reformulate the slack variables of the other. This modification indeed meets the principle of view complementarity. Along with the principle of view consensus instructing the formulation of SVM-2K, the principles of view complementarity are two basic principles of multiview learning. Thus PSVM-2V is a theoretically well-rooted multiview learning method.

References

Bach FR, Lanckriet GRG, Jordan MI (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st international conference on machine learning. ACM, p 6

Chao G, Sun S (2016) Alternative multiview maximum entropy discrimination. IEEE Trans Neural Netw 27(7):1445–1456

Farquhar JDR, Hardoon DR, Meng H, Shawe-Taylor J, Szedmak S (2006) Two view learning: SVM-2K, theory and practice. In: Advances in neural information processing systems, vol 18, pp 355–362

Gönen M, Alpaydın E (2011) Multiple kernel learning algorithms. J Mach Learn Res 12(Jul):2211–2268

Herbrich R, Lawrence ND, Seeger M (2003) Fast sparse gaussian process methods: the informative vector machine. Adv Neural Inf Process Syst 15:625–632

Jaakkola TS, Meila M, Jebara T (2000) Maximum entropy discrimination. In: Advances in neural information processing systems, vol 12, pp 470–476

Joachims T, Cristianini N, Shawe-Taylor J (2001) Composite kernels for hypertext categorisation. In: Proceedings of the 18th international conference on machine learning. ACM, pp 250–257

Lanckriet GR, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. J Mach Learn Res 5:27–72

Li J, Sun S (2010) Nonlinear combination of multiple kernels for support vector machines. In: Proceedings of the 20th international conference on pattern recognition, pp 2889–2892

Liu Q, Sun S (2017a) Multi-view regularized gaussian processes. In: Proceedings of the 21st Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 655–667

Liu Q, Sun S (2017b) Sparse multimodal Gaussian processes. In: Proceedings of 2017 international conference on intelligent science and big data engineering, pp 28–40

Mao L, Sun S (2016) Soft margin consistency based scalable multi-view maximum entropy discrimination. In: Proceedings of the 25th international joint conference on artificial intelligence, pp 1839–1845

Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y (2008) SimpleMKL. J Mach Learn Res 9(11):2491–2521

Sun S, Chao G (2013) Multi-view maximum entropy discrimination. In: Proceedings of the 23rd international joint conference on artificial intelligence, pp 1706–1712

Sun S, Hussain Z, Shawe-Taylor J (2014) Manifold-preserving graph reduction for sparse semi-supervised learning. Neurocomputing 124(1):13–21

Sun S, Shawe-Taylor J, Mao L (2017) Pac-bayes analysis of multi-view learning. Inf Fusion 35:117–131

Tang J, Tian Y, Zhang P, Liu X (2018) Multiview pivileged support vector machines. IEEE Trans Neural Netw 29(8):3463–3477

Xu C, Tao D, Xu C (2014) Large-margin multi-view information bottleneck. IEEE Trans Pattern Anal Mach Intell 36(8):1559–1572

Chapter 5
Multiview Clustering

Abstract This chapter introduces three kinds of multiview clustering methods. We begin with the multiview spectral clustering, where the clustering is carried out through the partition of a relationship graph of the data. It depends on the eigenvector of the adjacent matrix of the data. Then we consider the multiview subspace clustering, which aims at recovering the underlying subspace of the multiview data and performs clustering on it. Finally, we introduce distributed multiview clustering, which first learns the patterns from each view individually and then combines them together to learn optimal patterns for clustering, and multiview clustering ensemble. It combines the results of multiple clustering algorithms to obtain better performance. We also briefly introduce some other methods at the end of this chapter.

5.1 Introduction

Clustering is the task of partitioning the data into several different groups (clusters), such that the data points in the same cluster are more similar with each other than that from other clusters. In multiview cases, although the data points are represented by different feature sets (views), ideally, their relative similarity should be the same across the views. In other words, if there exists an underlying true clustering, it should assign a data point to the same cluster irrespective of the view. This sounds like the view consensus we have met in the previous chapters, and is indeed an assumption of many multiview clustering algorithms we will introduce below.

We mainly consider three kinds of multiview clustering methods in this chapter. We begin with the multiview spectral clustering, where the clustering is carried out through the partition of a relationship graph of the data, which depends on the eigenvector of the adjacent matrix of the data. Then we consider the multiview subspace clustering, which aims at recovering the underlying subspace of the multiview data and performs clustering on it. Finally, we introduce distributed multiview clustering and multiview clustering ensemble. Distributed multiview clustering first learns the patterns from each view individually and then combines them together to learn optimal patterns for clustering. Multiview clustering ensemble combines the results of multiple clustering algorithms to obtain better performance.

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_5

5.2 Multiview Spectral Clustering

5.2.1 Co-trained Spectral Clustering

As one of the most classical multiview learning algorithms, co-training is a quite reasonable start point for developing multiview learning algorithms for tasks other than semi-supervised learning. In this section, we will introduce co-trained spectral clustering (Kumar and Daum 2011), which is an adaption of the classical spectral clustering to multiview setting with the idea of co-training.

To begin with, we briefly review the spectral clustering (Von Luxburg 2007). Spectral clustering is a theoretically well-rooted algorithm. As the name suggests, it is based on the spectral graph theory. Specifically, the clustering is carried out through the partition of a relationship graph of the data, which depends on the eigenvector of the adjacent matrix of the data.

Given a dataset $\mathscr{U} = \{\mathbf{x}_u\}_{u=1,\ldots,U}$, spectral clustering first constructs a data adjacency matrix $\boldsymbol{W}$, where W_{ij} quantifies the similarity between data point $\mathbf{x}_i$ and $\mathbf{x}_j$. The normalized graph Laplacian L is calculated. One can either use an unnormalized one as usual, or a normalized one as in Kumar and Daum (2011)'s work. Here we follow the latter. In this case, $L = D^{-1/2}MD^{-1/2}$, with D a diagonal matrix whose diagonal element $D_{ii} = \sum_j W_{ij}$. After that, we calculate the top k eigenvectors of L, where k is the desired number of the clusters, to form a matrix U, whose the l-th column is the l-th eigenvector of L. After that a new matrix V is obtained by normalizing each row of U. Then the K-means algorithm is run to cluster the rows of V, and example i is assigned to the cluster where the i-th row of V belongs.

Now we come to the multiview case, with a multiview dataset $U = \{\mathbf{x}_u^1, \mathbf{x}_u^2\}$. First, recall that the basic idea of co-training for semi-supervised learning is to use the unlabeled data to restrict the search space of the classifiers from two views, that is, the classifiers from two views should agree on the unlabeled dataset. In practice, co-training first trains two base classifiers with the labeled data from each view, then alternatively updates the classifiers with the unlabeled data labeled by the classifier from the other view. For a clustering task, since there are no labeled data, the scheme used by co-training cannot be directly applied. However, the basic idea that the predict functions (clusterings here) should agree on unlabeled data is still reasonable. As mentioned, each data point should be assigned to the same cluster for all the views.

For multiview cases, one adjacency matrix W^v is constructed for each view, with the corresponding Laplacian L^v, $v \in \{1, 2\}$. It is known that the first k eigenvectors of L contain the discrimination information used for clustering. Thus it is reasonable to use the eigenvectors from one view to propagate its clustering information to the other view. Since the spectral clustering only depends on the adjacency matrices, co-trained spectral clustering uses these information to modify the adjacency matrices and thus the underlying graph structures. Specifically, each column $\mathbf{m}_i$ of M indicates the similarities of the i-th data point with all the points in the graph. Co-trained

spectral clustering first projects these column vectors along the directions of the first k-th eigenvectors of the Laplacian from the other view to incorporate the clustering information from it, then back-projects them to the original space to obtain a modified graph. In order to obtain a symmetric matrix, an additional symmetrization step is carried out. Formally, the following two steps are performed alternatively:

$$S^1 = sym\left(U^2(U^2)^\intercal M^1\right),$$
$$S^2 = sym\left(U^1(U^1)^\intercal M^2\right),$$

where S^v is the updated adjacency matrix, U^v is the matrix containing the first k eigenvectors of the Laplacian of S^v, and sym indicates the matrix symmetrization. These two steps are performed until reaching some maximum iteration number. After that, U^1 and U^2 are normalized and column-wise concatenated. The final clustering depends on the K-means performed on the concatenated matrix.

Co-trained spectral clustering utilizes the idea of co-training to clustering multiview data. The idea is to learn the clustering in one view and use it to label the data in the other view so as to modify the graph structure. The modification to the graph is dictated by the discriminative eigenvectors and is achieved by projection along these directions.

5.2.2 *Co-regularized Spectral Clustering*

Co-trained spectral clustering tacks the multiview clustering problem by propagating the clustering information from one view to the other alternatively. Co-regularization deals with the same semi-supervised learning problem under the regularization framework by solving a single optimization problem and is of great flexibility. It is natural to ask whether the idea of co-regularization can be applied to multiview clustering. The answer is, of course, yes. In this section, we will introduce co-regularized spectral clustering (Kumar et al. 2011), which proposes one single objective function for spectral clustering and implicitly combines graphs from different views for better clustering.

The first step to co-regularized spectral learning is an observation that the standard spectral clustering can be reformulated as an optimization problem. Given a single dataset set $U = \{\mathbf{x}_u\}$ with its normalized Laplacian L, the standard spectral clustering first solves the following optimization problem,

$$\begin{aligned} \max_{U \in R^{n \times k}} \quad & tr\,(U^\intercal L U) \\ s.t. \quad & U^\intercal U = I, \end{aligned}$$

where n is the size of the dataset and k is the expected cluster number. (To see the relationship between this problem and the eigenvalue problem that arises in the

spectral clustering problem introduced in the last section, recall how we solve the problem of CCA.) After that, the rows of matrix U are given to the K-means algorithm to obtain the clustering results.

In multiview cases, we deal with a multiview dataset $U = \{\mathbf{x}_u^1, \mathbf{x}_u^2\}$ with the normalized Laplacian L^v, $v \in 1, 2$, and solve for U^v for clustering. Since U^v is the data representation for the subsequent K-means clustering step, one can expect that enforcing the similarity of $\{U^v\}$ leads to the agreement of the clusterings for two views. Different schemes can be used to impose such enforcements. We will introduce one particular scheme arising in the original co-regularized spectral learning paper (Kumar et al. 2011) named pairwise co-regularization.

The basic idea of pairwise co-regularization is to encourage the representations of the same data point to be similar across the views. Specifically, the following cost function is used to measure the disagreement between clusterings of two views:

$$D(U^1, U^2) = ||\frac{K^1}{||K^1||_F^2} - \frac{K^2}{||K^2||_F^2}||_F^2, \tag{5.1}$$

where K^v is the similarity matrix for U^v, $||\cdot||_F$ denotes the Frobenius norm of the matrix. For the case that a linear kernel is used to measure the similarity for both views, we have $K^v = U^v U^{v\intercal}$, and the cost function (5.1) reduces to the following simple form

$$D(U^1, U^2) = -tr(U^1(U^1)^\intercal U^2(U^2)^\intercal).$$

Now we can easily incorporate this cost function as a regularization term into a naive multiview clustering objective function consisting of an addition of two single-view terms to obtain the first objective function of co-regularized spectral clustering,

$$\begin{aligned} \max_{U^1 \in R^{n\times k}, U^2 \in R^{n\times k}} \quad & tr\left((U^1)^\intercal L^1 U^1\right) + tr\left((U^2)^\intercal L^2 U^2\right) + \alpha\ tr(U^1(U^1)^\intercal U^2(U^2)^\intercal), \\ s.t. \quad & (U^1)^\intercal U^1 = I, (U^2)^\intercal U^2 = I, \end{aligned}$$

where α is a trade-off hyperparameter. This problem can be solved by alternative maximization with respect to U^1 and U^2. Given U^2, we can solve the following problem for U^1:

$$\begin{aligned} \max_{U^1 \in R^{n\times k}} \quad & tr\left((U^1)^\intercal \left(L^1 + \alpha U^2 U^{2\intercal}\right) U^2\right), \\ s.t. \quad & (U^1)^\intercal U^1 = I, \end{aligned}$$

which has the same form of the objective function of the standard spectral clustering and can be easily solved. In practice, this alternative optimization scheme converges very fast. Either U^1 or U^2 can be used in the final K-means clustering. The observed difference is marginal (Kumar et al. 2011).

5.3 Multiview Subspace Clustering

When dealing with high-dimensional data like images, it is inefficient to clustering in the original feature space because of the complexity and the computational expensiveness due to the high dimensions. Rather, one may assume the data are drawn from low-dimensional subspaces. Subspace clustering aims at finding such subspaces and performs clustering in these subspaces for better accuracy. For multiview cases, the subspace assumption also makes sense. However, it is of great freedom how a multiview clustering algorithm utilizes multiple subspaces corresponding to different views. Of course, usually there does not exist a single correct answer. In this section, we will introduce several multiview subspace clustering algorithms.

5.3.1 Multiview Clustering via Canonical Correlation Analysis

To cluster multiview data, a naive idea comes to one's mind may be to perform CCA on the original data, cascade the resulting subspace representations and apply any single-view clustering algorithm to it. It works. But we can do better. Indeed, under the assumption that the views are uncorrelated given the cluster label, with appropriate application of CCA and choice of the clustering algorithm, the success of the clustering can be promised when the true data comes from a mixture of K distributions (Chaudhuri et al. 2009). We will briefly introduce the related ideas in this section.

Given a multiview dataset $U = \{\mathbf{x}_u^1, \mathbf{x}_u^2\}$, with covariance matrices

$$\Sigma_{11} = E[x^1(x^1)^\intercal], \Sigma_{22} = E[x^2(x^2)^\intercal], \Sigma_{12} = E[x^1(x^2)^\intercal].$$

Data are assumed to be of zero mean for simplicity.

We work with two main assumptions (Chaudhuri et al. 2009). The first is called multiview condition, which assumes that conditioned on the cluster label l, the two views are uncorrelated, that is,

$$E[x^1(x^2)^\intercal | l = i] = E[x^1 | l = i]E[x^2 | l = i].$$

This assumption implies that $E_{12} = \sum_i w_i \mu_i^1 (\mu_i^2)^\intercal$, where μ_i^v is the mean of distribution i in view v, and w_i is the mixing weight for distribution i. The second assumption is called non-degeneracy condition, which assumes that Σ_{12} has rank $k-1$ and the minimal nonzero singular value of Σ_{12} is $\lambda_0 > 0$. Besides these two main assumptions, it is also assumed that the covariance matrix of the data in each view is the identity matrix, that is, $\Sigma_{11} = I$ and $\Sigma_{22} = I$, which can be easily realized by whitening the data.

Now we formally introduce the algorithm (Chaudhuri et al. 2009). Given the dataset $U = \{\mathbf{x}_u^1, \mathbf{x}_u^2\}$, assume that the data in view 1 obeys the separation condition, that is the means of distributions of the true data mixture distribution are separated at a margin. The algorithm contains three steps. First, the dataset U is randomly partitioned into two subsets of equal sizes denoted as A and B. After that, compute the top $k-1$ left singular vectors of the empirical covariance matrix $\hat{\Sigma}_{12}(A)$ between view 1 and view 2 of data points in subset A, and project the data points in B to the subspace spanned by these singular vectors. Finally, apply appropriate single-view clustering to the projected data.

When the true data distribution is a mixture of Gaussian distributions and the separation condition is obeyed by any one of the views, and the clustering algorithm used is the one in Section 3.5 of Achlioptas and McSherry (2005), it can be proved the algorithm correctly clustering the data with a probability if the data size is large enough. In practice, one neither has access to the true data distribution nor can exam the separation condition. Thus no guarantee of clustering correctness is available. Nevertheless, this algorithm works well empirically.

5.3.2 Multiview Subspace Clustering

Basing on a well-performed single-view algorithm and extending it to multiview settings is a basic idea to develop multiview learning algorithms. We have already shown its success on different learning tasks, and, of course, it works on multiview subspace clustering as well. In this section, we will introduce such an example, which combines subspaces learned by single-view subspace learning algorithms on each view with a common cluster structure to cluster multiview data (Gao et al. 2015).

To begin with, we briefly introduce the single-view algorithms used. Given a single-view dataset $U = \{\mathbf{x}_u\}$ with data matrix $X \in R^{D\times N}$. The subspace clustering algorithm uses the self-expression property of the data matrix to represent itself, that is,

$$X = XZ + E,$$

where $Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N] \in R^{N\times N}$ is representation matrix with $\mathbf{z}_i$ is the subspace representation of $\mathbf{x}_i$, and $E \in \mathbb{R}^{D\times N}$ is the representation error matrix. With this self-expression, the single-view subspace clustering algorithm solves the following optimization problem: (Elhamifar and Vidal 2013),

$$\begin{aligned} \min_{Z} \quad & ||X - XZ||_F^2 \\ s.t. \quad & Z_{ii} = 0, Z^\intercal \mathbf{1} = \mathbf{1}, \end{aligned}$$

where $||\cdot||_F$ denotes the Frobenius norm of the matrix. After solving this problem, we obtain the representation matrix Z as well as a similarity matrix $W = \frac{|Z|+|Z^\intercal|}{2}$. Then we can perform spectral clustering on the subspace by solving

$$\begin{aligned} \max_{U} \quad & tr\,(U^\intercal L U) \\ s.t. \quad & U^\intercal U = I, \end{aligned}$$

where L is the graph Laplacian calculated from W.

Then we consider the multiview case with data matrix X^v from view v. The key to multiview clustering is how to combine different views to obtain better subspace representations for clustering. One idea is to separate the subspace learning step and the clustering step, and introduce a unified subspace representation Z to force the similarity between the representation Z^v from different views. This consideration can be formulated as the following problem: (Gao et al. 2015),

$$\begin{aligned} \min_{Z^v, Z} \quad & \sum_v ||X^v - X^v Z^v||_F^2 + \lambda \sum_v ||Z - Z^v|| \\ s.t. \quad & Z^v_{ii} = 0, (Z^v)^\intercal \mathbf{1} = \mathbf{1}. \end{aligned} \tag{5.2}$$

After solving this, one can perform clustering on the unified representation Z. One drawback of this method is that the learned representation is not guaranteed to be suitable for clustering. Rather than consider a multiview subspace learning problem solely, enforce the clustering results using different Z^v to be similar and integrate this consideration into the learning of these subspace representations is a better idea. This can be done by solving the following problem (Gao et al. 2015):

$$\min_{Z^v, Z} \quad \sum_v ||X^v - X^v Z^v||_F^2 + \lambda \sum_v Tr(U^\intercal L^v U) \tag{5.3}$$

$$s.t. \quad Z^v_{ii} = 0, Z^{v\intercal} \mathbf{1} = \mathbf{1}, U^\intercal U, \tag{5.4}$$

where L^v is the graph Laplacian corresponding to the subspace representation of view v. This objective function is indeed a combination of that of the subspace learning and the clustering problems from all the views. Since U contains all the clustering information and can be viewed as a clustering indicator, using the same U across the views indeed forces the clustering results from different views to be the same. Thus the main difference of (5.3) from (5.2) is the replacement of a term encouraging the similarity between the subspace representations with a term enforcing the consistency of the clustering results from different views.

5.3.3 Joint Nonnegative Matrix Factorization

Nonnegative matrix factorization (NMF) is a low-rank matrix approximation algorithm with a strong relationship with kernel K-mean clustering (Ding et al. 2005) and is widely used for clustering. Given a nonnegative data matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in R_+^{D\times N}$. NMF seeks an approximation to X,

$$X \approx UV^\intercal,$$

where $U \in R_+^{D\times K}$ and $V \in R_+^{N\times K}$ are two nonnegative matrix factors known as the basis matrix and the coefficient matrix, respectively. One can take the n-th row of V as a K-dimensional subspace representation of the n-th data point $\mathbf{x}_n$ where the subspace is spanned by the columns of U, and thus use NMF for subspace learning.

NMF can also be reformulated as an optimization problem minimizing some divergence between the data matrix X and the approximation $UV^\intercal$. One of the common choices of the divergence is the Frobenius norm, leading to the following optimization problem:

$$\begin{aligned} \min_{U,V} \quad & ||X - UV^\intercal||_F^2 \\ s.t. \quad & U \geq 0, V \geq 0. \end{aligned}$$

Since this objective function is not convex in U and V, it is not guaranteed to find the global minimum. A commonly used scheme to minimize this function is the multiplicative update rule, that is,

$$U_{ik} \leftarrow U_{ik}\frac{(XV)_{ik}}{(UV^\intercal V)_{ik}}, V_{jk} \leftarrow V_{jk}\frac{(XU)_{jk}}{(VU^\intercal U)_{jk}}.$$

Also note that since

$$UV^\intercal = (UQ^{-1})(QV^\intercal)$$

for any invertible matrix $Q \in R^{K\times K}$, there could be many solutions to NMF. Thus further constraints need to be imposed to ensure the uniqueness of the factorization.

In multiview cases, we work with data matrices $\{X^v\}$ from multiple views. To combine multiple views, we require coefficient matrices learned from different views to be consensus, which can be performed by penalizing the objective with the Frobenius norm between the coefficient matrices and a consensus matrix. Under this consideration, the objective function for multiview NMF is given as follows (Liu et al. 2013),

$$\min_{\{U^v,V^v\},V^*} \quad \sum_v ||X^v - U^v\{V^v\}^\intercal||_F^2 + \sum_v \lambda_v ||V^v - V^*||_F^2, \tag{5.5}$$

$$s.t. \quad ||U_k^v||_1 = 1, and, U^v, V^v, V^* \geq 0, \tag{5.6}$$

where U^v and V^v are the base matrix and the coefficient matrix for view v, respectively, and V^* is the consensus matrix, and the constraints on U^k promise the uniqueness of the solution. Let Q^v be a diagonal matrix with $Q^v_{kk} = \sum_i^M U^v_{ik}$. Then (5.5) can be reformulated to be an equivalent but simpler problem,

$$\begin{aligned} \min_{\{U^v, V^v\}, V^*} & \sum_v ||X^v - U^v\{V^v\}^\intercal||_F^2 + \sum_v \lambda_v ||V^v Q^v - V^*||_F^2, \\ s.t. \;\; & U^v, V^v, V^* \geq 0. \end{aligned}$$

This can be solved by alternatively updating U^v and V^v, and V^* (Liu et al. 2013).

5.4 Distributed Multiview Clustering

Most of the multiview clustering algorithms introduced so far are of a centralized style, that is, different views are utilized simultaneously to discover the underlying patterns of the data, mostly based on the consensus of the views. However, algorithms of this style have some limitations that restrict their applications to some practical scenario. For example, when the representations of different views are of great diversity, it is usually not clear how to measure the consensus of the views. Also, in the case different views are collected from different sensors and stored distributively, centralized algorithms suffer a data transfer bottleneck. Thus it is of practical interest to discuss multiview clustering algorithms that learn the patterns from each view individually and then combine them together to learn optimal patterns for clustering, that is, the distributed style algorithms. In this section, we will introduce such a framework for distributed multiview unsupervised learning and focus its application on clustering (Long et al. 2008).

First, we give some words on notations. We have multiview data represented by multiple matrices $\{X^v\}$, each for a view. These matrices can either be the data matrices $X^v \in R^{N\times D}$, or the adjacency matrices $X^v \in R^{N\times N}$ as used in the spectral clustering. Rather than dealing with these matrices directly, a pattern matrix $A^v \in R^{N\times K_v}$ is learned from X^v of each view individually, and the framework focuses on learning an optimal pattern $B \in R^{N\times K}$ from multiple patterns $\{A^v\}$. Note that the pattern matrix A^v can be learned from any appropriate single-view learning algorithm, such as the subspace representation learned by a subspace clustering algorithm, the cluster indicator learned by a spectral learning algorithm, and so on.

To learn the optimal pattern B, we still need to exploit the intuition that the patterns from different view should have some consensus, and the optimal pattern should be the consensus pattern which is shared by multiple views. We nevertheless need some distance to measure the similarity between patterns. Since pattern matrices from different views lay in different space, a direct comparison is impossible. To address this problem, we can introduce a mapping between the spaces of two pattern matrices. Specifically, for the optimal pattern $B \in R^{N\times K}$ and the pattern $A^v \in R^{N\times K_v}$ from

view v, a mapping $f_v : R^{N\times K} \to R^{N\times K_v}$ makes $f_v(B)$ and A^v directly comparable. With a group of mapping functions f_v, we can learn the optimal pattern through the following optimization problem (Long et al. 2008):

$$\min_{B,\{f_v\}} \sum_v w_v l(A^v, f_v(B)),$$

where $\{w_v\}$ are a group of nonnegative weights encoding the belief on the relative importance of the views, and $l(\cdot,\cdot)$ is a distance function measuring the similarity between patterns, depending on the concrete learning task.

To apply the framework to clustering, we need to determine the parametric form of the mapping functions as well as the distance function. For convenience, we use the linear mapping functions here, that is, $f_v(B) = BP^v$, with $P^v \in R^{K\times K_v}$. For the distance function l, we use the generalized I-divergence function GI, leading to the following optimization problem: (Long et al. 2008),

$$\begin{aligned} \min_{B,\{P^v\}} \quad & \sum_v w_v GI(A^v||BP^v), \\ s.t. \quad & B\mathbf{1} = \mathbf{1}. \end{aligned}$$

Here the constraint $B\mathbf{1} = \mathbf{1}$ suggests that each row of the optimal solution can be a (soft) indicator of clustering. To simplify the optimization, one can transfer this constraint to a soft constraint by adding a penalty term $\alpha GI(\mathbf{1}||B\mathbf{1})$, leading to Long et al. (2008),

$$\min_{B,\{P^v\}} \sum_v w_v GI(A||BP) + \alpha GI(\mathbf{1}||B\mathbf{1}),$$

where $A = [A^1, \ldots, A^V]$ and $P = [P^1, \ldots, P^V]$, since generalized I-divergence is a separable distance function. Then the problem can be solved with the bound optimization procedure (Lee and Seung 1999; Salakhutdinov and Roweis 2003). We directly give the deduced updating rule for B

$$B_{ig} = \tilde{B}_{ig} \frac{\sum_j A_{ij} \frac{P_{gj}}{[\tilde{B}P]_{ij}} + \frac{\alpha}{[\tilde{B}\mathbf{1}]_i}}{\sum_j P_{gj} + \alpha},$$

and the updating rule for P

$$P_{gj} = \tilde{P}_{gj} \frac{\sum_i \frac{A_{ij} B_{ig}}{[B\tilde{P}]_{ij}}}{\sum_i N_{ig}},$$

where $\tilde{\cdot}$ is the solution from the previous iteration. The convergence of this distributed multiple clustering algorithm is guaranteed.

5.5 Multiview Clustering Ensemble

When applied to the same data, different clustering algorithms does not necessarily produce the same results. Here comes a natural question, say, which one is correct? Although there exist some indices evaluating the results of the clustering, different indexes are again diverse, which make us get stuck in a loop. Nevertheless, we can never answer the question unless we have the underlying true cluster labels, which makes the question itself uninteresting. Indeed, since one can only apply a finite (quite small, in practical) set of clustering algorithms to the data, one seldom expects to obtain a correct answer from them. Rather, the diversity of different clustering algorithms provides us a chance to improve any of them. In this section, we will introduce clustering ensembles, which combine different clustering results.

The basic idea of clustering ensembles (Topchy et al. 2005; Zhou and Tang 2006) is that different component clusterings are combined to a better final partition via a consensus function, where different component clusterings can be obtained by different clustering algorithms or other approaches. Given a dataset $U = \{\mathbf{x}_u\}_{u=1,dots,U}$ and H clustering algorithms. The set is partitioned H times to obtain H component clustering results $\Pi = \{\pi^h\}_{h\in[H]}$. Clustering ensembles use a consensus function τ to combine these components to obtain final clustering partition π^*. Since the clusters returned by a clustering algorithm are not labeled, the first step is to align the clusterings, based on the recognition that similar clusters should contain similar data items. A commonly used scheme is that the pair of clusters with the largest overlapped data items is matched and denoted by the same label. This process is repeated until all the clusters are matched. After that, we need to determine the consensus function τ. The one we used here is selective voting, that is, the clusters of $\mathbf{x}_n$ is determined by the plurality voting result of the clusters assigned to it.

This idea can be easily extended to multiview cases, with the only difference that the algorithms used to obtain the component clusterings are replaced with multiview clustering algorithms. Basically, any multiview clustering algorithms can be used in multiview clustering ensembles (Xie and Sun 2013).

5.6 Other Methods

Multiview Clustering via Structured Sparsity (Wang et al. 2013a) Most of the multiview learning algorithms combines the features by assign one weight for each type of features. However, for many clustering problems, the importance of a feature may vary across different clusters. Thus learning the weights for features with respect to each cluster individually can be a more principled way to fusion data. Wang et al. (2013a) proposed to use joint structured sparsity-inducing norms to perform feature learning and model selection. The basic idea is to regularize the objective of spectral embedding clustering (Nie et al. 2011) with the group sparsity norm (Wang et al. 2012, 2013b) of the weights of the features to enforce the sparsity between different

views. An $l_{1,2}$-norm regularizer is further added to impose the sparsity between all features and non-sparsity between clusters. With these two regularizers, the method can capture the importance of local features, and perform effective clustering.

Multiview Sparse Co-clustering (Sun et al. 2015) A subspace clustering problem can be formulated as the problem of seeking a sparse rank one approximation of the data matrix (Elhamifar and Vidal 2013), where the approximation can be represented by an outer product of two vectors. In the multiview case, multiple data matrices should be approximated simultaneously. One idea is to use a co-regularized objective function to seek the approximations as shown in Sect. 5.3.2. On the other hand, multiview sparse co-clustering uses the proximal alternating linearized minimization (PALM) (Bolte et al. 2014) algorithm to alternatively optimizing each block of the variables constructing the approximation to the data matrix from each view. PALM is an optimization algorithm particularly suitable for solving such problems with multiple blocks of variables. The learned approximations indicate a clustering of the multiview data.

Multiview K-Means Clustering (Cai et al. 2013) K-means clustering is a simple and effective clustering algorithm. Due to its low computational cost and easily parallelized process, K-means clustering is particularly suitable for solving large-scale clustering problems. Cai et al. (2013) proposed to extend K-means clustering to the multiview setting basing on the equivalence of nonnegative matrix factoring (NMF) (Ding et al. 2005) and the relaxed K-means clustering. By optimizing the summation of objective functions of NMF from different views and replacing the F-norm with the $l_{1,2}$-norm which induced from structured sparsity, Cai et al. (2013) obtained a multiview K-means clustering algorithm which is robust to data outliers and more stable. Multiview K-means clustering also exhibits the low computational cost and the easily parallelized process of the single-view K-means.

References

Achlioptas D, McSherry F (2005) On spectral learning of mixtures of distributions. In: Proceedings of the 18th annual conference on learning theory. ACM, Berlin, pp 458–469

Bolte J, Sabach S, Teboulle M (2014) Proximal alternating linearized minimization or nonconvex and nonsmooth problems. Math Program 146(1–2):459–494

Cai X, Nie F, Huang H (2013) Multi-view k-means clustering on big data

Chaudhuri K, Kakade SM, Livescu K, Sridharan K (2009) Multi-view clustering via canonical correlation analysis. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 129–136

Ding C, He X, Simon HD (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of the 5th SIAM international conference on data mining, SIAM, pp 606–610

Elhamifar E, Vidal R (2013) Sparse subspace clustering: algorithm, theory, and applications. IEEE Trans Pattern Anal Mach Intell 35(11):2765–2781

Gao H, Nie F, Li X, Huang H (2015) Multi-view subspace clustering. In: 2015 IEEE international conference on computer vision, pp 4238–4246

Kumar A, Daum H (2011) A co-training approach for multi-view spectral clustering. In: Proceedings of the 28th international conference on machine learning. ACM, pp 393–400

Kumar A, Rai P, Daume H (2011) Co-regularized multi-view spectral clustering. Adv Neural Inf Process Syst 24:1413–1421

Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788

Liu J, Wang C, Gao J, Han J (2013) Multi-view clustering via joint nonnegative matrix factorization. In: Proceedings of the 13th SIAM international conference on data mining, SIAM, pp 252–260

Long B, Yu PS, Zhang ZM (2008) A general model for multiple view unsupervised learning. In: Proceedings of the 8th SIAM international conference on data mining, SIAM, pp 822–833

Nie F, Zeng Z, Tsang IW, Xu D, Zhang C (2011) Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering. IEEE Trans Neural Netw 22(11):1796–1808

Salakhutdinov R, Roweis ST (2003) Adaptive overrelaxed bound optimization methods. In: Proceedings of the 20th international conference on machine learning, pp 664–671

Sun J, Lu J, Xu T, Bi J (2015) Multi-view sparse co-clustering via proximal alternating linearized minimization. In: Proceedings of the 32th international conference on machine learning, pp 757–766

Topchy A, Jain AK, Punch W (2005) Clustering ensembles: models of consensus and weak partitions. IEEE Trans Pattern Anal Mach Intell 27(12):1866–1881

Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416

Wang H, Nie F, Huang H, Risacher SL, Saykin AJ, Shen L, Initiative ADN (2012) Identifying disease sensitive and quantitative trait-relevant biomarkers from multidimensional heterogeneous imaging genetics data via sparse multimodal multitask learning. Bioinformatics 28(12):i127–i136

Wang H, Nie F, Huang H (2013a) Multi-view clustering and feature learning via structured sparsity. In: Proceedings of the 30th international conference on machine learning, pp 352–360

Wang H, Nie F, Huang H, Ding C (2013b) Heterogeneous visual features fusion via sparse multimodal machine. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3097–3102

Xie X, Sun S (2013) Multi-view clustering ensembles. In: Proceedings of the 5th international conference on machine learning and cybernetics, vol 1, pp 51–56

Zhou ZH, Tang W (2006) Clusterer ensemble. Knowl-Based Syst 19(1):77–83

Chapter 6
Multiview Active Learning

Abstract Active learning is proposed based on the fact that manually labeled examples are expensive, thus it picks the most informative points to label so as to improve the learning efficiency. Combined with multiview learning algorithm, it constructs multiple learners to select contention points among different views. In this chapter, we introduce five multiview active learning algorithms as examples. At first, we introduce co-testing, the first algorithm applying active learning to multiview learning, and discuss how to process the contradiction between multiple learners. Bayesian co-training is proposed under the mutual information framework, which considers the unobserved labels as latent variables and marginalizes them out. We focus on multiview multi-learner learning active learning, which introduces the ambiguity of an example to measure its confidence. In the situation that active learning with extremely sparse labeled examples, there is a detailed derivation of CCA in two view. At last, we retell a practical active learning algorithm combined with semi-supervised learning. Besides, there are other methods briefly mentioned at the end of this chapter.

6.1 Introduction

Multiview learning is related to many popular topics in machine learning including active learning. Active learning aims to minimize the effort of labeling by detecting the most informative examples in the domain and asking the user to label them, thus reducing the user's involvement in the data labeling process. In typical methods of multiview active learning, since multiple views provide different features to detect the contention points, it converges quickly to the *target concept* with high confidence. The combination of active learning and multiple views alleviates the burden of labeling training data and increases the efficiency of the learning process.

The concept to be learned is called the *target concept*, which can be written as a function $c : X \to l_1, l_2, ..., l_N$ that classifies any instance $\mathbf{x}$ as a member of the one of the N classes of interest $l_1, l_2, ..., l_N$. The user provides a set of training examples noted as $\langle \mathbf{x}, c(\mathbf{x}) \rangle$ representing an instance $\mathbf{x} \in X$ and its label, $c(\mathbf{x})$. The symbol $\mathscr{L}$ is used to denote the set of labeled training examples, which is also called

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_6

the training set. In selective sampling algorithm, the examples chosen for labeling, namely queries, are chosen from a given working set of unlabeled examples $\mathscr{U}$, in which examples are noted as $\langle \mathbf{x}, ? \rangle$.

Machine learning aims to choose the most appropriate hypothesis $h : X \rightarrow l_1, l_2, ..., l_N$ by an inductive learning algorithm $\mathscr{A}$, so that h is hopefully consistent with the training set L namely $\forall \langle x, c(\mathbf{x}) \rangle \in L, h(\mathbf{x}) = c(\mathbf{x})$. Active learning algorithms have the ability to choose the informative examples in the instance space X and ask the user to label only them. Meanwhile, multiview learning partitions the domain's features to k different views $V_1, V_2, ..., V_k$, which are combined to learn the target concept. For any example $\mathbf{x}$, $V_i(\mathbf{x})$ denotes the description $\mathbf{x}_i$ of $\mathbf{x}$ in V_i.

6.2 Co-testing

Co-testing is the first multiview active learning approach proposed by Muslea et al. (2006), which is a two-step iterative algorithm requiring a few labeled and many unlabeled examples. Unlabeled ones are applied by learned hypotheses through labeled examples and are probably predicted to be contention points between different views. At least one of the learned hypotheses makes a mistake when the predictions from different view disagree. Compared with other active learning algorithms, co-testing relies on the idea of learning from mistakes.

The main iteration of co-testing contains the following two steps. First, as the base learner, algorithm $\mathscr{A}$ is applied to learn a hypothesis h_i from each view through labeled examples in A. Second, it detects contention points $\langle x, ? \rangle$ which are certain unlabeled examples for which different hypotheses predict a different label. Then, it queries one of the contention points by different strategies and moves it to the training set A. These two steps are repeated until convergence. After a number of iterations, an output hypothesis is generated and used for the actual predictions.

Several co-testing based algorithms differ from each other in the style of querying the contention points and creating the output hypothesis. There are three types of query selection strategies: choose randomly in a naive way, choose the least confident of the hypotheses in an aggressive way, or choose evenly to shrink the gap between the highest and the lowest confidence of the hypotheses in the conservative way. The aggressive method achieves high accuracy but suffers from noise, formally,

$$Q = \underset{x \in Contention\,Points}{\arg\max} \min_{i \in \{1,2,...,k\}} Confidence(h_i(x)).$$

In contrast, the conservation strategy can handle noisy domains, that is,

$$Q = \underset{x \in Contention\,Points}{\arg\min} \left(\max_{f \in \{h_1,...,h_k\}} Confidence\,(f(x)) - \min_{g \in \{h_1,...,h_k\}} Confidence\,(g(x)) \right).$$

Also, there are three choices of creating the output hypothesis, that is weighted vote, majority vote and winner-takes-all. The first method weights every hypothesis and combines the vote of each hypothesis, while the second one chooses the output given by most of the hypotheses. The last method chooses the hypothesis which makes the least mistakes as the winner and uses it as the final predictor.

In practice, it is often not the case that each view is sufficient to learn the target concept. A *weakview* is referred to as a view that can learn a more general or more specific concept than the target concept. As an extension of co-testing, contention points are redefined as the unlabeled examples where the strong views predict a different label. The predictions from weak views are ignored because the concept learned from a weak view typically disagrees with the strong views on a large number of unlabeled examples and the mistakes made by a weak view are meaningless to be fixed.

6.3 Bayesian Co-training

As mentioned in the previous chapter, Bayesian co-training is a co-regularization style algorithm for semi-supervised learning (Yu et al. 2011), which considers the unobserved labels as latent variables and marginalizes them out. In this section, we will focus on selecting the best unobserved pairs (samples and views) for sensing based on the mutual information framework to improve the overall classification performance.

For simplicity, we only consider the binary classification using the logistic function as an example. To distinguish between the observed and unobserved pairs (samples and views), we give two notions as $\mathscr{D}_O$ and $\mathscr{D}_U$, respectively. The marginal joint distribution through integrating out $\{\mathbf{f}^c\}$ reduces to

$$p(\mathbf{y}_l, \mathbf{f}^c) = \frac{1}{Z}\psi(\mathbf{f}^c)\prod_{i=1}^{n_l}\psi(y_i, f^c(x_i)),$$

where n_l is the number of labeled examples contained in $\mathbf{y}_l$, $\psi(\mathbf{f}^c)$ is defined as co-training kernel, and $\psi(y_i, f^c(x_i))$ is the logistic function. Thus, the log marginal likelihood of the output $\mathbf{y}_l$ conditioned on input data $\mathbf{X}$ and model parameters Θ is given as below:

$$\begin{aligned}\log p(\mathbf{y}_l|\mathbf{X}, \Theta) &= \log \int p(\mathbf{y}_l|\mathbf{f}^c, \Theta)p(\mathbf{f}^c|\mathbf{X}, \Theta)d\mathbf{f}^c - \log Z \\ &= \log \int \prod_{i=1}^{m}\lambda(y_i f^c(x_i)) \cdot \exp\left\{-\frac{1}{2}(\mathbf{f}^c)^\intercal (K^c)^{-1} f^c\right\} d\mathbf{f}^c - \log Z,\end{aligned}$$

where K^c is the co-training kernel.

In active sensing, the best unobserved pairs (i^*, j^*) are learned by the maximum mutual information criterion,

$$(i^*, j^*) = \arg\max_{\mathbf{x}_i^{(j)} \in \mathscr{D}_U} I(\mathbf{f}^c, \mathbf{x}_i^{(j)}) = \arg\max_{\mathbf{x}_i^{(j)} \in \mathscr{D}_U} \mathbb{E}[\log\det(\Delta_{post}^{x(i,j)})],$$

where the mutual information is calculated between the consensus view function $\mathbf{f}^c$ and the unobserved pair $\mathbf{x}_i^{(j)} \in \mathscr{D}_U$,

$$I(\mathbf{f}^c, \mathbf{x}_i^{(j)}) = \mathbb{E}[H(\mathbf{f}^c)] - \mathbb{E}[H(\mathbf{f}^c|\mathbf{x}_i^{(j)})] = -\frac{1}{2}\log\det(\Delta_{post}) + \frac{1}{2}\mathbb{E}[\log\det(\Delta_{post}^{x(i,j)})].$$

Actually, the mutual information comes from the differential entropy of the consensus view function $\mathbf{f}^c$. By using the Laplace approximation, the posteriori distribution of $\mathbf{f}^c$ is assumed to be a Gaussian distribution

$$\mathscr{N}(\hat{\mathbf{f}_c}, (\Delta_{post})^{-1})$$

with the posteriori precision matrix, $\Delta_{post} = (K^c)^{-1} + \Phi$, where $\hat{\mathbf{f}_c}$ is the maximum a posteriori estimate of $\mathbf{f}^c$. The differential entropy of $\mathbf{f}^c$ is

$$H(\mathbf{f}^c) = -\frac{n}{2}\log(2\pi e) - \frac{1}{2}\log\det(\Delta_{post}).$$

Thus, the mutual information is the expected decrease in entropy of $\mathbf{f}^c$ when $\mathbf{x}_i^{(j)}$ is observed, eliminating the constant term and keeping the posteriori precision matrix.

In order to seek a maximum of the mutual information, we will calculate an approximation of the expectation of the logarithm of $\Delta_{post}^{x(i,j)}$'s matrix determinant as target expectation. Derived from the original posteriori precision matrix, Δ_{post}, after observing one pair $x_i^{(j)}$, it turns to

$$\Delta_{post}^{x(i,j)} = \left((K^c)^{x(i,j)}\right)^{-1} + \Phi,$$

where $(K^c)^{x(i,j)}$ is the new K^c after the new pair is observed. Though it is difficult to calculate the target expectation, one can take advantage of the concavity of $\log\det(\cdot)$ with the inequality, $\mathbb{E}[\log\det(Q)] \leq \log\det(\mathbb{E}[Q])$ and take the upper bound $\log\det(\mathbb{E}[\Delta_{post}^{x(i,j)}])$ as the selection criteria. However, the best pair (i, j) that optimizes the upper bound probably fails to optimize the original target expectation. Thus, direct derivation of expectation is recommended to find the best pair (Yu et al. 2011).

According to the expansion of the one-view information matrix to the full size $n \times n$,

$$A^j(w^j, w^j) = \left(K^j + (\sigma^j)^2 I\right)^{-1}, \text{ and 0 otherwise,}$$

the precision matrix can be derived into

$$\Delta_{post}^{x(i,j)} = (A^j)^{x(i,j)} + \sum_{k \neq j} A^k + \Phi,$$

where $(A^j)^{x(i,j)}$ is the new A^j after the new pair is observed. The new kernel $(A^j)^{x(i,j)}$ is refreshed by recalculating the kernel for the j-th view, K^j,

$$(K^j)^{x(i,j)} = \begin{bmatrix} K^j & b^j \\ (b^j)^\intercal & a^j \end{bmatrix},$$

where $a^j = k_j(x_i^{(j)}, x_i^{(j)}) \in \mathbb{R}$, and $b^j \in \mathbb{R}^{n_j}$ has the l-th entry as $k_j(x_l^{(j)}, x_i^{(j)})$. Then $(A^j)^{x(i,j)}$ changes to

$$\begin{aligned} \left((K^j)^{x(i,j)} + (\sigma^j)^2 I\right)^{-1} &= \begin{bmatrix} K^j + (\sigma^j)^2 I & b^j \\ (b^j)^\intercal & a^j + (\sigma^j)^2 I) \end{bmatrix}^{-1} \\ &= \begin{bmatrix} q^j + s^j q^j b^j (b^j)^\intercal q^j & -s^j q^j b^j \\ -s^j (b^j)^\intercal q^j & s^j \end{bmatrix}, \end{aligned}$$

where $q^j = (K^j + (\sigma^j)^2 I)^{-1}$ and $s^j = \frac{1}{a^j + (\sigma^j)^2 - (b^j)^\intercal q^j b^j}$. With this step, the target expectation is deduced to $\mathbb{E}[s^j]$, $\mathbb{E}[s^j b^j]$ and $\mathbb{E}[s^j b^j (b^j)^\intercal]$, where the expectations are with respect to $p(\mathbf{x}_i^{(j)} | \mathscr{D}_O, \mathbf{y})$.

As we have assumed that the posteriori distribution of $\mathbf{f}^c$ is a Gaussian distribution, then we use a special Gaussian mixture model (GMM) to model the conditional density of the unobserved pairs, $p(\mathbf{x}_i^{(j)} | \mathscr{D}_O, \mathbf{y}_l)$, with which we can calculate the expectations above. When the label y_i is available, for example, $y_i = +1$, we have (Yu et al. 2011)

$$p(\mathbf{x}_i^{(j)} | \mathscr{D}_O, \mathbf{y}_l) = p(\mathbf{x}_i^{(j)} | \mathbf{x}_i^{(O)}, y_i = +1) = \sum_d \pi_d^{+(j)}(\mathbf{x}_i^{(O)}) \cdot \mathscr{N}(\mathbf{x}_i^{(j)} | \mu_d^{+(j)}, \Sigma_d^{+(j)}),$$

where $\pi_d^{+(j)}(\cdot)$ is the mixing weight function, $(\mu^d)^{+(j)}$, and $(\Sigma^d)^{+(j)}$ are the mean and covariance matrix for view j in component d. When the label y_i is not available, we integrate out the labeling uncertainty below,

$$p(\mathbf{x}_i^{(j)} | \mathscr{D}_O, \mathbf{y}_l) = p(\mathbf{x}_i^{(j)} | \mathbf{x}_i^{(O)}) = \sum_{t_i \in \{-1,+1\}} p(y_i = t_i) p(\mathbf{x}_i^{(j)} | \mathbf{x}_i^{(O)}, y_i = t_i).$$

Under this model, all the marginals are still GMMs.

This approach selects the best pair by calculating the mutual information between the consensus view function and the unobserved pair to dig out the informative one. Thus, it performs better in improving the classification performance compared with random sensing.

6.4 Multiple-View Multiple-Learner

Past theoretical and experimental active learning work can be roughly divided into four categories depending on the number of views and learners: multiple views or single view, multiple learners or single learner. Therefore, there are four combinatorial ways to investigate active learning: SVSL, MVSL, SVML and MVML.

In active learning problems, SVSL active learning is the most basic method that uses only one view and requires an effective learner to avoid a strong inductive bias. For example, the expectation-maximization (EM) algorithm can also be considered as an SVSL algorithm. Instead, MVSL active learning uses each view to infer the query, but can be further improved by a certain kind of ensembles. Co-testing is also an active learning algorithm that belongs to MVSL active learning. Benefited from the ensemble technology, SVML active learning combines the results of multiple learners on the same feature set, but still needs improvements when several views are available. Thus, MVML active learning was proposed to overcome the disadvantages above (Zhang and Sun 2010). In this section, we will focus on MVML active learning, which benefits from the advantages of both multiple views and multiple learners.

Similar to the iterative process in co-testing, MVML active learning repeats selecting data to be labeled as follows. At the very beginning, we assume that a small quantity of labeled examples contained in set L and a large quantity of unlabeled examples contained in set U are available. A number of learners $\mathscr{L}_i$ of multiple views are initialized by the small quantity of labeled examples. At the first step of iteration, some unlabeled examples from U are randomly selected and moved to a pool P. Then, every learner will estimate the labels of examples from P in each view and produce different confidence. The algorithm will randomly select several most ambiguous examples on which these views have the largest disagreement. Next, the user will give the correct labels to them and move those newly labeled examples to L. In that case, every learner will be rebuilt again based on the enlarged labeled set L. Then we go back to the first step of the iteration to repeat the process for some times. At the classification stage, the final hypothesis is calculated by the combined results of the learners of these views.

As mentioned above, the MVML active learning algorithm aims to provide good accuracy by incrementally enlarging the labeled set and boosting the learners. To evaluate the uncertainty of binary classification results between multiple views, we introduce the $ambiguity$ of a sample x as follows:

$$ambiguity(x) = \begin{cases} 1 + P_r^1 \log_2(P_r^1) + P_s^2 \log_2(P_s^2), H_1 \neq H_2 \\ -1 - P_r^1 \log_2(P_r^1) - P_s^2 \log_2(P_s^2), H_1 = H_2 \end{cases}$$

where H_1 and H_2 are the outputs made by two views, P_j^i is the confidence that the i-th view V_i considers the predicted example x as class j. Particularly, P_r^1 and P_s^2 denote the confidences of the suggested classes of V_1 and V_2, that is

$$r = \arg\max_j P_j^1,\ s = \arg\max_j P_j^2,$$

which can also be extended to the multi-class case.

Ambiguity completely depends on the disagreement between the views rather than puts all the learners of all views together. The definition of $ambiguity$ is reasonable from two aspects integrating with the function property of $x \log_2(x)$. When $H_1 \neq H_2$, the $ambiguity$ of x is greater than 0 and is an increasing function. In this scenario, the example which has a larger confidence will have a larger $ambiguity$ value, implying the larger disagreement between different views, and it will be considered as the more informative one to be selected. In contrast, when $H_1 = H_2$, the $ambiguity$ of x is less than 0 and is a decreasing function. The example which has a smaller confidence will have a larger $ambiguity$ value, hinting the uncertainty towards classification, and then it will be selected to label on hand. In conclusion, the $ambiguity$ of an example x reflects the priority to be selected whenever the outputs of different views are the same or not.

To measure the disagreement between different learners within each view, selective sampling based on entropy is qualified. Suppose a c-class classification problem contains multiple views $V_1, ..., V_m$ with k learners. n_j^i denotes the number of learners that the i-th view V_i considers the predicted example x as class j. From this, we can get the confidence that all the learners consider the predicted example x as class j in view V_i,

$$P_j^i = \frac{n_j^i + \frac{\varepsilon}{2}}{k + \varepsilon},$$

where ε is a small positive constant ensuring nonzero confidence. Then the $entropy^i$ within view V_i can be deduced by P_j^i as

$$entropy^i(x) = -\sum_{j=1}^{c} P_j^i \log_2(P_j^i).$$

The MVML active learning combines two methods and updates the definition of $ambiguity$ as:

$$ambiguity' = ambiguity + \sum_{i=1}^{m} entropy^i$$

where $ambiguity$ is the disagreement between multiple views, and $entropy^i$ is the within-view disagreement between multiple learners of V_i.

As an ensemble-style approach, the MVML active learning approach is effective owing to the advantages of both multiview learning and ensemble learning. It performs better than the other three kinds of active learning and is flexible to choose different methods for selective sampling.

6.5 Active Learning with Extremely Spare Labeled Examples

In some applications, there are a limited number of labeled examples available, which in turn hampers the ability to construct an effective classifier. In this context, active learning with extremely sparse labeled examples (ALESLE) was proposed based on canonical correlation analysis (CCA) (Sun and Hardoon 2010). ALESLE will work under the multiview setting in high-dimensional spaces with a particular assumption that there is only one example labeled from each class. To ensure the performance of ALESLE, we demand that each view is sufficient for correct classification.

In brief, CCA aims to find basis vectors for two feature sets views so that the correlation between the projections of the feature sets onto these basis vectors are mutually maximized. To illustrate explicitly, we will give the nomenclature used in the coming context. We discuss a two-view binary classification problem within instance space $V = V^1 \times V^2$, where V^1 and V^2 corresponds to two views, respectively. Examples from the space are denoted as $(\langle \mathbf{x}, \mathbf{y} \rangle, z)$, where vector $\mathbf{x}$ and $\mathbf{y}$ are features from spaces V^1 and V^2, and scalar $z \in \mathbb{N}$ is a class label. When the corresponding z is given, $(\langle \mathbf{x}, \mathbf{y} \rangle, z)$ is a labeled example. Particularly, we assume that every class has definitely one labeled example and define them as $(\langle \mathbf{x}_1, \mathbf{y}_1 \rangle, 1)$ and $(\langle \mathbf{x}_2, \mathbf{y}_2 \rangle, -1)$ that are included in set L. In another case that z is not given and $(\langle \mathbf{x}, \mathbf{y} \rangle, z)$ is an unlabeled example, we define an unlabeled set $U = \{(\langle \mathbf{x}_i, \mathbf{y}_i \rangle, z_i) | i = 3, ..., l\}$. The task is to select informative examples from U to be labeled and then induce a classifier for classifying new data.

In CCA, different views are assumed to have close relationship with semantic characteristics of the underlying patterns. In other words, we need to maximize the correlation coefficient between projections $\mathbf{w}_x^\intercal \mathbf{X}$ and $\mathbf{w}_y^\intercal \mathbf{Y}$, where $\mathbf{w}_x$ and $\mathbf{w}_y$ are two basis vectors for feature matrices $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l)$, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_l)$, respectively. Let C_{xy} denote the correlation coefficient of between-set covariance matrix of $\mathbf{X}$ and $\mathbf{Y}$ while C_{xx} and C_{yy} are the coefficients of the within-set covariance matrices of $\mathbf{X}$ and $\mathbf{Y}$, respectively. The objective function for CCA is

$$\max_{\mathbf{w}_x, \mathbf{w}_y} \quad \mathbf{w}_x^\intercal C_{xy} \mathbf{w}_y$$
$$\text{s.t.} \quad \begin{cases} \mathbf{w}_x^\intercal C_{xx} \mathbf{w}_x = 1 \\ \mathbf{w}_y^\intercal C_{yy} \mathbf{w}_y = 1 \end{cases}.$$

The Lagrangian is

$$L(\lambda_x, \lambda_y, \mathbf{w}_x, \mathbf{w}_y) = \mathbf{w}_x^\intercal C_{xy} \mathbf{w}_y - \frac{\lambda_x}{2}(\mathbf{w}_x^\intercal C_{xx} \mathbf{w}_x - 1) - \frac{\lambda_y}{2}(\mathbf{w}_y^\intercal C_{yy} \mathbf{w}_y - 1).$$

Let the derivatives to $\mathbf{w}_x$ and $\mathbf{w}_y$ be equal to zero, that is

$$\frac{\partial L}{\partial \mathbf{w}_x} = C_{xy}\mathbf{w}_y - \lambda_x C_{xx}\mathbf{w}_x = 0$$
$$\frac{\partial L}{\partial \mathbf{w}_y} = C_{yx}\mathbf{w}_y - \lambda_y C_{yy}\mathbf{w}_y = 0.$$

According to the constraint condition, we introduce λ and obtain

$$\lambda = \lambda_x = \lambda_y = \mathbf{w}_x^{\mathsf{T}} C_{xy}\mathbf{w}_y,$$

implying that the objective function is equal to the Lagrangian coefficient, and

$$C_{xx}^{-1}C_{xy}\mathbf{w}_y = \lambda_x \mathbf{w}_x$$
$$C_{yy}^{-1}C_{yx}\mathbf{w}_y = \lambda_y \mathbf{w}_y.$$

So that we can write them in form of matrices

$$\begin{pmatrix} C_{xx}^{-1} & 0 \\ 0 & C_{yy}^{-1} \end{pmatrix} \begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix}.$$

Denote matrices $\mathbf{A}$, $\mathbf{B}$ and vector $\mathbf{w}$ as

$$\mathbf{B} = \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix}, \mathbf{w} = \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix}.$$

We obtain a generalized eigenvalue equation

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{w} = \lambda \mathbf{w}.$$

Or from the derivatives, we can also obtain a similar form to solve out λ and $\mathbf{w}_x$

$$C_{xx}^{-1}C_{xy}C_{yy}^{-1}C_{yx}\mathbf{w}_x = \lambda^2 \mathbf{w}_x$$

with

$$\mathbf{w}_y = \frac{1}{\lambda} C_{yy}^{-1}C_{yx}\mathbf{w}_x$$

to solve out $\mathbf{w}_y$. After that, we normalize these basis vectors to unit vectors.

Note that λ equals the initial objective function for CCA. So the largest eigenvalue λ^* with corresponding vector pair $(\mathbf{w}_x^*, \mathbf{w}_y^*)$ is exactly what we want. In practical applications, we preserve a series of vector pairs to reflect different correlations. For instance, we selects the larger m eigenvalues λ_j for $j = 1, ..., m$ and the corresponding projection vectors, with each of which an example $\langle \mathbf{x}, \mathbf{y} \rangle$ is projected into

$\langle P_j(\mathbf{x}), P_j(\mathbf{y})\rangle$. To calculate the similarity between one labeled example and another unlabeled example in the j-th projection, ALESLE uses

$$S_{j,i}^k = \exp\{-(P_j(\mathbf{x}_i) - P_j(\mathbf{x}_k))^2\} + \exp\{-(P_j(\mathbf{y}_i) - P_j(\mathbf{y}_k))^2\}$$

where $\langle \mathbf{x}_k, \mathbf{y}_k\rangle$ for $k = 1, 2$ is the labeled one and $\langle \mathbf{x}_i, \mathbf{y}_i\rangle$ from U. Summing up all projection we obtain the total similarity as

$$\rho_i^k = \sum_{j=1}^{m} \lambda_j S_{j,i}^k$$

where the value of λ_j reflects the weight of similarity in the j-th projection. One unlabeled example has different similarity towards every labeled example, and so the following values are calculated to describe the certainty of classification:

$$d_i = \|\rho_i^1 - \rho_i^2\|$$

Unlabeled examples with small d_i values tend to lie at the boundary between these two classes and are valuable to be selected automatically and labeled manually. After labeling, these examples will be moved to the labeled pool of examples L from U.

This method makes the best of limited labeled examples and performs well in high-dimensional spaces. But it needs to work under the assumption that there exist multiple views that are sufficient for correct classification and still has room for progress with more relaxed assumptions.

6.6 Combining Active Learning with Semi-supervising Learning

The situation that training sets contain a few labeled examples and a large quantity of unlabeled examples has been common in many real-world applications. There are two main machine learning paradigms for tackling this problem: semi-supervised learning and active learning. Thus it is a natural idea that combining these two paradigms to better handle the data and perform learning. In this section, we will introduce an approach, semi-supervised active image retrieval (SSAIR) that combines active learning with semi-supervised learning in the multiview learning settings (Zhou et al. 2004).

SSAIR was originally proposed for content-based image retrieval (CBIR), which aims to pick up relevant images from the database when the queries are given as the labeled data. The method will rank every image in a database such that images with higher ranks are more relevant to the user query. Due to the fact that the number of the labeled data is small in CBIR, SSAIR utilized the techniques used in semi-supervised learning. Specifically, SSAIR trains two learners with co-training to rank

the images. Besides, by active learning, images without ensured relevance to the quires are selected for user labeling.

To illustrate the method clearly, let $\mathscr{U}$ and $\mathscr{L}$ denote the set of unlabeled examples and labeled examples, respectively. Then we have $\mathscr{L} = \mathscr{P} \cup \mathscr{N}$, where $\mathscr{P}$ and $\mathscr{N}$ denote the set of positive and negative labeled examples, respectively. Initially, $\mathscr{U}$ is the whole database, $\mathscr{P}$ is the set of the user query, and $\mathscr{N}$ is empty.

The method utilizes two learners to help classify the examples from the database. Depend on $\mathscr{L}$, the two learners give every example from $\mathscr{U}$ a rank ranging from -1 to 1 to measure the closeness to the query. Examples with larger rank are closer to the query, and the ones with a bigger absolute value of rank are more confident in their judgments. According to the rank information, the first learner will pass on the most relevant/irrelevant examples from $\mathscr{U}$ as the newly labeled positive/negative examples to the second learner, and vice versa in a co-training style. Both the learners will be retrained with the newly provided labeled training sets and give every example in $\mathscr{U}$ a rank. After co-training, examples with the smallest absolute value of rank will be put into the set $pool$, while the largest ones will be put into the set $result$. The set $result$ includes the most relevant examples towards the query, while the set $pool$ contains the most informative examples. The user can choose examples from the set $pool$ to label, and move them to $\mathscr{L}$ ($\mathscr{P}$ or $\mathscr{N}$). After obtaining the enlarged labeled set $\mathscr{L}$, it will go back to the first step of the iteration and retrain the two learners.

To calculate the relevance between two examples, the similarity Sim_i in the i-th view ($i = 1, 2$) between two feature vectors, $\mathbf{x}$ and $\mathbf{y}$, is defined as the reciprocal of the Minkowski distance with orders p_i (Zhou et al. 2004),

$$Sim_i(\mathbf{x}, \mathbf{y}) = \left(\left(\sum_{j=1}^{d} |\mathbf{x}_j - \mathbf{y}_j|^{p_i} \right)^{\frac{1}{p_i}} + \varepsilon \right)^{-1},$$

where ε is a small positive constant ensuring nonzero denominator. Feature $\mathbf{x}$ is trained in the training set T_i as

$$T_i(\mathbf{x}, P_i, N_i) = \frac{1}{Z_{norm}} \left(\sum_{\mathbf{y} \in P_i} \frac{Sim_i(\mathbf{x}, \mathbf{y})}{|P_i| + \varepsilon} + \sum_{\mathbf{z} \in N_i} \frac{Sim_i(\mathbf{x}, \mathbf{z})}{|N_i| + \varepsilon} \right)$$

where Z_{norm} is used to normalize the results to $(-1, 1)$. The rank of $\mathbf{x}$ will be measured by summing up T_i for two learners. To ensure the learners' diversity, there are flexible approaches such as setting different p_i for two learners, and using different supervised learning algorithms to partition the instance space into series of classes.

Consider that usually only a small set of examples will be relevant to a particular query while most are irrelevant. From the rank information, there may exist more negative examples than positive ones in reality. In this scenario, we identify the k-nearest neighboring unlabeled examples for each negative example and compute their averages to substitute the original negative example.

6.7 Other Methods

Co-EMT (Muslea et al. 2002): Co-EMT is a robust multiview learning method for its combination of semi-supervised learning and active learning. It makes improvements in the original co-testing algorithm through running co-EM on both labeled and unlabeled examples rather than establishing hypotheses-based solely on labeled examples. Then it detects a set of contention points by the combined prediction of hypotheses with least confidence, and asks the user to label it. The newly labeled examples will be moved from the unlabeled set to the labeled set. The learning process and labeling process will be repeated several times. Finally, it combines the prediction of different hypotheses. Since co-EMT applies co-EM to learn hypotheses with unlabeled data, it is more accurate than the original co-testing algorithm.

References

Muslea I, Minton S, Knoblock CA (2002) Active + semi-supervised learning = robust multi-view learning. In: Proceedings of the 9th international conference on machine learning, pp 435–442

Muslea I, Minton S, Knoblock CA (2006) Active learning with multiple views. J Artif Intell Res 27(1):203–233

Sun S, Hardoon DR (2010) Active learning with extremely sparse labeled examples. Neurocomputing 73(16–18):2980–2988

Yu S, Krishnapuram B, Rosales R, Rao RB (2011) Bayesian co-training. J Mach Learn Res 12(9):2649–2680

Zhang Q, Sun S (2010) Multiple-view multiple-learner active learning. Pattern Recognit 43(9):3113–3119

Zhou ZH, Chen KJ, Jiang Y (2004) Exploiting unlabeled data in content-based image retrieval. In: Proceedings of the 15th European conference on machine learning. Springer, pp 525–536

Chapter 7
Multiview Transfer Learning and Multitask Learning

Abstract Transfer learning is proposed to transfer the learned knowledge from source domains to target domains where the target ones own fewer training data. Multitask learning learns multiple tasks simultaneously and makes use of the relationship among these tasks. Both of these learning methods can combine with the multiview learning, which exploits the information from the consistency of diverse views. In this chapter, we introduce four multiview transfer learning methods and three multiview multitask learning methods. We review research on multiview transfer learning under the large margin framework, discuss multiview discriminant transfer learning in detail, and introduce how to adapt Adaboost into multiview transfer learning. Three multiview multitask learning methods concentrate on the shared structures between tasks and views. The most natural way is to represent the relationships based on the bipartite graph and use an iterative algorithm to optimize its objective function. Another method constructs additional regularization function to ensure the view consistency. In general, convex shared structure learning algorithm provides structure parameters to share information. Besides, we introduce other methods; as supplements, where multi-transfer, multitask multiview discriminant analysis, and clustering are briefly mentioned.

7.1 Introduction

For many classical machine learning methods, a major assumption needs to be met is that training and test data must be in the same feature space and have the same distribution. In many real-world applications, this condition is difficult to satisfy (Pan et al. 2010). For example, data from the domain of interest are expensive, while sufficient training data from another domain are available. To handle this problem, transfer learning is proposed to transfer the learned knowledge from one domain to others with fewer training data. Multitask learning is a closely related learning framework which learns multiple tasks simultaneously, making use of the relationship among different tasks. In this chapter, we introduce both transfer learning and multitask learning and their extensions to the multiview scenario and show their difference.

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_7

To distinguish a "domain" and a "task", we give their notations below. A domain is noted as $\mathscr{D}$ consisting of a feature space $\mathscr{X}$ and a marginal probability distribution $P(X)$, where an example $\mathbf{x}$ is a D-dimensional vector in space $\mathscr{X}$. A task is noted as $\mathscr{T}$ consisting of a label space $\mathscr{Y}$ and an objective predictive function $f(\cdot)$, which can be written as $P(y|x)$ if predicting the corresponding label of an instance $\mathbf{x}$, where $\mathbf{x} \in X$ and $y \in \mathscr{Y}$. Transfer learning aims to transfer specific learned knowledge in a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_S$ to a target domain $\mathscr{D}_T$ and a learning task $\mathscr{T}_T$ without the traditional assumption that $\mathscr{D}_S = \mathscr{D}_T$ and $\mathscr{T}_S = \mathscr{T}_T$. In such a case, the feature spaces, marginal probability distributions or the label spaces between two domains will be different.

Based on the different setting of source and target domains and tasks, we can categorize transfer learning into many types. Multitask learning is related to labeled inductive transfer learning, where labels of source and target domains are available but the target task is different from the source task. In contrast to transfer learning that cares most about the target task, multitask learning tries to learn all of the source and target tasks simultaneously in a symmetric way.

In this chapter, we integrate the theory of multiview learning into transfer learning and multitask learning, and introduce several multiview transfer learning and multitask learning methods.

7.2 Multiview Transfer Learning with a Large Margin

Multiview transfer learning is designed for transferring knowledge from one domain to another, each of which is of multiple views. This learning methodology can make full use of labeled data from source tasks to help train models in target tasks. The large margin principle is a widely used principle to construct classifiers, and has been applied to transfer learning as well. However, most large margin transfer learning methods do not exploit the multiple views of data.

In this section, we introduce multiview transfer learning with a large margin (MvTL-LM) (Zhang et al. 2011), which considers the multiview setting, models the difference of data distribution between the source domain and the target domain, and uses weighted labeled data from the source domain to construct a large margin classifier for the target domain.

We first give some notations. Suppose we are given a set of labeled source domain examples from V views, $\mathscr{D}_S = \{(\mathbf{x}_i^j, y_i^S)\}_{i=1,\ldots,D_S, j=1,2,\ldots,V}$, where $\mathbf{x}_i^j$ is the i-th example in view j with its class label y_i^S in the source domain. We also have a set of unlabeled target domain examples, $\mathscr{D}_T = \{\mathbf{z}_i^j\}_{i=1,\ldots,D_T, j=1,2,\ldots,V}$, where $\mathbf{z}_i^j$ is the i-th example in view j with its class label y_i^S in the target domain.

MvTL-LM makes linear classifier assumptions for all the views. These classifiers are parameterized by weight vectors $\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^V$, which are obtained by minimizing the following optimization problem:

$$\min_{\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^V} \sum_{p=1}^{V} \gamma_p \Omega(\mathbf{w}^p) + \sum_{p=1}^{V} C_p R(P_T, l(\mathbf{x}^p, y, \mathbf{w}^p)) + \sum_{p=1}^{V}\sum_{q=1}^{V} C_{p,q} R_c(P_T^{p,q}, l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q)),$$

where γ_p, C_p and $C_{p,q}$ are nonnegative hyperparameters that balance the relative importance of three terms (Zhang et al. 2011). Among these terms, $\Omega(\mathbf{w}^p)$ is the regularization term defined on the pth view weight vector $\mathbf{w}^p$, $R(P_T, l(\mathbf{x}^p, y, \mathbf{w}^p))$ is the expected classification loss with respect to the data distribution P_T of the target domain examples, measuring the deviations between the true labels and the predicted labels based on the pth view. $R_c(P_T^{p,q}, l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q))$ is the expected classification squared loss with respect to the joint distribution of the pth and the qth views in the target domain. $l(\mathbf{x}^p, y, \mathbf{w}^p)$ and $l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q)$ are the classification loss and the squared loss, respectively. Correspondingly, $R(P_T, l(\mathbf{x}^p, y, \mathbf{w}^p))$ and $R_c(P_T^{p,q}, l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q))$ are called loss term and consistency term.

Particularly, when $C_{p,q}$ equals zero, the objective function above is equivalent to training a large margin classifier on each view independently. When $C_{p,q}$ does not equal zero, MvTL-LM tries to minimize the objective function and obtain final classifier as the average of the large margin classifiers on all the views calculated as

$$f(\mathbf{x}) = \frac{1}{V}\sum_{p=1}^{V} (\mathbf{w}^p)^\intercal \mathbf{x}^p.$$

The loss term actually is an expectation function of classification loss, which can be written as

$$\begin{aligned} R(P_T, l(\mathbf{x}^p, y, \mathbf{w}^p)) &= E_{P_T}[l(\mathbf{x}^p, y, \mathbf{w}^p)] \\ &= E_{P_S}[\beta(\mathbf{x}) l(\mathbf{x}^p, y, \mathbf{w}^p)] \\ &\approx \frac{1}{D_S}\sum_{i=1}^{D_S} \beta(\mathbf{x}_i) l(\mathbf{x}_i^p, y_i, \mathbf{w}^p), \end{aligned}$$

where $\beta(\mathbf{x})$ is the weight function defined as the ratio between distributions of the source domain and the target domain, i.e.,

$$\beta(\mathbf{x}) = \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})},$$

with which the classification error for the target domain can be estimated without labeling.

Similar to the loss term, the consistency term is also an expectation function of the consistency loss function, which penalizes the deviations between the output of the $\mathbf{x}^p$ and $\mathbf{x}^q$ under the classifiers $\mathbf{w}^p$ and $\mathbf{w}^q$ and can be written as

$$\begin{aligned} &R_c(P_T^{p,q}, l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q)) = E_{P_T^{p,q}}[l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q)] \\ \approx &\frac{1}{D_S + D_T}\left(\sum_{i=1}^{D_S}\beta(\mathbf{x}_i) l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q) + \sum_{i=1}^{D_T} l_c(\mathbf{x}^p, \mathbf{x}^q, \mathbf{w}^p, \mathbf{w}^q)\right). \end{aligned}$$

The consistency term regulates the consistency between D_S examples on the source domain and D_T examples on the target domain. Besides, through the consistency term, MvTL-LM exploits multiview learning by calculating the expectation with respect to the joint distribution of the pth and the qth views, which performs better than single-view approach.

In the case of two views, i.e., $V = 2$, one can use hinge loss and squared loss to define the loss term and the consistency term, respectively, giving

$$\begin{aligned} \min_{\mathbf{w}^1, \mathbf{w}^2} \quad &\sum_{p=1}^{2}\frac{\gamma_p}{2}||\mathbf{w}^p||^2 + \sum_{p=1}^{2} C_p \sum_{i=1}^{D_S}\beta(\mathbf{x}_i)\xi_i^p \\ &+C_{1,2}\left(\sum_{i=1}^{D_S}\beta(\mathbf{x}_i)||(\mathbf{w}^1)^\intercal \mathbf{x}_i^1 - (\mathbf{w}^2)^\intercal \mathbf{x}_i^2||^2 + \sum_{i=1}^{D_T}||(\mathbf{w}^1)^\intercal \mathbf{z}_i^1 - (\mathbf{w}^2)^\intercal \mathbf{z}_i^2||^2\right) \\ s.t. \quad &\forall i \in \{1, 2, \ldots, D_S\}, \quad y_i(\mathbf{w}^1)^\intercal \mathbf{x}_i^1 \geq 1 - \xi_i^1, \quad y_i(\mathbf{w}^2)^\intercal \mathbf{x}_i^2 \geq 1 - \xi_i^2, \end{aligned}$$

where the parameters C_p and $C_{1,2}$ in front of two terms are redefined by absorbing the scaling constants $\frac{1}{D_S}$ and $\frac{1}{D_S+D_T}$ (Zhang et al. 2011).

The MvTL-LM approach constructs a large margin classifier for the target domain with weighted labeled data from the source domain. To ensure the classification consistency between different views, it employs both the unlabeled data from the target domain and labeled data from the source domain, taking advantages of both multiview learning and transfer learning.

7.3 Multiview Discriminant Transfer Learning

As transfer learning allows that training and test data sets have different feature spaces, multiview discriminant transfer learning (MDT) (Yang and Gao 2013) focuses on mining the correlations between views together with the domain distance measure to improve transfer on the feature level. The idea of MDT comes from the classical framework of Fisher discriminant analysis (FDA) and its two-view extension FDA2. The goal of MDT is to extend FDA2 by considering the domain discrepancy and enhancing the view consistency under a similar framework to com-

promise transfer learning and multiview learning. To find the optimal discriminant weight vectors for each view, it maximizes the correlations between the projections of the data onto these weight vectors in two views and minimizes the discrepancies of domains and views.

Given n labeled samples from the source domain $\mathscr{D}_S = \{(\mathbf{x}_i^1, \mathbf{x}_i^2, y_i)\}_{i=1,\ldots,n}$ and m unlabeled samples from the target domain $\mathscr{D}_T = \{(\mathbf{x}_i^1, \mathbf{x}_i^2, ?)\}_{i=n+1,\ldots,n+m}$, where $\mathbf{x}_i^j$ is the ith example in the jth view, $y_i \in \{-1, 1\}$ is the class label with n^+ and n^- being the number of positive and negative instances in the source domain, respectively. The class labels compose a vector denoted as $\mathbf{y} = \{y_1, \ldots, y_n\}^\intercal$. To map the instances from the original feature space to a reproducing kernel Hilbert space (RKHS), it introduces $\phi(\cdot)$ as a feature mapping function, with which data matrix can be written as $\mathbf{X}_S^j = (\phi(\mathbf{x}_1^j), \ldots, \phi(\mathbf{x}_n^j))^\intercal$, $\mathbf{X}_T^j = (\phi(\mathbf{x}_{n+1}^j), \ldots, \phi(\mathbf{x}_{n+m}^j))^\intercal$, and $\mathbf{X}^j = ((\mathbf{X}_S^j)^\intercal, (\mathbf{X}_T^j)^\intercal)^\intercal$.

The basic framework of FDA2 (Diethe et al. 2008) is similar with that of canonical correlation analysis (CCA). It seeks the weight vectors $\mathbf{w}^1, \mathbf{w}^2$ in two views by maximizing

$$\max_{\mathbf{w}^1, \mathbf{w}^2} \frac{(\mathbf{w}^1)^\intercal \mathbf{M}_{1,2} \mathbf{w}^2}{\sqrt{(\mathbf{w}^1)^\intercal \mathbf{M}_1 \mathbf{w}^1} \cdot \sqrt{(\mathbf{w}^2)^\intercal \mathbf{M}_2 \mathbf{w}^2}}$$

where

$$\mathbf{M}_{1,2} = (\mathbf{X}_S^1)^\intercal \mathbf{y}\mathbf{y}^\intercal \mathbf{X}_S^2,$$
$$\mathbf{M}_j = \frac{1}{n} \sum_{i=1}^{n} (\phi(\mathbf{x}_i^j) - \boldsymbol{\mu}_j)(\phi(\mathbf{x}_i^j) - \boldsymbol{\mu}_j)^\intercal,$$

where $\boldsymbol{\mu}_j$ is the mean of the source data in the jth view with $j = 1, 2$. Compared with two-view CCA, there exists the numerator reflecting the between-class distance of two views, which should be maximized. To simplify the objective function, it can be rewritten as an equivalent Rayleigh quotient

$$r = \frac{\boldsymbol{\xi}^\intercal \mathbf{Q}_{1,2} \boldsymbol{\xi}}{\boldsymbol{\xi}^\intercal \mathbf{P} \boldsymbol{\xi}},$$

where

$$\mathbf{Q}_{1,2} = \begin{pmatrix} 0 & \mathbf{M}_{1,2} \\ \mathbf{M}_{1,2}^\intercal & 0 \end{pmatrix}, \mathbf{P} = \begin{pmatrix} \mathbf{M}_1^\intercal & 0 \\ 0 & \mathbf{M}_2 \end{pmatrix}, \boldsymbol{\xi} = \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \end{pmatrix},$$

Incorporating the framework of FDA2, MDT reconstructs its objective function as

$$r = \frac{\boldsymbol{\xi}^\intercal \mathbf{Q} \boldsymbol{\xi}}{\boldsymbol{\xi}^\intercal \mathbf{P} \boldsymbol{\xi}},$$

where

$$\mathbf{Q} = \mathbf{Q}_{1,2} - c_1 \mathbf{Q}_d - c_2 \mathbf{Q}_c$$

is proposed to measure domain discrepancy and view consistency with additional trade-off coefficients. The domain discrepancy term denoted as $\mathbf{Q}_d$ is composed of maximum mean discrepancy (MMD) between the source domain and the target domain from two views. And the view consistency term denoted as $\mathbf{Q}_c$ calculates the distance between two views by enumerating every instance.

One can find the optimal weight vectors by transforming the Rayleigh quotient to the generalized eigenvalue problem as

$$\mathbf{Q}\boldsymbol{\xi} = \lambda \mathbf{P}\boldsymbol{\xi}$$

where λ is the eigenvalue, and $\boldsymbol{\xi}$ is the eigenvector. Thus, the eigenvectors corresponding to the largest eigenvalues are the optimal weight vectors resulting in maximal Rayleigh quotient, with which the prediction function can be obtained as

$$f(\mathbf{x}_i^1, \mathbf{x}_i^2) = \sum_{j=1}^{2} (\mathbf{w}^j)^\intercal \phi(\mathbf{x}_i^j) - b^j,$$

where b_j is the middle boundary of source positive and negative instances satisfying $(w^j)^\intercal \mu_+^j - b^j = b^j - (w^j)^\intercal \mu_-^j$ where μ_+^j and μ_-^j are the means of source positive and negative instances, respectively.

During training, it employs co-training by solving the generalized eigenvalue problem, predicting the target instance simultaneously and moving several most confident instances from the target domain to the source domain. It repeats for sometime until convergence.

The MDT algorithm is well designed for measuring both the discrepancy of domains and the consistency of different views by heuristic reconstruction of FDA2. Yang and Gao (2013) also gave the reason why the proposed method is not applicable under some situations through theoretical analysis.

7.4 Multiview Transfer Learning with Adaboost

In the previous sections, we have introduced how to integrate both multiview learning and transfer learning into existed frameworks. In this section, we will illustrate how to utilize the idea of ensemble learning to perform multiview transfer learning. In particular, we will illustrate how to use multiview learning methods under the framework of Adaboost as an example. Multiview transfer learning with Adaboost (Mv-TLAdaboost) learns two weak learners from the source domain in different views, which facilitate the classification tasks.

The situation that the learned knowledge from the unitary source domain undermines a related performance in the target domain is called negative transfer (Perkins et al. 1992). Although Mv-TLAdaboost uses diverse views, it may cause negative transfer because only a single source domain is adopted. Negative transfer typically

occurs in the early stages of learning a new domain without enough sufficient information. It performs worse if the data is from the single source domain. To overcome this problem, one can mix with multisource learning to enrich source domains.

In the following subsections, we will introduce the basic framework of Adaboost algorithm in Sect. 7.4.1, combine it with multiview transfer learning in Sect. 7.4.2 and overcome negative transfer through multisource learning in Sect. 7.4.3.

7.4.1 Adaboost

Mv-TLAdaboost is a general supervised learning technique, which is proposed with the similar framework of Adaboost. Thus, we first introduce the basic of Adaboost. The traditional framework of Adaboost deals with a set of L labeled examples $\mathscr{L} = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L}$. Adaboost requires an iterative procedure, during which it maintains a distribution P over the L examples, as well as a weight vector $\mathbf{w} = [w_1, \ldots, w_L]^\intercal$. Initially, P is assumed to be a discrete uniform distribution, thus $w_i = \frac{1}{L}, i = 1, 2, \ldots, L$. Given the number of iteration, T, Adaboost begins with computing P through regularizing $\mathbf{w}$,

$$P = \frac{\mathbf{w}}{\sum_{i=1}^{L} w_i}.$$

Then it forms a weak learner h, namely a hypothesis

$$h : \mathscr{L} \to \{0, 1\},$$

and calculates the following weighted 0–1 loss ε:

$$\varepsilon = \sum_{i=1}^{L} p_i |h(\mathbf{x}_i) - y_i|.$$

To update the weight vector credibly, it introduces parameter $\beta = \frac{\varepsilon}{1-\varepsilon}$, then let

$$w_i^{t+1} = w_i^t \beta^{1-|h(\mathbf{x}_i)-y_i|}$$

where w_i^t is the ith element of vector $\mathbf{w}$ in the t-th iteration. After that, it goes back to the step of computing distribution P and repeats these steps for T times. In the end, it generates the hypothesis $h_f(\mathbf{x})$ by combining T weak learners as below

$$h_f(\mathbf{x}) = \begin{cases} 1, & \text{if } \frac{\sum_{t=1}^{\intercal} \log(1/\beta_t) h_t(\mathbf{x})}{\sum_{t=1}^{\intercal} \log 1/\beta_t} \geq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases},$$

where β_t and h_t are the values of β and h in the tth iteration.

7.4.2 Multiview Transfer Learning with Adaboost

So far, we introduce the basic of Adaboost in Sect. 7.4.1. In this subsection, we will show how to adapt Adaboost to transfer learning, with the idea of multiview learning (Mv-TLAdaboost) (Xu and Sun 2011). Due to the additional domains from the transfer learning setting, some modifications to the notations need to be made. We use $\mathscr{L}_T = \{(\mathbf{x}_l, y_l)\}_{l=1,\ldots,L_T}$ to denote the set of L_T labeled examples from target tasks, aggrandizes $\mathscr{L}_S = \{(\mathbf{x}_{L_T+l}, y_{L_T+l})\}_{l=1,\ldots,L_S}$ as the set of L_S labeled examples from source tasks, and enlarges $\mathbf{w}$ as $[w_1, \ldots, w_{L_T}, w_{L_T+1}, \ldots, w_{L_T+L_S}]^\intercal$ where w_i is the weight of example with index i. Thus, P becomes the distribution over $L_T + L_S$ examples, where

$$P = \frac{\mathbf{w}}{\sum_{i=1}^{L_T+L_S} w_i}.$$

Inspired by multiview learning, all the features are divided into two views. Mv-TLAdaboost constructs two weak learners from the two views:

$$h = (h^1, h^2).$$

Therefore, the error of h is replaced by

$$\varepsilon = \sum_{i=1}^{L_T} p_i \{\max\{|h^1(\mathbf{x}_i^1) - y_i|, |h^2(\mathbf{x}_i^2) - y_i|\}\}$$

and the value of ε is required to be less than $1/2$. To avoid the inconsistence between two views, it calculates the percentage of the examples with the same prediction from two weak learners:

$$agreement = 1 - \frac{\sum_{i=1}^{L_T+L_S} |h^1(\mathbf{x}_i^1) - h^2(\mathbf{x}_i^2)|}{L_T + L_S},$$

then rescales the error ε to $\varepsilon \cdot agreement$ and sets

$$\beta = \frac{\varepsilon \cdot agreement}{1 - \varepsilon \cdot agreement}.$$

However, in transfer learning, there are usually more examples from the source task than those from the target task, which may lead to a final weak learner inclining to the source task, ignoring the characteristics of the target task. Consequently, it is necessary to make a difference between data from two tasks in every iteration. With the development of Mv-TLAdaboost, the difference will come to fading resulting

from the integration of them. To support this idea, it introduces the ratio of overall accuracy rate Acc^{S+T} to the partial accuracy rate Acc^{T} noted as η and

$$\eta = \frac{Acc^{S+T}}{Acc^{T}}.$$

The calculation of Acc^{S+T} and Acc^{T} depend on the distributions over $\mathscr{L}_T \cup \mathscr{L}_S$ and $\mathscr{L}_T$, respectively. To calculate the distribution of $\mathscr{L}_T$, it needs R as

$$R = \frac{(w_1, \ldots, w_{L_T})}{\sum_{j=1}^{L_T} w_j},$$

so that Acc^{T} can be written as

$$Acc^{T} = 1 - \sigma,$$

where

$$\sigma = \sum_{i=1}^{L_T} r_i \{\max\{|h^1(\mathbf{x}_i^1) - y_i|, |h^2(\mathbf{x}_i^2) - y_i|\}\}.$$

Similarly, Acc^{S+T} is written as

$$Acc^{S+T} = 1 - \varepsilon.$$

Because of the ambiguity according to data set X, the classification accuracy of it will be less than the one of overall data set, resulting in $\eta \geq 1$. And one hopes the value of η will gradually reach 1 as a result of the balance between partial and overall accuracy. Then, Mv-TLAdaboost can update the weight vector according to

$$w_i^{t+1} = \begin{cases} w_i^t (\beta_t \eta_t)^{-\max\{|h_t^1(\mathbf{x}_i^1) - y_i|, |h_t^2(\mathbf{x}_i^2) - y_i|\}}, & i = 1, \ldots, L_T \\ w_i^t \alpha^{\max\{|h_t^1(\mathbf{x}_i^1) - y_i|, |h_t^2(\mathbf{x}_i^2) - y_i|\}}, & \text{otherwise} \end{cases},$$

where β_t, η_t, h_t^1, h_t^2 is the values of β, η, h^1, h^2 in the tth iteration and α is an hyperparameter defined as

$$\alpha = \frac{1}{1 + \sqrt{2 \ln L_S / T}}.$$

Similar to Adaboost, the iteration then come back to the step of computing distribution P and repeats for T times. Finally, it obtains a classifier

$$h_f(\mathbf{x}^1, \mathbf{x}^2) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} h_t^1(\mathbf{x}^1) + h_t^2(\mathbf{x}^2) \geq T, \\ 0, & \text{otherwise.} \end{cases}$$

As an adaption of Adaboost to transfer learning, Mv-TLAdaboost emphasizes that the target domain $\mathscr{L}_T$ is more representative than the source domain $\mathscr{L}_S$. Thus, it increases the weight of the wrongly classified examples from $\mathscr{L}_T$ while keeps that of the others. With the incorporation of the idea of multiview learning, Mv-TLAdaboost outperforms some Adaboost-based transfer learning methods and shows excellent performance.

7.4.3 *Multisource Transfer Learning with Multiview Adaboost*

Due to the difference between the distributions of target tasks and source tasks, not all of the knowledge from source tasks can be reused in the target task. To avoid negative transfer caused by some contradictory information, multisource transfer learning with multiview Adaboost (MsTL-MvAdaboost) (Xu and Sun 2012) integrates multisource learning into Mv-TLAdaboost, with which it can prevent the negative transfer and promote the effectiveness of transfer learning. In this setting, it assumes that multisource tasks can be obtained simultaneously.

To find useful data among multiple sources in transfer learning, it employs Adaboost algorithm by voting on every data point. Assume there are M data sets $\mathscr{S}_1, \ldots, \mathscr{S}_M$ from different sources with $\mathscr{S}_k = \{(\mathbf{x}_l^k, y_l^k)\}_{l=1,\ldots,S_k}$, where S_k indicates the size of the k-th source data set, $k = 1, \ldots, M$. The target data set $\mathscr{L}_T$ is denoted as $\mathscr{L}_T = \{(\mathbf{x}_l^{target}, y_l)\}_{l=1,\ldots,L_T}$. The basic flow of MsTL-MvAdaboost algorithm is the same as that of Mv-TLAdaboost algorithm. A set of inner iteration is inserted into the progress before the computation of $agreement$, expecting to calculate the minimal error among multiple sources. The error of the weaker learners from the k-th source domain in two views, h^{k1} and h^{k2}, is computed as

$$\varepsilon^k = \sum_{i=1}^{L_T} p_i^k \{\max\{|h^{k1}(\mathbf{x}_i^{target1}) - y_i|, |h^{k2}(\mathbf{x}_i^{target2}) - y_i|\}\},$$

where $\mathbf{x}_i^{target1}$ and $\mathbf{x}_i^{target2}$ is the i-th example from target domain in two views, respectively, P is the overall distribution including M source data sets as

$$P = \frac{\mathbf{w}}{\sum_{i=1}^{L_T+S_1+\cdots+S_M} w_i}.$$

By numerating the errors in Q source domains, it picks the two weak learners from the k_m-th source domain with minimum error as $h = \{h_1, h_2\} = \{h_{k_m 1}, h_{k_m 2}\}$. The k_m-th source domain plays the same role as the single domain in Mv-TLAdaboost. Following the steps of Mv-TLAdaboost, the calculation of $agreement$, η, Acc^T and Acc^{S+T} is similar, holding the characteristics of target data set.

However, the update of weight vector is slightly different during the $(t+1)$-th iteration as below

$$w_i^{t+1} = \begin{cases} w_i^t(\beta_t \eta_t)^{-\max\{|h_t^1(\mathbf{x}_i^{target1}) - y_i|, |h_t^2(\mathbf{x}_i^{target2}) - y_i|\}}, & 1 \le i \le L_T \\ w_i^t \alpha^{\min\{\xi,\zeta\}}, & \text{otherwise} \end{cases},$$

where ξ and ζ are defined as

$$\xi = \max\{|h_t^1(\mathbf{x}_i^{k1}) - y_i^k|, |h_t^2(\mathbf{x}_i^{k2}) - y_i^k|\},$$
$$\zeta = \max\{|h_t^{1k}(\mathbf{x}_i^{k1}) - y_i^k|, |h_t^{2k}(\mathbf{x}_i^{k2}) - y_i^k|\},$$

where $\mathbf{x}_i^{k1}$ and $\mathbf{x}_i^{k2}$ is the ith example from the kth source domain in two views, respectively. The newly proposed parameter ζ is designed for judging whether it is classified correctly by its own weaker learners $\{h^{k1}, h^{k2}\}$ because the final weaker learners $\{h^1, h^2\}$ may not be suitable to make a classification for all the source domains.

By selecting the nearest data set among multiple sources, MsTL-MvAdaboost algorithm further reduces classification error rates compare to Mv-TLAdaboost algorithm. The optimization for MsTL-MvAdaboost algorithm aims at solving the negative transfer problem, and helps move forward in multiview transfer learning.

7.5 Multiview Multitask Learning

In traditional machine learning, multiview learning focus on the consistency among diverse views of single task, while multitask learning ignores different views. Multiview multitask (MVMT) learning is proposed to meet the challenges that single view or single task learning are limited when label samples are dispersed in different tasks or views.

7.5.1 Graph-Based Interative Multiview Multitask Learning

$IteM^2$ is an iterative algorithm under the graph-based MVMT framework ($GraM^2$) (He and Lawrence 2011) which models the relationship between the examples and features in diverse views within each task through bipartite graph as shown in Fig. 7.1. There are two tasks with shared views and their own specific views.

Suppose there are T tasks and V views where the ith task has V_i views and n_i samples $\mathbf{X}_i = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{in_i}\}$ accompanied with m_i labeled samples $y_{i1}, y_{i2}, \ldots, y_{im_i}$, $m_i \ll n_i$, $i = 1, \ldots, T$. For the ith task in the kth view, there are d_{ik} features. In general, multiple related tasks have both shared and task-specific views, which can be represented as S_{ij}, the set of indexes of common views between

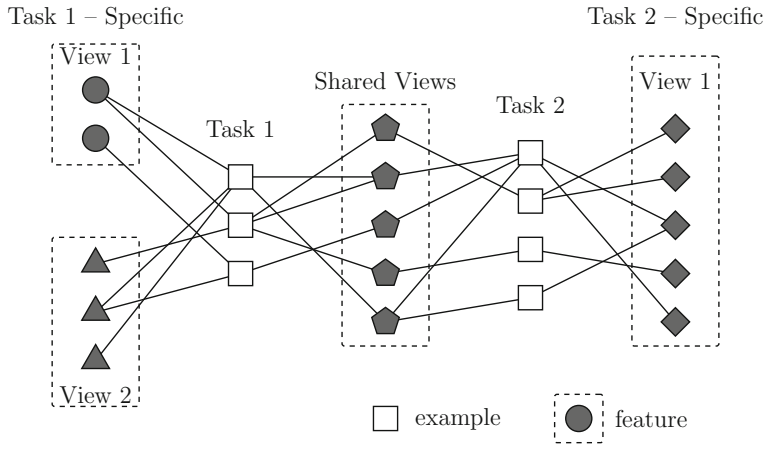


Fig. 7.1 Bipartite graph on MVMT learning

the ith task and the jth task. Combined with all tasks and views, it constructs a bipartite graph $G = \bigcup G_{ik}$, where G_{ik} is a subgraph of the ith task in the kth view which consists of a node set N_{ik} and an edge set E_{ik}. In N_{ik}, there are two types of nodes namely, the examples in the ith task, and the features in the kth view. An edge between the node of the s-th example and the node of the tth feature will be added into E_{ik} if and only if the feature value for the example is possible, with the edge weight equaling the feature value.

To measure the overall consistency of the i-th task, it introduces a weight matrix $\mathbf{A}_{ik}$ and two functions $g_i(\cdot)$ and $f_{ik}(\cdot)$ on the examples and features, respectively. The sign of $g_i(s)$ indicates the sign of the s-th example's class label for the i-th task, while the sign of $f_{ik}(t)$ determines the class label of an example having such features and the value of $|f_{ik}(t)|$ shows the degree of polarity. For clarity, $\mathbf{g}_i$ concatenates the values of $g_i(\cdot)$ on $1, \ldots, n_i$ and $\mathbf{f}_{ik}$ concatenates the values of $f_{ik}(\cdot)$ on $1, \ldots, d_{ik}$. $\mathbf{A}_{ik}$ is an $n_i \times d_{ik}$ matrix with its element $A_{ik}(s, t)$ in the s-th row and the t-th column set as feature value. With the help of $\mathbf{A}_{ik}$, g_i and f_{ik}, the consistency for the i-th task in the k-th view is measured by

$$C_{ik} = \sum_{s=1}^{n_i} \sum_{t=1}^{d_{ik}} A_{ik}(s,t) \left(\frac{g_i(s)}{\sqrt{D_{ik}(s)}} - \frac{f_{ik}(t)}{\sqrt{D_{ik}(n_i + t)}} \right)^2$$

where $\mathbf{D}_{ik}$ is a diagonal matrix to normalize feature values by summing up the s-th row of $\mathbf{W}_{ik}$ where

$$\mathbf{W}_{ik} = \begin{pmatrix} \mathbf{0}_{n_i \times n_i} & \mathbf{A}_{ik} \\ \mathbf{A}_{ik}^{\mathsf{T}} & \mathbf{0}_{d_{ik} \times d_{ik}} \end{pmatrix}.$$

Then the consistency can be simplified by

$$C_{ik} = \|\mathbf{g}_i\|^2 + \|\mathbf{f}_{ik}\|^2 - 2\mathbf{g}_i^{\mathsf{T}}\mathbf{L}_{ik}\mathbf{f}_{ik}$$

where $\mathbf{L}_{ik}$ is an $n_i \times d_{ik}$ matrix with $\mathbf{L}_{ik} = T_{ik}(s, n_i + t)$ and

$$\mathbf{T}_{ik} = \mathbf{D}_{ik}^{-1/2}\mathbf{W}_{ik}\mathbf{D}_{ik}^{-1/2}.$$

By summing up the consistency among all different views for the ith task, it obtains the consistency of the ith task:

$$C_i = \sum_{k=1}^{V_i} a_{ik}C_{ik} + \mu_i\|\mathbf{g}_i - \mathbf{y}_i\|^2,$$

where a_{ik}, μ_i are positive parameters, and $\mathbf{y}_i$ is an n_i-dimension vector consists of labels of m_i samples and zeros remained. Additionally, the second term measures the consistency with the label information, avoiding $\mathbf{g}_i$ away from labeling truth.

For each view, it calculates $\|\mathbf{f}_{ik} - \mathbf{f}_{jk}\|^2$ to measure the similarity of the i-th task and the j-th task, which helps to improve performance of one task with the information from its related tasks. As a consequence, the view consistency of each task and the task similarity in each view is mixed as

$$Q(f, g) = \sum_{i=1}^{\mathsf{T}} C_i + b\sum_{i=1}^{\mathsf{T}}\sum_{j=1}^{\mathsf{T}}\sum_{k\in S_{ij}} \|\mathbf{f}_{ik} - \mathbf{f}_{jk}\|^2,$$

under which there is $IteM^2$ algorithm to optimize the above objective function above.

To begin with, it initializes $\mathbf{g}_i$ as $\mathbf{y}_i$ and calculates $\mathbf{T}_{ik}$ and $\mathbf{L}_{ik}$ for every task in every view. In each iteration, it updates both $\mathbf{g}_i$ and $\mathbf{f}_{ik}$ alternatively. The specific expressions and corresponding theoretical analysis are showed in (reference), which begins with the first derivative of the objective function $Q(f, g)$ with respect to $\mathbf{g}_i$ and discusses the optimal solution on different classification criteria. After several times of iterations, it predicts the labels of examples by sorting $\mathbf{g}_i$ in descending order and selecting a proportion of examples to be positive.

Due to the fact that traditional multiview learning and multitask learning are two special cases of MVMT learning with $T = 1$ and $V = 1$, respectively, $IteM^2$ on $GraM^2$ is potentially applied for many scenarios. Nevertheless, the treatment of MVMT learning in $IteM^2$ is limited because it is unable to generate predictive models on independent, unknown testing samples. It is designed primarily for text categorization where the feature values are all nonnegative, which is unable to handle complex datasets.

7.5.2 Co-regularized Multiview Multitask Learning Algorithm

Based on the fact that MVMT learning is a general form of single task multiview learning, co-regularization, as a common multiview learning method applied in real applications, can be extended to MVML learning (regMVMT) (Zhang and Huan 2012). Across different tasks, additional regularization functions are utilized to ensure the functions learned in each view have similarity. Therefore, regMVMT considers information from multiple views and learn multiple related tasks simultaneously under a supervised learning framework.

We first give some notations. There are T tasks in total where the t-th task has N_t labeled samples and M_t unlabeled samples with $N_t \ll M_t$. Additionally, there are V views to describe each example where the vth view has D_v features. For the tth task in the vth view, the feature matrices of the labeled and unlabeled examples are $\mathbf{X}_t^v \in \mathbb{R}^{N_t \times D_v}$ and $\mathbf{U}_t^v \in \mathbb{R}^{M_t \times D_v}$. For labeled examples, the label vector of them is $\mathbf{y}^t \in \{1, -1\}^{N_t \times 1}$. Thus, the concatenated feature matrices of V views can be written as $\mathbf{X}_t = (\mathbf{X}_t^1, \mathbf{X}_t^2, \ldots, \mathbf{X}_t^V)$ and $\mathbf{U}_t = (\mathbf{U}_t^1, \mathbf{U}_t^2, \ldots, \mathbf{U}_t^V)$. As each task has different types of views, respectively, an indicator matrix $\mathbf{I}_d \in \{0, 1\}^{T \times V}$ is introduced to reflect the relationship between tasks and views.

For the tth task in the vth view, it introduces a mapping function $\mathbf{f}_t^v : \mathbb{R}^{D_v} \to \{1, -1\}$, which is supposed to behave similarly in different tasks for a given view, and agree with the other views for a given task. Recall the basic framework of co-regularization for single task where f^v is the prediction from the v-th view. The prediction is measured by averaging those of each view:

$$f(\mathbf{x}) = \frac{1}{V} \sum_{v=1}^{V} f^v(\mathbf{x}^v);$$

and the objective function is

$$\min_{f^v} \; L(\mathbf{y}, f(\mathbf{X})) + \frac{\lambda}{2} \|f\|^2 + \frac{\mu}{2} \sum_{v' \neq v} \|f^{v'}(\mathbf{U}^{v'}) - f^v(\mathbf{U}^v)\|^2$$

where $L(\cdot, \cdot)$ is the loss function, λ is a regularization coefficient and μ is a coupling parameter.

To extend co-regularization across different tasks, an additional regularization function penalizing the difference of the view specific function on the same view is defined as

$$f_t(\mathbf{X}_t) = \frac{1}{V} \sum_{v=1}^{V} \mathbf{X}_t^v \mathbf{w}_t^v = \frac{\mathbf{X}_t \mathbf{w}_t}{V},$$

where $\mathbf{w}_t \in \mathbb{R}^{D\times 1}$ is a column vector concatenated by $\mathbf{w}_t^v$ from all the views v. And $\mathbf{w}_t^v \in \mathbb{R}^{D_v\times 1}$ is the parameters of the prediction function from the vth view. Thus, the objective function across multiple tasks is given by

$$\min_{f_t^v} \sum_t L(\mathbf{y}_t, f_t(\mathbf{X}_t)) + \frac{\lambda}{2}\|f_t\|^2 + \frac{\mu}{2}\sum_{v'\neq v}\|f_t^{v'}(\mathbf{U}_t^{v'}) - f_t^v(\mathbf{U}_t^v)\|^2.$$

Then, substitute f_t^v by weight vectors as follows:

$$\begin{aligned}\min_{\mathbf{w}_t^v} \sum_{t=1}^{\mathsf{T}} \frac{1}{2}\|\mathbf{y}_t - \frac{\mathbf{X}_t\mathbf{w}_t}{V}\|^2 + \frac{\lambda}{2}\sum_{v=1}^{V}\|\mathbf{w}_t^v\|^2 \\ + \frac{\mu}{2}\sum_{v'\neq v}^{V}\|\mathbf{U}_t^{v'}\mathbf{w}_t^{v'} - \mathbf{U}_t^v\mathbf{w}_t^v\|^2 + \frac{\gamma}{2}\sum_{t'\neq t}^{\mathsf{T}}\|\mathbf{w}_t^{v'} - \mathbf{w}_t^v\|^2,\end{aligned}$$

which is denoted as F from now on. In objective function F, it adds parameter γ as a new regularization parameter to penalize the difference between tasks for the same view, which is the reason for so-called regMVMT. It ensures the learned functions are similar during the process of optimization.

To find the optimal solution of F, it calculates the derivative with respect to each $\mathbf{w}_t^v$ as

$$\begin{aligned}\frac{\partial F}{\partial \mathbf{w}_t^v} = \frac{1}{V}(\mathbf{X}_t^v)^{\mathsf{T}}\left(\frac{\mathbf{X}_t\mathbf{w}_t}{V} - \mathbf{y}_t\right) + \lambda\mathbf{w}_t^v + \mu(\mathbf{U}_t^v)^{\mathsf{T}}\sum_{v'\neq v}^{V}(\mathbf{U}_t^{v'}\mathbf{w}_t^{v'} - \mathbf{U}_t^v\mathbf{w}_t^v) \\ + \frac{\gamma}{2}\sum_{t'\neq t}^{\mathsf{T}}\|\mathbf{w}_t^{v'} - \mathbf{w}_t^v\|^2,\end{aligned}$$

Set the derivative to zero, one can obtain the following equation:

$$E_{tv} = A_{tv} + \sum_{v'\neq v} B_{vv'}^t\mathbf{w}_t^{v'} + \sum_{v'\neq v} Ct'v^t\mathbf{w}_{t'}^v,$$

with coefficients A_{tv}, $B_{vv'}^t$, $Ct'v^t$ and E_{tv} absorbing corresponding terms as follows:

$$\begin{aligned}A_{tv} &= \lambda + \gamma(T-1) + \mu(V-1)(\mathbf{U}_t^v)^{\mathsf{T}}\mathbf{U}_t^v + \frac{(\mathbf{X}_t^v)^{\mathsf{T}}\mathbf{X}_t^v}{V^2},\\ B_{vv'}^t &= \frac{(\mathbf{X}_t^v)^{\mathsf{T}}\mathbf{X}_t^{v'}}{V^2} - \mu(\mathbf{U}_t^v)^{\mathsf{T}}\mathbf{U}_t^{v'},\\ Ct'v^t &= \gamma I_{D_v},\\ E_{tv} &= \frac{(\mathbf{X}_t^v)^{\mathsf{T}}\mathbf{y}_t}{V},\end{aligned}$$

where I_{D_v} is a $D_v \times D_v$ identity matrix. The coefficients can be further simplified and be concatenated as a sparse matrix. In such a scenario, it will obtain a minimalist matrix equation as

$$\mathbf{LW} = \mathbf{R}$$

where $\mathbf{L}$ contains coefficients such as A_{tv}, $B^t_{vv'}$, $Ct'v^t$, $\mathbf{W}$ is the solution of this equation containing E_{tv}, and $\mathbf{R}$ is a vector consisting of E_{tv}. As saving this equation directly suffer heavy computational expense, an iterable procedure is recommended to catch the solution until it goes convergence.

The regMVMT algorithm also has the ability of complementing structured missing data by changing the indicator matrix in Zhang and Huan (2012). However, the proposed algorithm is designed for tens of tasks as a result of sparse matrix operation, which may fail to handle hundreds of tasks or more.

7.5.3 *Convex Shared Structure Learning Algorithm for Multiview Multitask Learning*

The setting of regMVMT mentioned in 7.5.2 is restricted since it requires similar tasks producing similar model parameters. To deal with more general problems, (Jin et al. 2013) proposes a shared structure learning framework called CSL-MTMV, in which both the shared predictive structure among multiple tasks and prediction consistency among different views within a single task are considered. The regMVMT can be regarded as a special case of CSL-MTMV.

In CSL-MTMV, different tasks can share information through the structure parameters Θ flexibly, making it possible to learn underlying predictive functional structures in the hypothesis space. The notations of CSL is similar to those of regMVMT as labeled samples $\mathbf{X}_t$, unlabeled samples $\mathbf{U}_t$, a class label vector $\mathbf{y}_t$ and an indicator matrix I_d.

Shared structure learning assumes that the underlying structure is a shared low-dimensional subspace, and a linear form of feature map is considered for simplicity as follows:

$$f_t(\mathbf{x}) = \mathbf{w}_t^\mathsf{T}\mathbf{x} + \mathbf{z}_t^\mathsf{T}\Theta\mathbf{x} = \mathbf{a}_t^\mathsf{T}\mathbf{x}$$

where the structure parameter Θ takes the form of an $h \times d$ matrix with orthonormal rows. It updates the linear mapping in regMVMT by exploiting a shared feature space with Θ. An improved alternating structure optimization (iASO) formulation is given as follows:

$$\min_{\mathbf{a}_t, \mathbf{z}_t, \Theta} \sum_{t=1}^{\mathsf{T}} \left(\frac{1}{N_t} \sum_{i=1}^{N_t} L(f_t(x_{t,i}), y_{t,i}) + g_t(\mathbf{a}_t, \mathbf{z}_t, \Theta) \right)$$

where $g_t(\mathbf{a}_t, \mathbf{z}_t, \Theta)$ is the regularization function defined as

$$g_t(\mathbf{a}_t, \mathbf{z}_t, \Theta) = \alpha \|\mathbf{a}_t - \Theta^\intercal \mathbf{z}_t\|^2 + \beta \|\mathbf{a}_t\|^2.$$

The iASO consists of two terms. The first term controls optimal mapping close to label class, while the second term controls the task relatedness and complexity of the predictor function.

To scale the shared structure learning framework to MVMT learning, the directest way is to split V views, learn predictor function separately, and combine all hypothesis learned in each view. However, it is worthwhile to dig out the relationships between diverse views with co-regularization term:

$$\min_{\mathbf{a}_t, \mathbf{z}_t, \Theta} \sum_{t=1}^{\intercal} \sum_{v=1}^{V} \left(\frac{1}{N_t} \sum_{i=1}^{N_t} L(f_t^v(x_{t,i}^v), y_{t,i}) + g_t^v(\mathbf{a}_t^v, \mathbf{z}_t^v, \Theta^v) \right.$$
$$\left. + \gamma \frac{1}{M_t} \sum_{j=1}^{M_t} \sum_{v' \neq v} (f_t^{v'}(u_{t,j}^{v'}) - (f_t^v(u_{t,j}^v))^2 \right)$$

with

$$g_t^v(\mathbf{a}_t^v, \mathbf{z}_t^v, \Theta^v) = \alpha \|\mathbf{a}_t^v - (\Theta^v)^\intercal \mathbf{z}_t^v\|^2 + \beta \|\mathbf{a}_t^v\|^2.$$

The objective function is difficult to solve as its non-convexity, and it is desirable to convert it into a convex formulation. Assume $A^v = \{a_1^v, a_2^v, \ldots, a_T^v\} \in \mathbb{R}^{D_v \times T}$ and $Z^v = \{z_1^v, z_2^v, \ldots, z_T^v\} \in \mathbb{R}^{h \times T}$ with $z_t^v = \Theta^v a_t^v$, so $Z^v = \Theta^v \mathbf{A}^v$. Thus it denotes:

$$G(\mathbf{A}^v, \Theta^v) = \min_{Z^v} \sum_{t=1}^{\intercal} g_t^v(\mathbf{a}_t^v, \mathbf{z}_t^v, \Theta^v) = \alpha tr\left((\mathbf{A}^v)^\intercal((1+\eta)I - (\Theta^v)^\intercal \Theta^v)\mathbf{A}^v\right).$$

Under the assumption that $\eta = \beta/\alpha$ and the domain of Θ^v is a convex set, G_0 is reduced to

$$G(\mathbf{A}^v, \Theta^v) = \alpha\eta(1+\eta) tr\left((\mathbf{A}^v)^\intercal(\eta I + M^v)^{-1}\mathbf{A}^v\right).$$

with $M^v = (\Theta^v)^\intercal \Theta^v$ relaxing the orthonormal constraints on Θ^v. Thus, the objective function is updated as

$$\min_{\mathbf{a}_t, M^v} \sum_{t=1}^{\intercal} \sum_{v=1}^{V} \left(\frac{1}{N_t} \sum_{i=1}^{N_t} L\left(f_t^v(x_{t,i}^v), y_{t,i}\right) + \gamma \frac{1}{M_t} \sum_{j=1}^{M_t} \sum_{v' \neq v} \left(f_t^{v'}(u_{t,j}^{v'}) - f_t^v(u_{t,j}^v)\right)^2 \right)$$
$$+ \sum_{v=1}^{V} G(\mathbf{A}^v, \Theta^v),$$
$$s.t.\ \ tr(M^v) = h, M^v \preceq I, M^v \in \mathbb{S}_+.$$

With the help of convex relaxation, the optimal solution can be approximated using the top h eigenvectors corresponding to the largest h eigenvalues.

In CSL-MTMV algorithm, the two variables to be optimized is updated alternately. It computes $\mathbf{A}^v$ for given $\mathbf{M}^v$ at first, then computes $\mathbf{M}^v$ for given $\mathbf{A}^v$ later. At the first step, the way of representing derivative of objective function is similar to that of regMVMT, which using coefficients to simplify the representation of objective equation. At the second step, it uses singular value decomposition to catch the approximate solution.

7.6 Other Methods

Multi-transfer (Tan et al. 2014): Transfer learning with multiple views and multiples sources (TL-MVMS) is the general form of multiview transfer learning (MVTL) and multisource transfer learning (MSTL). The most naive method to solve TL-MVMS is to directly employ MVTL and MSTL respectively, but different sources may follow different distributions and different views from different sources may be inconsistent with each other. Although co-training is effective to utilize multiview data, it may cause problems that decision boundaries of source and target domains can be very different and the predictions across domains are no longer consistent. Multi-transfer extends co-training and overcomes these problems. It introduces a harmonic function based criterion to select the appropriate target instances, then applies a density ratio weighting scheme and a nonparametric method to measure and revise the distribution shift in both the marginal distribution and conditional probability between the source and target domains. Multi-transfer performs well due to the fact of two aspects: first, by embedding transfer learning in a co-training framework, the knowledge from multiple views can be well exploited. Second, the distribution revision helps build a consistent model for the target domain with the robustness of knowledge transfer from multiple source domains.

MultitAsk MUltiview Discriminant Analysis (MAMUDA) (Jin et al. 2014): The most works for multitask multiview learning assume that tasks use the same set of class labels, which is not the case of reality. In order to be close to the real world, multitask multiview discriminant analysis (MAMUDA) is proposed under the assumption that tasks with multiple views correspond to different set of class labels. The idea of MAMUDA comes from the classical dimensionality reduction technique, linear discriminant analysis (LDA). A naive way to use LDA is to optimize the transformation matrix for dimensionality reduction for view v in task t in a trace ratio form. To better use the knowledge of multiple tasks with multiple views, MAMUDA divides the transformation process into two steps: first, the data from all the tasks are transformed from their corresponding original feature space into a common intermediate latent semantic space. Second, data from each view of each task are transformed from the common intermediate latent space into their corresponding discriminant space.

The common structure shared by multiple tasks reflects the intrinsic characteristics of the applications. Multiple views of one task are used to jointly learn some specific characteristics that are only contained in this task.

Multitask Multiview Clustering (Zhang et al. 2015): In real applications, there are clustering problems from multiple tasks in multiple views, which cannot simply employ multitask clustering algorithms or multiview clustering algorithms for they will leave out information of other views or tasks. Multitask multiview clustering (MTMVC) is proposed by combining within-view-task clustering, multiview relationship learning and multitask relationship learning together in a linear form. To ensure the consistency among different views in each task, it computes the similarity matrices under two different views. It uses the bipartite graph co-clustering (BiCo) algorithm (Dhillon 2001) in two parts: first, during the within-view-task clustering, it helps establish associations between the examples and any view within each task. Second, it learns the shared subspace among the related tasks in multitask relationship learning. Also, BiCo has the limit that it can only be used for nonnegative data. Thus, it maps the variables to the Stiefel manifold (Absil et al. 2010) for optimization by gradient ascent method.

References

Absil PA, Mahony R, Sepulchre R (2010) Optimization on manifolds: methods and applications. In: Recent advances in optimization and its applications in engineering, pp 125–144

Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining, pp 269–274

Diethe T, Hardoon DR, Shawe-Taylor J (2008) Multiview fisher discriminant analysis. In: Proceedings of NIPS workshop on learning from multiple sources

He J, Lawrence R (2011) A graphbased framework for multi-task multi-view learning. In: Proceedings of the 28th international conference on machine learning, pp 25–32

Jin X, Zhuang F, Wang S, He Q, Shi Z (2013) Shared structure learning for multiple tasks with multiple views. In: Proceedings of joint european conference on machine learning and knowledge discovery in databases, Springer, pp 353–368

Jin X, Zhuang F, Xiong H, Du C, Luo P, He Q (2014) Multi-task multi-view learning for heterogeneous tasks. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 441–450

Pan SJ, Yang Q et al (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

Perkins DN, Salomon G et al (1992) Transfer of learning. Int Encycl Educ 2:6452–6457

Tan B, Zhong E, Xiang EW, Yang Q (2014) Multi-transfer: transfer learning with multiple views and multiple sources. Statist Anal Data Min 7(4):282–293

Xu Z, Sun S (2011) Multi-view transfer learning with adaboost. In: Proceedings of the 23rd IEEE international conference on tools with artificial intelligence, IEEE, pp 399–402

Xu Z, Sun S (2012) Multi-source transfer learning with multi-view adaboost. In: Proceedings of the 19th international conference on neural information processing, Springer, pp 332–339

Yang P, Gao W (2013) Multi-view discriminant transfer learning. In: Proceedings of the 23rd international joint conference on artificial intelligence, pp 1848–1854

Zhang D, He J, Liu Y, Si L, Lawrence R (2011) Multi-view transfer learning with a large margin approach. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1208–1216

Zhang J, Huan J (2012) Inductive multi-task learning with multiple view data. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 543–551

Zhang X, Zhang X, Liu H (2015) Multi-task multi-view clustering for non-negative data. In: Proceedings of the 24th international joint conference on artificial intelligence, pp 4055–4061

Chapter 8
Multiview Deep Learning

Abstract The multiview deep learning described in this chapter deals with multiview data or simulates constructing its intrinsic structure by using deep learning methods. We highlight three major categories of multiview deep learning methods through three different thoughts. The first category of approaches focuses on obtaining a shared joint representation from different views by building a hierarchical structure. The second category of approaches focuses on constructing structured spaces with different representations of multiple views which gives some constraints between representations on a different view. The third major category approaches focuses on explicitly constructing connections or relationships between different views or representations, which allows different views to be translated or mapped to each other.

8.1 Introduction

Deep learning, developed over the past two decades, is an approach to learning data representation with models having hierarchical structure. In recent years, deep learning has seen tremendous growth in its popularity and usefulness, due in large part to more powerful computers, larger datasets, and techniques to train deeper networks. Here, the modern term "deep learning" is not limited the neuroscientific aspect on the current machine learning models. It can be a more general principle of learning *multiple levels of composition*, which can be applied to machine learning frameworks that are not necessarily neurally inspired. As more and more of social activities take place on computers and the internet, more and more of multiview data are recorded easily. A variety of multiview deep methods have been proposed to learn robust multiview representations and deal with multiview data.

Multiview deep learning has a wide range of applications such as natural language descriptions, multimedia content indexing and retrieval, and understanding human multiview behaviors during social interactions. Recently, researchers have been widely interested in the applications of deep learning in the natural language processing and computer vision area. Deep learning can intuitively construct feature hierarchies for modeling real-world data distribution, which can help multiview methods to extract multiview representations, relationship and complementarity information.

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_8

In this section, we first introduce joint representations which map single-view representations into a multiview space together. Second, we review the methods that mainly build complement structured space where each view has a corresponding projection function that maps it into a complementary structured multiview space. Finally, we discuss an important part of multiview machine learning, which learns to map a view to other views.

Overall, this section aims to provide an insightful overview of theoretical basis and state-of-the-art developments in the field of multiview deep learning and to help researchers find the most appropriate methods for particular applications.

8.2 Joint Representation

Compared to single-view learning, multiview learning explores the similarities and differences between different views. In this chapter, the representation is to jointly optimize all embedding to take advantage of the rich information from multiple views and improve the performance of subsequent learning tasks. The joint presentation emphasizes that each different view is combined into the same representation space for further training.

In mathematics, the joint representation is extracted as follows:

$$\mathbf{x}^{\star} = f(\mathbf{x}^1, \ldots, \mathbf{x}^n), \tag{8.1}$$

The multiview representation $\mathbf{x}^{\star}$ is calculated by using a function f (for example, a deep belief network, a restricted Boltzmann machine, or a deep neural network, etc.), relying on each view representation $\mathbf{x}^1, \ldots, \mathbf{x}^n$.

8.2.1 Probabilistic Graphical Models

8.2.1.1 Restricted Boltzmann Machines

Hinton et al. (2006) proposed a classical method which is an undirected graphical model named restricted Boltzmann machine (RBM) that can model distributions over binary-valued data. The Boltzmann machine and their extension to exponential family distributions have been successfully used in many applications. Restricted Boltzmann machine consists of stochastic hidden units $\mathbf{h} \in \{0, 1\}^F$ and stochastic visible units $\mathbf{v} \in \{0, 1\}^D$, where each visible unit $\mathbf{v}$ connected to each hidden unit $\mathbf{h}$. The following energy function $E : \{0, 1\}^{D+F} \rightarrow \mathbb{R}$ is defined in the model:

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\mathbf{v}^{\intercal} \boldsymbol{W} \mathbf{h} - \mathbf{b}^{\intercal} \mathbf{v} - \mathbf{a}^{\intercal} \mathbf{h}. \tag{8.2}$$

Here, $\boldsymbol{\theta} = \{\mathbf{a}, \mathbf{b}, \boldsymbol{W}\}$ are the model parameters. We can obtain the following exponential family distributions by normalization, which defines the joint distribution on

the visible units and hidden units:

$$P(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right) \tag{8.3}$$

where $Z(\boldsymbol{\theta})$ is the normalizing constant with $\boldsymbol{\theta}$.

The above restricted Boltzmann machine as building blocks can be extended to replicate the softmax model which has proven to be a valid model for sparse word count vector, and Gaussian RBMs which have been used to model real-value inputs for speech and visual tasks.

The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ of the Gaussian RBM is defined as follows:

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{1}{2}(\mathbf{v} - \mathbf{b})^{\intercal} \boldsymbol{\Sigma}^{-1}(\mathbf{v} - \mathbf{b}) - \mathbf{v}^{\intercal} \boldsymbol{\Sigma}^{-\intercal} \boldsymbol{W}\mathbf{h} - \mathbf{a}^{\intercal}\mathbf{h} \tag{8.4}$$

where $\boldsymbol{\theta} = \{\mathbf{a}, \mathbf{b}, \boldsymbol{W}, \boldsymbol{\Sigma}\}$ are the model parameters.

8.2.1.2 Multimodal Deep Boltzmann Machine

A deep Boltzmann machine (DBM) (Salakhutdinov and Larochelle 2010) is also an undirected graphical model, which is a hierarchal structural extension of RBM with bipartite connections between adjacent layers of hidden units. The key idea of DBM is to flexibly represent the joint density model over the multiview input space by probabilistic undirected graphical models. Further, the problem of the missing views can be dealt with sampling from the conditional distributions. In contrast to neural networks, DBM is a probabilistic undirected graph model, so the representation of data is probabilistic. Thus when DBM is training, supervised data is not necessary. From the aspect of the network structure, in theoretical expected DBM, data can be continuously abstracted by each successive layer to represent data with a higher level of abstraction, which is similar to neural networks. Specifically, DBM also can be converted to a deterministic neural network (Salakhutdinov and Larochelle 2010), which will lose some performance of the generative capacity in the model.

A DBM is a network of symmetrically coupled stochastic binary units. It contains a set of visible units $\mathbf{v} \in \{0, 1\}^D$, and a sequence of layers of hidden units $\mathbf{h}_1 \in \{0, 1\}^{F_1}$, $\mathbf{h}_2 \in \{0, 1\}^{F_2}$, $\ldots$, $\mathbf{h}_L \in \{0, 1\}^{F_L}$. There are connections only between hidden units in adjacent layers. Let us first consider a DBM with two hidden layers. The energy of the joint configuration $\{\mathbf{v}, \mathbf{h}\}$ is defined as (ignoring bias terms)

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\mathbf{v}^{\intercal} \boldsymbol{W}_1 \mathbf{h}_1 - \mathbf{h}_1^{\intercal} \boldsymbol{W}_2 \mathbf{h}_2 \tag{8.5}$$

where $\mathbf{h} = \{\mathbf{h}_1, \mathbf{h}_2\}$ represents the set of hidden units, and $\theta = \{\boldsymbol{W}_1, \boldsymbol{W}_2\}$ are the model parameters that represent visible-to-hidden and hidden-to-hidden symmetric interaction terms. Similar to RBMs, this binary–binary DBM can be easily extended to modeling dense real-valued or sparse count data.

Srivastava and Salakhutdinov (2012b) proposed the model of a multimodal DBM by using an image-text bimodal DBM . Let $\mathbf{v}^{\mathrm{m}} \in \mathbb{R}^D$ denote an image input and $\mathbf{v}^{\mathrm{t}} \in \mathbb{N}^K$ denote a text input. Consider modeling each data view using separate two-layer DBMs. The probability that each DBM model assigns to a visible vector is (ignoring bias terms on the hidden units for clarity)

$$P(\mathbf{v}^{\mathrm{m}}) = \sum_{\mathbf{h}_1, \mathbf{h}_2} P(\mathbf{v}^{\mathrm{m}}, \mathbf{h}_1, \mathbf{h}_2; \boldsymbol{\theta}) \tag{8.6}$$

$$\begin{aligned} P(\mathbf{v}^{\mathrm{t}}) &= \sum_{\mathbf{h}_1, \mathbf{h}_2} P(\mathbf{v}^{\mathrm{t}}, \mathbf{h}_1, \mathbf{h}_2; \boldsymbol{\theta}) \\ &= \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-\frac{1}{2}(\mathbf{v}^{\mathrm{t}} - \mathbf{b})^{\intercal} \boldsymbol{\Sigma}^{-1}(\mathbf{v}^{\mathrm{t}} - \mathbf{b}) + (\mathbf{v}^{\mathrm{t}})^{\intercal} \boldsymbol{\Sigma}^{-\intercal} \boldsymbol{W}_1 \mathbf{h}_1 + \mathbf{h}_1^{\intercal} \boldsymbol{W}_2 \mathbf{h}_2\right) \end{aligned} \tag{8.7}$$

Here (8.4) is used for the energy function from the Gaussian RBM.

Note that, the visible–hidden interaction term is from the Gaussian RBM and the hidden–hidden one is from the binary RBM. Similarly, the text-specific DBM will use terms from the replicated softmax model for the visible–hidden interactions and the hidden–hidden ones from the binary RBM.

To form a multimodal DBM, two models are combined by adding an additional layer of binary hidden units on top of them. The resulting graphical model is shown in Fig. 8.1a in Sect. 8.2.1.3. The joint distribution over the multimodal input can be written as

$$P(\mathbf{v}^{\mathrm{m}}, \mathbf{v}^{\mathrm{t}}) = \sum_{\mathbf{h}_2^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{t}}, \mathbf{h}_3} P\left(\mathbf{h}_2^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{t}}, \mathbf{h}_3\right) \left(\sum_{\mathbf{h}_1^{\mathrm{m}}} P(\mathbf{v}^{\mathrm{m}}, \mathbf{h}_1^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{m}})\right) \left(\sum_{\mathbf{h}_1^{\mathrm{t}}} P(\mathbf{v}^{\mathrm{t}}, \mathbf{h}_1^{\mathrm{t}}, \mathbf{h}_2^{\mathrm{t}})\right) \tag{8.8}$$

Like RBM, exact maximum likelihood learning in this model is also intractable, while efficient approximate learning can be implemented by using mean-field inference to estimate data-dependent expectations, and an MCMC based stochastic approximation procedure to approximate the models expected sufficient statistics (Salakhutdinov and Larochelle 2010).

It is also very natural to extend DBM to the multiview learning area because one of the DBMs great strengths lies in its performance of generative capacity with representations. In this way, multiview DBM is allowed to deal with missing data in a simple way. Even if the entire view of the data is missing, the model has a natural response. For example, a multiview DBM can generate samples in a variety of ways, either from the representation of another view or from a shared joint representation. Similar to DBM and the multiview autoencoders introduced later, when training multiview DBMs, supervised data are not necessary. Of course, multiview DBM also has problems that DBM also has. The main disadvantage of multiview DBM is that it is difficult to train with high computational cost, and it needs to use the approximate variational training methods (Srivastava and Salakhutdinov 2012b).

Multimodal DBM has been widely used for multiview representation learning (Huang and Kingsbury 2013; Hu et al. 2013; Ge et al. 2013). Hu et al. (2013) employed the multimodal DBM to learn joint representation for predicting answers in community-based question answering (cQA) portals. Ge et al. (2013) applied the multimodal RBM to determining information trustworthiness, in which the learned joint representation denotes the consistent latent reasons that underline users ratings from multiple sources. Pang and Ngo (2015) proposed to learn a joint density model for emotion prediction in user-generated videos with a deep multimodal Boltzmann machine. This multimodal DBM is exploited to model the joint distribution over visual, auditory, and textual features. Here, Gaussian RBM is used to model the distributions over the visual and auditory features, and replicated softmax topic model is applied for mining the textual features. Ouyang et al. (2014) proposed a multisource deep model based on multimodal DBM for the task of human pose estimation. The proposed method nonlinearly integrates three information sources and improves pose estimation accuracy and is superior to single-modal data. Similarly, (Suk et al. 2014) used multimodal DBM to achieve better performance on the task of disease classification with multisource data.

Further, (Sohn et al. 2014) investigated an improved multimodal RBM framework via minimizing the variation of information between data modalities through the shared latent representation. Recently, (Ehrlich et al. 2016) presented a multitask multimodal RBM (MTM-RBM) approach for facial attribute classification. In particular, a multitask RBM is proposed by extending the formulation of discriminative RBM (Larochelle and Bengio 2008) to account for multiple tasks. And then multitask RBM is naturally extended to MTM-RBM by combining a collection of unimodal MT-RBM, one for each visible modality. Unlike the typical multimodal RBM, the learned joint feature representation of MTM-RBM enables interaction between different tasks so that it is suited for multiple attribute classification.

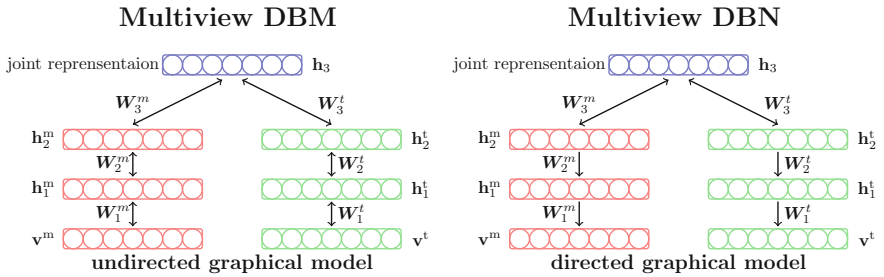
8.2.1.3 Multimodal Deep Belief Network

Srivastava and Salakhutdinov (2012a) illustrate the construction of a multimodal DBN by using an image-text bimodal DBN (Hinton and Salakhutdinov 2006) in Fig. 8.1b. Let $\mathbf{v}^{\mathrm{m}} \in \mathbb{R}^D$ denote an image input and $\mathbf{v}^{\mathrm{t}} \in \mathbb{N}^K$ denote a text input. Consider modeling each data view using separate two-layer DBNs. The probability that each DBN model assigns to a visible vector is (ignoring bias terms on the hidden units for clarity)

$$P(\mathbf{v}^{\mathrm{m}}) = \sum_{\mathbf{h}_1, \mathbf{h}_2} P(\mathbf{h}_2, \mathbf{h}_1) P(\mathbf{v}^{\mathrm{m}} | \mathbf{h}_1) \tag{8.9}$$

$$P(\mathbf{v}^{\mathrm{t}}) = \sum_{\mathbf{h}_1, \mathbf{h}_2} P(\mathbf{h}_2, \mathbf{h}_1) P(\mathbf{v}^{\mathrm{t}} | \mathbf{h}_1) \tag{8.10}$$

To form a multimodal DBN, two DBNs are combined by learning an additional layer of binary hidden units on top of them. The resulting graphical model is shown in Fig. 8.1b. The joint distribution can be written as



(a) the multimodal DBM (b) the multimodal DBN

Fig. 8.1 The difference of graphical model with bipartite connections between adjacent layers of hidden units

$$P(\mathbf{v}^{\mathrm{m}}, \mathbf{v}^{\mathrm{t}}) = \sum_{\mathbf{h}_2^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{t}}, \mathbf{h}_3} P\left(\mathbf{h}_2^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{t}}, \mathbf{h}_3\right) \left(\sum_{\mathbf{h}_1^{\mathrm{m}}} P(\mathbf{v}^{\mathrm{m}}, \mathbf{h}_1^{\mathrm{m}}) P(\mathbf{h}_1^{\mathrm{m}} | \mathbf{h}_2^{\mathrm{m}})\right) \left(\sum_{\mathbf{h}_1^{\mathrm{t}}} P(\mathbf{v}^{\mathrm{t}}, \mathbf{h}_1^{\mathrm{t}}) P(\mathbf{h}_1^{\mathrm{t}} | \mathbf{h}_2^{\mathrm{t}})\right) \tag{8.11}$$

Although deep belief networks (DBNs) and deep Boltzmann machines (DBMs) diagrammatically look very similar, they are actually qualitatively very different. This is because DBNs are directed graphical models and DBMs are undirected graphical models. In a DBN the connections between layers are directed. Therefore, the first two layers form an RBM (an undirected graphical model), and then the subsequent layers form a directed generative model. In a DBM, the connection between all layers is undirected, and thus each pair of layers forms an RBM. Hinton (2009). Further, Kim et al. (2013) used a deep belief network to model each view separately and extracted joint representation for audiovisual emotion recognition.

Recently, the advantages of the deep Gaussian processes with the multiview learning were combined. Sun and Liu (2018) proposed the model of multiview deep Gaussian processes (MvDGPs), which takes full account of the characteristics of multiview data. MvDGPs can independently determine the modeling depths for each view, which is more flexible and powerful than the deep Gaussian process counterpart.

8.2.2 Fusion of Networks

8.2.2.1 Split Multiview Autoencoders

Although multimodal DBMs are very successful in learning shared joint representations in different views, they still have the limitations described in the previous sections. For example, there is no explicit objective for the models to discover correlations across the views such that some hidden units are tuned only for one view while others are tuned only for the other. Multiview deep autoencoders become good

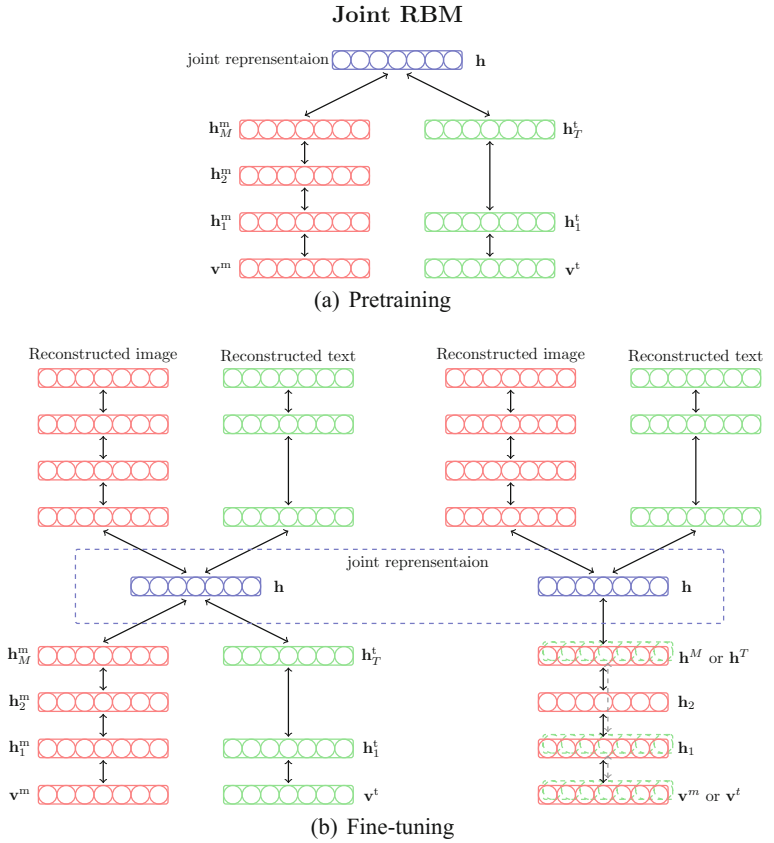


Fig. 8.2 *Top*: The multimodal DBN in pretraining *Bottom*: The multimodal AutoEncoder and the cross-modality Autoencoder in fine-tuning. They are optimized together

alternatives for learning a shared representation between views due to the sufficient flexibility of their objectives.

In the multiview feature learning, we have access to paired observations from two views, denoted $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$, where N is the sample size and $\mathbf{x}_i \in \mathbb{R}^{D_x}$ and $\mathbf{y}_i \in \mathbb{R}^{D_y}$ for $i = 1, \ldots, N$. We also denote the data matrices for each view $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$. We use bold-face letters, e.g., $\mathbf{f}$, to denote multidimensional mappings implemented by DNNs. A DNN $\mathbf{f}$ of depth $K_\mathbf{f}$ implements a nested mapping of the form $\mathbf{f}(\mathbf{x}) = \mathbf{f}_{K_\mathbf{f}}((\cdots \mathbf{f}_1(\mathbf{x}; \mathbf{W}_1) \cdots); \mathbf{W}_{K_\mathbf{f}})$, where $\mathbf{W}_j$ are the weight parameters of layer $j, j = 1, \ldots, K_\mathbf{f}$, and $\mathbf{f}_j$ is the mapping of layer j which takes the form of a linear mapping followed by an element-wise activation: $\mathbf{f}_j(\mathbf{t}) = s(\mathbf{W}_j^\top \mathbf{t})$, and typical choices for s include sigmoid, tanh, ReLU, etc. The collection of the learnable parameters in model $\mathbf{f}$ is denoted $\mathbf{W}_\mathbf{f}$, e.g., $\mathbf{W}_\mathbf{f} = \mathbf{W}_1, \ldots, \mathbf{W}_{K_\mathbf{f}}$ in the DNN case. We write the $\mathbf{f}$-projected (view 1) data matrix as $\mathbf{f}(\mathbf{X}) = [\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_N)]$. The dimensionality of the projection (feature vector), which is equal to the number of output units in the DNN case, is denoted L.

Ngiam et al. (2011) proposed a model that extended the idea of using autoencoders to the multiview domain. They extract shared representations via training the bimodal deep autoencoder in a denoising fashion (Vincent et al. 2008), using an augmented dataset with examples that require the network to reconstruct both views given only one view that is available at test time. Both models are pretrained using sparse RBMs. The key idea of this split multiview autoencoders is to use greedy layer-wise training with an extension to sparse RBMs followed by fine-tuning, since a single layer RBM tends to learn single-view units, and it is much more efficient to learn separate models for each view.

In this approach, the joint representation of feature extraction network f is shared to reconstruct networks p and q for each view (Fig. 8.2), separately. The optimization function of this model is the sum of reconstruction errors for the two views (omit the ℓ_2 weight decay term):

$$\min_{\boldsymbol{W}_f, \boldsymbol{W}_p, \boldsymbol{W}_q} \frac{1}{N} \sum_{i=1}^{N} \left(\|\mathbf{x}_i - p(f(\mathbf{x}_i))\|^2 + \|\mathbf{y}_i - q(f(\mathbf{x}_i))\|^2 \right) \tag{8.12}$$

In practice, the examples $\mathbf{X}$ have zero values for one of the input views (e.g., video) and original values for the other input view (e.g., audio), but still require the network to reconstruct both views (audio and video). Thus, one-third of the training data have only video for input, while another one-third of the data have only audio, and the last one-third of the data have both audio and video.

The intuition for this model is that the shared representation can be extracted from a single view, and only one view can be used to reconstruct all views in thetest. The loss function in the autoencoder is an empirical expectation of the loss caused by each training sample, and thus stochastic gradient descent (SGD) can be used to effectively optimize the objective using the gradient estimated on the small minibatches of examples.

Moreover, the model can naturally alleviate a difficult problem in multiview learning to deal with missing data, that is, it is easy to obtain multiview representation in the absence of some views. Finally, similar to multimodal DBMs and multimodal DBNs, it can be possible to fill in the missing views in the observed ones.

Silberer and Lapata (2014) similarly used a split multiview autoencoder to accomplish the task of semantic concept grounding. In training the joint representation, they not only use the reconstruction errors of each view as the objective, but also introduce a term in the loss function to predict the object label with the representation.

8.2.2.2 Representation Fusion

A classic neural network consists of three layers namely, the input layer, the output layer, and the middle layer (also called the hidden layer), which are linked by nonlinear activation functions. Due to the multilayer nature of deep neural networks, each successive layer is hypothesized to represent the data in a more abstract way. Hence it is common to use the final or penultimate neural layers as a form of data representation.

To construct a multiview representation using neural networks each view starts with several individual neural layers followed by a hidden layer that projects the views into a joint space. The joint multiview representation is then passed through multiple hidden layers or used directly for prediction.

Ouyang et al. (2014) proposed to build a multisource deep model in order to extract nonlinear representation from three important information sources for human pose estimation. In order to make the learned representation more accurate and compact, and reduce the redundant information lying in the multimodal representations and incorporate different complexities of different modalities in the deep models, (Wang et al. 2015a) proposed a model, which fully exploits intra-modality and intermodality correlations and imposes an orthogonal regularizer on the weighting matrices of the model. They found that a better representation can be attained with different numbers of layers for different modalities, due to their different complexities. Jiang et al. (2018) proposed a unified framework that jointly exploits relationships between the views and relationships between the classes for improved categorization performance by imposing regularizations in the learning process of a deep neural network (DNN). Moreover, in order to cope with the challenge of a limited amount of annotated data, (Nojavanasghari et al. 2016) proposed a deep multiview fusion architecture that can use the additional information of each view for prediction.

8.2.3 Sequential Models

8.2.3.1 Conditional Random Field

Conditional random fields (CRFs) models paired sequential data using a conditional probability distribution of the output variables with given input variables. The characteristic of the CRFs is that the output variable constitutes the Markov random fields. CRFs can simulate arbitrary characteristics of the observed sequence and the model can adapt to overlapping features. CRFs can be expressed through undirect graphical models which provide a simple way to visualize the structure of probabilistic models and relationship between nodes.

Song et al. (2012) presented multiview latent variable discriminative models—multiview hidden CRFs (MV-HCRFs) and multiview latent-dynamic CRFs (MV-LDCRFs) that jointly learn both view-shared and view-specific substructures. Their approach makes the assumption that observed features from different views are conditionally independent given their respective sets of latent variables, and uses disjoint sets of latent variables to capture the interaction between views.

The task of the MV-HCRFs is to model the relationship between observed data $\mathbf{x}$ and labels $\mathbf{y}$ by constructing latent variables $\mathbf{h}$. Given $\mathbf{x} = \{\mathbf{x}^1, \ldots, \mathbf{x}^C\}$, $\mathbf{y} = [y_1, \ldots, y_T]$ and parameters $\boldsymbol{\theta} \in \Theta$, the conditional probability distribution $p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$ takes the form as

$$p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h}|\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\theta})} \sum_{\mathbf{h}} \exp\left\{\langle \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h}), \boldsymbol{\theta} \rangle\right\}, \tag{8.13}$$

where $\langle \cdot, \cdot \rangle$ denotes the vector inner product, and the denominator is

$$Z(\mathbf{x}, \boldsymbol{\theta}) = \sum_{\mathbf{y}} \sum_{\mathbf{h}} \exp\{\langle \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h}), \boldsymbol{\theta} \rangle\}. \tag{8.14}$$

Suppose that the conditional probability distribution $p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$ factorizes according to an undirected graph G and $\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h})$ is a vector of feature functions of the clique potentials over the maximal cliques of G. The feature function vector $\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h})$ plays an important role in improving the performance of the CRFs model. Here, we give a universal representation of the feature functions for CRFs where the size of all the largest clique is assumed to be two. Taking the clique of two nodes $\{y_i, y_j\}$ in G as an example, there are two kinds of clique potentials constructed by feature functions, where the edge clique $\mathscr{E}_{ij}$ has its associated pairwise potential and the node clique $\mathscr{N}_i$ has its associated unary potential. Thus, the whole feature function vector can be expressed as

$$\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \left[F_{\mathscr{E}_{ij},m}(\mathbf{x}, \mathbf{y}, \mathbf{h}), \ldots, F_{\mathscr{N}_i,n}(\mathbf{x}, \mathbf{y}, \mathbf{h}), \ldots \right]^\top. \tag{8.15}$$

The subscript $\{\mathscr{E}_{ij}, m\}$ denotes the mth feature function of the pairwise potential for the edge $\mathscr{E}_{ij}$ and the subscript $\{\mathscr{N}_i, n\}$ denotes the nth different feature function of the unary potential for the node $\mathscr{N}_i$. If suppose that the total number of various feature functions over all cliques is K, then $\mathbf{F}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^K$.

The feature function $\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{h})$ encodes both view-shared and view-specific substructures.

8.2.3.2 Recurrent Neural Network

A recurrent neural network (RNN) (Mikolov et al. 2010) is a neural network which processes a variable-length sequence $\mathbf{x} = (x_1, \ldots, x_T)$ through hidden state representation $\mathbf{h}$. At each time step t, the hidden state h_t of the RNN is estimated by

$$h_t = f(h_{t-1}, x_t) \tag{8.16}$$

where f is a nonlinear activation function and is selected based on the requirement of data modeling. For example, a simple case may be a common element-wise logistic sigmoid function and a complex case may be a long short-term memory (LSTM) unit (Hochreiter and Schmidhuber 1997).

An RNN is known as its ability of learning a probability distribution over a sequence by being trained to predict the next symbol in a sequence. In this training, the prediction at each time step t is decided by the conditional distribution $p(x_t|x_{t-1}, \ldots, x_1)$. For example, a multinomial distribution can be learned as output with a softmax activation function.

$$p(x_{t,j} = 1|x_{t-1}, \ldots, x_1) = \frac{\exp(w_j h_t)}{\sum_{j'=1}^{K} \exp(w_{j'} h_t)}$$

where $j = 1, ..., K$ denotes the possible symbol components and w_j are the corresponding rows of a weight matrix W. Further, based on the above probabilities, the probability of the sequence $\mathbf{x}$ can be computed as

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots, x_1) \tag{8.17}$$

With this learned distribution, it is straightforward to generate a new sequence by iteratively generating a symbol at each time step.

The usage of RNN representations is not limited to the single view. Cho et al. (2014) proposed an RNN encoder–decoder model by using RNN to connect multi-view sequence. This neural network first encodes a variable-length source sequence into a fixed-length vector representation and then decodes this fixed-length vector representation back into a variable-length target sequence. In fact, it is a general method to learn the conditional distribution over an output sequence conditioned on another input sequence, e.g., $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, where the input and output sequence lengths T and T' can be different. In particular, the encoder of the proposed model is an RNN which sequentially encodes each symbol of an input sequence $\mathbf{x}$ into the corresponding hidden state according to (8.16). After reading the end of the input sequence, a summary hidden state of the whole source sequence $\mathbf{c}$ is acquired. The decoder of the proposed model is another RNN which is exploited to generate the target sequence by predicting the next symbol y_t with the hidden state h_t. Based on the recurrent property, both y_t and h_t are also conditioned on y_{t-1} and on the summary $\mathbf{c}$ of the input sequence. Thus, the hidden state of the decoder at time t is computed by

$$h_t = f(h_{t-1}, y_{t-1}, \mathbf{c}) \tag{8.18}$$

and the conditional distribution of the next symbol is

$$p(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g(h_t, y_{t-1}, \mathbf{c}) \tag{8.19}$$

where g is an activation function and produces valid probabilities with a softmax function. The main idea of the RNN-based encoder–decoder framework can be summarized by jointly training two RNNs to maximize the conditional log-likelihood (Cho et al. 2014)

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\mathbf{y}_n|\mathbf{x}_n) \tag{8.20}$$

where θ is the set of the model parameters and each pair $(\mathbf{x}_n, \mathbf{y}_n)$ consists of an input sequence and an output sequence from the training set. The model parameters can be estimated by a gradient-based algorithm.

Sutskever et al. (2014) also presented a general end-to-end approach for multimodal sequence to sequence learning based on deep LSTM networks, which is helpful for solving problems with long-term time dependences (Bengio et al. 1994; Hochreiter and Schmidhuber 1997). The goal of the LSTM in this method is to estimate the conditional probability $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$ where $(x_1, \ldots, x_T)$is an input sequence and $(y_1, \ldots, y_{T'})$ is its corresponding output sequence whose length T' may differ from T. The LSTM computes this conditional probability by first obtaining the fixed- dimensional representation v of the input sequence $(x_1, \ldots, x_T)$ given by the last hidden state of the LSTM, and then computing the probability of $(y_1, \ldots, y_{T'})$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $x_1, \ldots, x_T$:

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \ldots, y_{t-1}) \tag{8.21}$$

where each $p(y_t|v, y_1, \ldots, y_{t-1})$ distribution is represented with a softmax over all the words in the vocabulary.

Besides, multiview RNNs have been widely applied in image captioning (Karpathy and Fei-Fei 2015; Mao et al. 2015), video captioning (Donahue et al. 2015; Venugopalan et al. 2015, 2014), visual question answering (Antol et al. 2015), and information retrieval (Palangi et al. 2016).

By applying the attention mechanism (Bahdanau et al. 2015) to visual recognition (Ba et al. 2015; Mnih et al. 2014; Xu et al. 2015a), introduced an attention-based multiview RNN model, which trains the multiview RNN in a deterministic manner using the standard backpropagation. The advantage of the proposed model lies in attending to salient part of an image while generating its caption.

8.3 Complementary Structured Space

In this section, we mainly introduce some methods of constructing complementary structured space. Although those methods do not focus on the joint representation, they separately process the data for each view. And they emphasize certain similarity constraints to form the relationship between views. Mathematically, constructing a complementary structured space $\mathscr{K}$ is expressed as follows:

$$f(\mathbf{x}^1) \overset{\mathscr{K}}{\sim} g(\mathbf{x}^2). \tag{8.22}$$

Each view has its own corresponding projection functions (f and g above) that map each view to the same multiview space. Although the projection to the multiview space is independent for each view, each view can be linked and restricted through the resultant space $\mathscr{K}$.

8.3.1 Deep Canonical Correlation Analysis

8.3.1.1 Reviews of CCA and KCCA

Canonical correlation analysis (Hotelling 1936) has become increasingly popular for its capability of effectively modeling the relationship between two or more sets of variables. From the perspective of multiview representation learning, CCA computes a shared embedding of both sets of variables through maximizing the correlations among the variables between the two sets.

Given a pair of datasets $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$, CCA tends to find linear projections $\mathbf{w}_x$ and $\mathbf{w}_y$, which make the corresponding examples in the two datasets maximally correlated in the projected space. The correlation coefficient between the two datasets in the projected space is given by

$$\rho = \mathrm{corr}(\mathbf{w}_x^\top \mathbf{X}, \mathbf{w}_y^\top \mathbf{Y}) = \frac{\mathrm{cov}(\mathbf{w}_x^\top \mathbf{X}, \mathbf{w}_y^\top \mathbf{Y})}{\sqrt{\mathrm{var}(\mathbf{w}_x^\top \mathbf{X}), \mathrm{var}(\mathbf{w}_y^\top \mathbf{Y})}} = \frac{\mathbf{w}_x^\top \mathbf{C}_{xy} \mathbf{w}_y}{\sqrt{(\mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x)(\mathbf{w}_y^\top \mathbf{C}_{yy} \mathbf{w}_y)}} \tag{8.23}$$

where covariance matrix $\mathbf{C}_{xy}$ is defined as

$$\mathbf{C}_{xy} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i - \mathbf{m}_x)(\mathbf{y}_i - \mathbf{m}_y)^\top \tag{8.24}$$

with $\mathbf{m}_x$ and $\mathbf{m}_y$ being the means from the two views, respectively

$$\mathbf{m}_x = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i, \quad \mathbf{m}_y = \frac{1}{m} \sum_{i=1}^{m} \mathbf{y}_i \tag{8.25}$$

and $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ can be defined analogously. Since the scales of $\mathbf{w}_x$ and $\mathbf{w}_y$ have no effects on the value of (8.23), each of the two factors in the denominator can be constrained to have value 1. This results in another widely used objective for CCA

$$\begin{aligned} (\mathbf{w}_x^\star, \mathbf{w}_y^\star) = \arg\max_{\mathbf{w}_x, \mathbf{w}_y} \ & \mathbf{w}_x^\top \mathbf{C}_{xy} \mathbf{w}_y \\ \text{s.t.} \ & \mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x = 1, \ \mathbf{w}_y^\top \mathbf{C}_{yy} \mathbf{w}_y = 1. \end{aligned} \tag{8.26}$$

By formulating the Lagrangian dual of (8.26), it can be shown that the solution to (8.26) is equivalent to solving a pair of generalized eigenvalue problems,

$$\begin{aligned} \mathbf{C}_{xy} \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \mathbf{w}_x &= \lambda^2 \mathbf{C}_{xx} \mathbf{w}_x, \\ \mathbf{C}_{yx} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \mathbf{w}_y &= \lambda^2 \mathbf{C}_{yy} \mathbf{w}_y. \end{aligned} \tag{8.27}$$

A nonlinear extension of CCA, kernel CCA has been successfully applied in many situations. The key idea of kernel CCA lies in embedding the data into a higher dimensional feature space $\phi_x : \mathscr{X} \to \mathscr{H}$, where $\mathscr{H}_x$ is the reproducing kernel

Hilbert space (RKHS) associated with the real numbers, $k_x : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ and $k_x(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi_x(\mathbf{x}_i), \phi_x(\mathbf{x}_j) \rangle$. k_y, $\mathscr{H}_y$, and ϕ_y can be defined similarly.

Correspondingly, replace vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ in the previous CCA formulation (8.23) with $f_x = \sum_i \alpha_i \phi_x(\mathbf{x}_i)$ and $f_y = \sum_i \beta_i \phi_y(\mathbf{y}_i)$, respectively, and replace the covariance matrices. The kernel CCA objective can be written as follows:

$$\rho = \text{corr}(f_x(\mathbf{X}), f_y(\mathbf{Y})) = \frac{\text{cov}(f_x(\mathbf{X}), f_y(\mathbf{Y}))}{\sqrt{\text{var}(f_x(\mathbf{X})), \text{var}(f_y(\mathbf{Y}))}} \tag{8.28}$$

Let $\mathbf{K}_x$ denote the kernel matrix such that $\mathbf{K}_x = \mathbf{H}\tilde{\mathbf{K}}_x\mathbf{H}$, where $[\tilde{\mathbf{K}}_x]_{ij} = k_x(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is a centering matrix and $\mathbf{1} \in \mathbb{R}^n$ is a vector of all ones. $\mathbf{K}_y$ is defined similarly. Further, substitute them into (8.28) and formulate the objective of kernel CCA as the following optimization problem:

$$(\mathbf{w}_x^\star, \mathbf{w}_y^\star) = \arg\max_{\mathbf{w}_x, \mathbf{w}_y} \mathbf{w}_x^\top \mathbf{K}_x \mathbf{K}_y \mathbf{w}_y \tag{8.29}$$

$$\text{s.t.} \quad \mathbf{w}_x^\top \mathbf{K}_x^2 \mathbf{w}_x + \varepsilon_x \mathbf{w}_x^\top \mathbf{K}_x \mathbf{w}_x = 1,$$
$$\mathbf{w}_y^\top \mathbf{K}_y^2 \mathbf{w}_y + \varepsilon_y \mathbf{w}_y^\top \mathbf{K}_y \mathbf{w}_y = 1. \tag{8.30}$$

Similar to the optimized case of CCA, by formulating the Lagrangian dual of (8.29) with the constraints in (8.29), it can be shown that the solution to (8.29) is also equivalent to solving a pair of generalized eigenproblems,

$$(\mathbf{K}_x + \varepsilon_x \mathbf{I})^{-1} \mathbf{K}_y (\mathbf{K}_y + \varepsilon_y \mathbf{I})^{-1} \mathbf{K}_x \mathbf{w}_x = \lambda^2 \mathbf{w}_x,$$
$$(\mathbf{K}_y + \varepsilon_y \mathbf{I})^{-1} \mathbf{K}_x (\mathbf{K}_x + \varepsilon_x \mathbf{I})^{-1} \mathbf{K}_y \mathbf{w}_y = \lambda^2 \mathbf{w}_y. \tag{8.31}$$

8.3.1.2 Deep Canonical Correlation Analysis

While deep networks are commonly used for learning classification labels or mapping to another vector space with supervision, here they can learn nonlinear transformations of two datasets to a space in which the data is highly correlated, just as KCCA does. Andrew et al. (2013) proposed a DNN extension of CCA termed deep CCA (DCCA; see Fig. 8.3. In DCCA, two DNNs $\mathbf{f}$ and $\mathbf{g}$ are used to extract nonlinear features for each view and the canonical correlation between the extracted features $\mathbf{f}(\mathbf{X})$ and $\mathbf{g}(\mathbf{Y})$ is maximized:

$$(\mathbf{w}_x^\star, \mathbf{w}_y^\star, \theta_\mathbf{f}^\star, \theta_\mathbf{g}^\star) = \underset{\mathbf{w}_x, \mathbf{w}_y, \theta_\mathbf{f}, \theta_\mathbf{g}}{\arg\max} \frac{1}{N} \text{tr}(\mathbf{w}_x^\top \mathbf{f}(\mathbf{X}) \mathbf{g}(\mathbf{Y})^\top \mathbf{w}_y) \tag{8.32}$$

$$\text{s.t.} \quad \mathbf{w}_x^\top \left(\frac{1}{N} \mathbf{f}(\mathbf{X}) \mathbf{f}(\mathbf{X})^\top + r_x \mathbf{I} \right) \mathbf{w}_x = \mathbf{I},$$
$$\mathbf{w}_y^\top \left(\frac{1}{N} \mathbf{g}(\mathbf{Y}) \mathbf{g}(\mathbf{Y})^\top + r_y \mathbf{I} \right) \mathbf{w}_y = \mathbf{I},$$
$$\mathbf{w}_{xi}^\top \mathbf{f}(\mathbf{X}) \mathbf{g}(\mathbf{Y})^\top \mathbf{w}_{yj} = 0, \quad \text{for} \quad i \neq j,$$

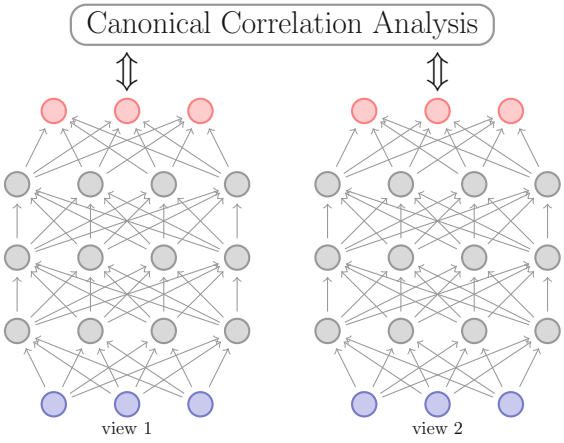


Fig. 8.3 The framework of deep CCA, where the output layers of two deep networks are maximally correlated

where $\mathbf{w}_x = [\mathbf{w}_{x1}, \ldots, \mathbf{w}_{xL}]$ and $\mathbf{w}_y = [\mathbf{w}_{y1}, \ldots, \mathbf{w}_{yL}]$ are the CCA directions that project the DNN outputs and $(r_x, r_y) > 0$ are regularization parameters for sample covariance estimation (Hardoon et al. 2004). In DCCA, $\mathbf{w}_x^\top \mathbf{f}(\cdot)$ is the final projection mapping used for testing. The same properties that may account for deep networks success in other tasks high model complexity and the ability to concisely represent a hierarchy of features for modeling real-world data distributions could be particularly useful in a setting where the output space is significantly more complex than a single label.

Deep CCA and its extensions have been widely applied in learning representation tasks in which multiple views of data are provided. Yan and Mikolajczyk (2015) learned a joint latent space for matching images and captions with a deep CCA framework, which adopts a GPU implementation and could deal with overfitting. To exploit the multilingual context when learning word embeddings, (Lu et al. 2015) learned deep nonlinear embeddings of two languages using the deep CCA.

8.3.2 Methods Based on Autoencoders

8.3.2.1 Deep Canonically Correlated Autoencoders (DCCAE)

Inspired by both CCA and reconstruction-based objectives, a new variant (Wang et al. 2015b) was proposed that consists of two autoencoders and optimizes the combination of canonical correlation between the learned bottleneck representations and the reconstruction errors of the autoencoders. In other words, the optimization is the following objective:

$$\begin{aligned}\min_{\mathbf{W_f},\mathbf{W_g},\mathbf{W_p},\mathbf{W_q},\mathbf{U},\mathbf{V}} \quad & -\frac{1}{N}\text{tr}(\mathbf{U}^\top \mathbf{f}(\mathbf{X})\mathbf{g}(\mathbf{Y})^\top \mathbf{V}) \\ & +\frac{\lambda}{N}\sum_{i=1}^{N}(\|\mathbf{x}_i - \mathbf{p}(\mathbf{f}(\mathbf{x}_i))\|^2 + \|\mathbf{y}_i - \mathbf{q}(\mathbf{g}(\mathbf{x}_i))\|^2) \qquad (8.33) \\ \text{s.t.} \quad & \mathbf{U}^\top \left(\frac{1}{N}\mathbf{f}(\mathbf{X})\mathbf{f}(\mathbf{X})^\top + r_x\mathbf{I}\right)\mathbf{U} = \mathbf{I}, \\ & \mathbf{V}^\top \left(\frac{1}{N}\mathbf{g}(\mathbf{Y})\mathbf{g}(\mathbf{Y})^\top + r_y\mathbf{I}\right)\mathbf{V} = \mathbf{I}, \\ & \mathbf{u}_i^\top \mathbf{f}(\mathbf{X})\mathbf{g}(\mathbf{Y})^\top \mathbf{v}_j = 0, \quad \text{for} \quad i \neq j,\end{aligned}$$

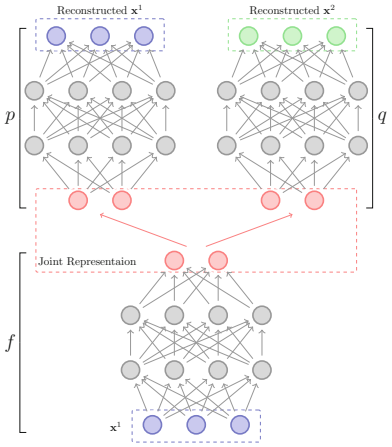
where $\lambda > 0$ is a trade-off parameter, $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_L]$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_L]$ are the CCA directions that project the DNN outputs and $(r_x, r_y) > 0$ are regularization parameters for sample covariance estimation. In DCCAE, $\mathbf{U}^\top \mathbf{f}(\cdot)$ is the final projection mapping used for testing. Similarly to DCCA, we can also apply stochastic optimization to the DCCAE objective. We can explore the minibatch sizes for each term separately (by training DCCA and an autoencoder) and select them using a validation set. The stochastic gradient is then the sum of the gradient for the DCCA term (usually estimated using large minibatches) and the gradient for the autoencoder term (usually estimated using small minibatches).

CCA maximizes the mutual information between the projected views for certain distributions, while training an autoencoder to minimize reconstruction error amounts to maximizing a lower bound on the mutual information between inputs and learned features (Vincent et al. 2010). The DCCAE objective offers a trade-off between the information captured in the (input, feature) mapping within each view on the one hand, and the information in the (feature, feature) relationship across views on the other hand. Intuitively, this is the same principle as the information bottleneck method, and indeed, in the case of Gaussian variables, the information bottleneck method finds the same subspaces as CCA.

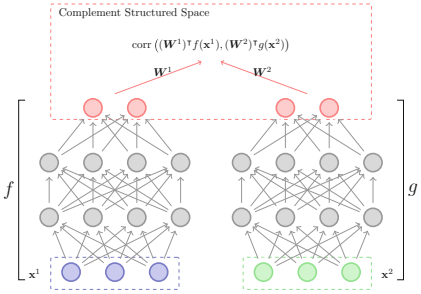
8.3.2.2 Correlated Autoencoders (Corr-AE)

In this model, the CCA term in the DCCAE objective is replaced with the sum of the scalar correlations between the pairs of learned dimensions across views, which is an alternative measure of agreement between views. In other words, the feature dimensions within each view are not constrained to be uncorrelated with each other. This model is intended to test how important the original CCA constraint is. It is called correlated autoencoders (Corr-AE), as shown in Fig. 8.4c. Its objective can be equivalently written in a constrained form as

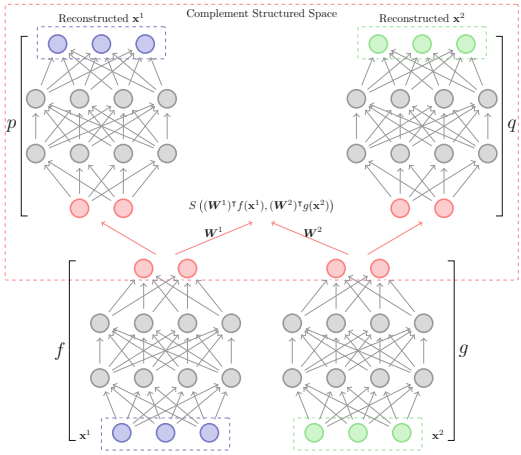
Fig. 8.4 The difference of multiview representation learning method based on DNN



(a) Split Multiview Autoencoders



(b) Deep CCA



(c) DCCAE/CorrAE/DistAE

$$\begin{aligned}\min_{\mathbf{W_f},\mathbf{W_g},\mathbf{W_p},\mathbf{W_q},\mathbf{U},\mathbf{V}} \quad & -\frac{1}{N}\mathrm{tr}(\mathbf{U}^\top \mathbf{f}(\mathbf{X})\mathbf{g}(\mathbf{Y})^\top \mathbf{V}) \\ & +\frac{\lambda}{N}\sum_{i=1}^{N}(\|\mathbf{x}_i - \mathbf{p}(\mathbf{f}(\mathbf{x}_i))\|^2 + \|\mathbf{y}_i - \mathbf{q}(\mathbf{g}(\mathbf{x}_i))\|^2) \qquad (8.34) \\ \text{s.t.} \quad & \mathbf{u}_i^\top \mathbf{f}(\mathbf{X})\mathbf{f}(\mathbf{X})^\top \mathbf{u}_i = \mathbf{v}_i^\top \mathbf{g}(\mathbf{Y})\mathbf{g}(\mathbf{Y})^\top \mathbf{v}_i = N, \quad 1 \le i \le L\end{aligned}$$

where $\lambda > 0$ is a trade-off parameter. It is clear that the constraint set in (8.34) is a relaxed version of that of (8.33).

Corr-AE is similar to the model of (AP et al. 2014), who tries to learn vectorial word representations using parallel corpora from two languages. They used DNNs in each view (language) to predict the bag-of-words representation of the input sentences, or that of the paired sentences from the other view, while encouraging the learned bottleneck layer representations to be highly correlated.

Further, (Feng et al. 2014) also proposed a correspondence autoencoder (Corr-AE) via constructing correlations between hidden representations of two unimodal deep autoencoders. The details of the architecture of the basic Corr-AE is shown in Fig. 8.4c. As illustrated in Fig. 8.4c, this Corr-AE architecture consists of two subnetworks (each a basic deep autoencoder) that are connected by a predefined similarity measure on a specific code layer. Each subnetwork in the Corr-AE serves for each modality.

Let $\mathbf{f}(\mathbf{X}; \mathbf{W_f})$ and $\mathbf{g}(\mathbf{Y}; \mathbf{W_g})$ denote the mapping from the inputs $\{\mathbf{X}, \mathbf{Y}\}$ to the code layers, respectively. $\theta = \{\mathbf{W_f}, \mathbf{W_g}\}$ denotes the weight parameters in these two networks. Here the similarity measure between the ith pair of image feature $\mathbf{x}_i$ and the given text feature $\mathbf{y}_i$ is defined as follows:

$$C(\mathbf{x}_i, \mathbf{y}_i; \theta) = \|\mathbf{f}(\mathbf{x}_i; \mathbf{W_f}) - \mathbf{g}(\mathbf{y}_i; \mathbf{W_g})\|_2^2$$

where $\mathbf{f}$ and $\mathbf{g}$ are logistic activation functions.

Consequently, the loss function on any pair of inputs is then defined as follows:

$$L(\mathbf{x}_i, \mathbf{y}_i; \theta) = (1-\alpha)\big(L_I(\mathbf{x}_i, \mathbf{y}_i; \theta) + L_T(\mathbf{x}_i, \mathbf{y}_i; \theta)\big) + \alpha L_C(\mathbf{x}_i, \mathbf{y}_i; \theta) \qquad (8.35)$$

where

$$\begin{aligned} L_I(\mathbf{x}_i, \mathbf{y}_i; \theta) &= \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 \\ L_T(\mathbf{x}_i, \mathbf{y}_i; \theta) &= \|\mathbf{y}_i - \hat{\mathbf{y}}_i\| \\ L_C(\mathbf{x}_i, \mathbf{y}_i; \theta) &= C(\mathbf{x}_i, \mathbf{y}_i; \theta). \end{aligned}$$

L_I and L_T are the losses caused by data reconstruction errors for the given inputs of the two subnetworks, specifically image and text modalities. L_C is the correlation loss and α is a parameter for trade-off between two groups of objectives. $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_i$ are the reconstruction data from $\mathbf{x}_i$ and $\mathbf{y}_i$, respectively.

Overall, optimizing the objective in (8.35) enables Corr-AE to learn similar representations from bimodal features. Besides, based on two other multiview autoencoders, Corr-AE is extended to two other correspondence deep models, called Corr-Cross-AE and Corr-Full-AE (Feng et al. 2015).

8.3.2.3 Minimum-Distance Autoencoders (DistAE)

The CCA objective can be seen as minimizing the distance between the learned projections of the two views, while satisfying the whitening constraints for the projections (Hardoon et al. 2004). The constraints complicate the optimization of CCA-based objectives. This observation motivates people to consider additional objectives that decompose into sums over training examples, while maintaining the intuition of the CCA objective as a reconstruction error between two mappings. Wang et al. (2015b) were proposed a variant as minimum-distance autoencoders (DistAE).

The first variant DistAE-1 optimizes the following objective:

$$\begin{aligned}\min_{\mathbf{W_f},\mathbf{W_g},\mathbf{W_p},\mathbf{W_q}} \quad & \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathbf{f}(\mathbf{x}_i)-\mathbf{g}(\mathbf{y}_i)\|^2}{\|\mathbf{f}(\mathbf{x}_i)\|^2+\|\mathbf{g}(\mathbf{y}_i)\|^2} \\ & +\frac{\lambda}{N}\sum_{i=1}^{N}(\|\mathbf{x}_i-\mathbf{p}(\mathbf{f}(\mathbf{x}_i))\|^2+\|\mathbf{y}_i-\mathbf{q}(\mathbf{g}(\mathbf{x}_i))\|^2) \end{aligned} \tag{8.36}$$

which is a weighted combination of the reconstruction errors of two autoencoders and the average discrepancy between the projected sample pairs. The denominator of the discrepancy term is used to keep the optimization from improving the objective by simply scaling down the projections (although they can never become identically zero due to the reconstruction terms). This objective is unconstrained and is the empirical average of the loss incurred at each training sample, and so normal SGD applies using small (or any size) minibatches.

The second variant DistAE-2 optimizes a somewhat different objective:

$$\begin{aligned}\min_{\mathbf{W_f},\mathbf{W_g},\mathbf{W_p},\mathbf{W_q},\mathbf{A},\mathbf{b}} \quad & \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{f}(\mathbf{x}_i)-\mathbf{A}\mathbf{g}(\mathbf{y}_i)-\mathbf{b}\|^2 \\ & +\frac{\lambda}{N}\sum_{i=1}^{N}(\|\mathbf{x}_i-\mathbf{p}(\mathbf{f}(\mathbf{x}_i))\|^2+\|\mathbf{y}_i-\mathbf{q}(\mathbf{g}(\mathbf{x}_i))\|^2) \end{aligned} \tag{8.37}$$

where $\mathbf{A}\in\mathbb{R}^{L\times L}$ and $\mathbf{b}\in\mathbb{R}^{L}$. The underlying intuition is that the representation of the primary view can be linearly predicted from the representation of the other view. This relationship is motivated by the fact that when $\mathbf{g}(\mathbf{y})$ and $\mathbf{f}(\mathbf{x})$ are perfectly linearly correlated, then there exists an affine transformation that can map from one to the other. This approach, hence, alleviates the burden on $\mathbf{g}(\mathbf{y})$ being simultaneously predictive of the output and close to $\mathbf{f}(\mathbf{x})$ by itself.

8.3.3 Similarity Models

Another category of methods for constructing multiview complementary space with the representation of each view is referred to herein as the similarity model. Complementary spaces are constructed by constraining the representation of each view to be similar or not. The objective of these methods is to minimize the distance between views in a complementary constructed space. The representation distances for different views around the same topic are closer than for different views around different topics (Frome et al. 2013).

8.3.3.1 WSABIE

The earliest similarity model to construct such a complementary constructed space is the web scale annotation by image embedding (WSABIE) model (Weston et al. 2010, 2011), where a complementary constructed space was constructed with images and their annotations. They used the WSABIE model to construct a simple linear mapping from image and text features such that the corresponding representations of the annotations and the images have a higher inner product than a non-corresponding one between them.

Specifically, WSABIE learns a mapping into a feature space where images and annotations are both represented. The mapping functions are therefore different, but are learnt jointly to optimize the supervised loss of interest for the final tasks. The model starts with a representation of images $x \in \mathbb{R}^d$ and a representation of annotations $i \in \mathscr{Y} = \{1, \ldots, Y\}$, indices into a dictionary of possible annotations. It then learns a mapping from the image feature space to the joint space $\mathbb{R}^D$:

$$\Phi_I(x) : \mathbb{R}^d \rightarrow \mathbb{R}^D, \tag{8.38}$$

while jointly learning a mapping for annotations:

$$\Phi_W(i) : \{1, \ldots, Y\} \rightarrow \mathbb{R}^D. \tag{8.39}$$

These are chosen to be linear maps, i.e., $\Phi_I(x) = Vx$ and $\Phi_W(i) = W_i$, where W_i indexes the ith column of a $D \times Y$ matrix, but potentially any mapping could be used. In this method, WSABIE uses sparse high-dimensional feature vectors of bags-of-visual terms for image vectors x and each annotation has its own learnt representation (even if, for example, multi-word annotations share words).

The goal is, for a given image, to rank the possible annotations such that the highest ranked annotations best describe the semantic content of the image:

$$f_i(x) = \Phi_W(i)^\mathsf{T} \Phi_I(x) = W_i^\mathsf{T} V x \tag{8.40}$$

where the possible annotations i are ranked according to the magnitude of $f_i(x)$, largest first, and our family of models have constrained norms:

$$\|V_i\|_2 \leq C, \quad i = 1, \ldots, d, \tag{8.41}$$

$$\|W_i\|_2 \leq C, \quad i = 1, \ldots, Y. \tag{8.42}$$

which act as a regularizer.

With the task of ranking labels $i \in \mathscr{Y}$ given an example x, labeled pairs (x, y) will be provided for training where only a single annotation $y_i \in \mathscr{Y}$ is labeled correctly. Let $f(x) \in \mathbb{R}^Y$ be a vector function providing a score for each of the labels, where $f_i(x)$ is the value for label i. A class of ranking loss functions was recently defined in Usunier et al. (2009) as

$$loss(f(x), y) = L\left(\text{rank}_y(f(x))\right) \tag{8.43}$$

$$\text{rank}_y(f(x)) = \sum_{i \neq y} \mathbf{1}\left(f_i(x) \geq f_y(x)\right) \tag{8.44}$$

where $\text{rank}_y(f(x))$ is the rank of the true label y given by (8.40) and $\mathbf{1}(\cdot)$ is the indicator function, and $L(\cdot)$ transforms this rank into a loss function. This class of functions allows one to define different choices of $L(\cdot)$ with different minimizers.

Then WSABIE uses stochastic gradient descent to optimize a ranking objective function and is evaluated on datasets with ten million training examples. WSABIE learns a common embedding for visual and tag features. However, it is only a two-view model and thus does not explicitly represent the distinction between the tags used to describe the image and the underlying image content. Also, WSABIE is not explicitly designed for multi-label annotation, just evaluated on datasets whose images come with single labels (or single paths in a label hierarchy).

8.3.3.2 DeViSE

More recently, deep cross-view embedding models have become increasingly popular in the applications including cross-media retrieval and multimodal distributional semantic learning. Their advantage is that they can learn the similarities between views in an end-to-end model. Frome et al. (2013) proposed a deep visual-semantic embedding model (DeViSE), which connects two deep neural networks by a cross-modal mapping. Similar to WSABIE, DeViSE proposed by Frome et al. (2013) also uses similar inner product and rank loss functions. In addition, DeViSE uses more complex image data and word embedding. It is first initialized with a pretrained neural network language model and a pretrained deep visual-semantic model. Consequently, a linear transformation is exploited to map the representation at the top of the core visual model into the learned dense vector representations by the neural language model.

The core visual model, with its softmax prediction layer now removed, is trained to predict these vectors for each image, by means of a projection layer and a similarity metric. The projection layer is a linear transformation that maps the 4096-D representation at the top of the core visual model into the 500-D representation for the language model.

The choice of the loss function is proved to be important. A combination of dot-product similarity and hinge rank loss are used, such that the model is trained to produce a higher dot-product similarity between the visual model output and the vector representation of the correct label than between the visual output and other randomly chosen text terms. The per training example hinge rank loss is defined as

$$\text{loss}(\text{image}, \text{label}) = \sum_{i \neq \text{label}} \max[0, \text{margin} - \mathbf{t}_{\text{label}} M \mathbf{v}(\text{image}) + \mathbf{t}_i M \mathbf{v}(\text{image})] \tag{8.45}$$

where $\mathbf{v}(\text{image})$ is a column vector denoting the output of the top layer of the core visual network for the given image, M is the matrix of trainable parameters in the linear transformation layer, $\mathbf{t}_{\text{label}}$ is a row vector denoting learned embedding vector for the provided text label, and $\mathbf{t}_i$ are the embeddings of other text terms (Frome et al. 2013). This DeViSE model is trained by asynchronous stochastic gradient descent on a distributed computing platform.

Thereafter, (Kiros et al. 2014b) extended DeViSe to sentence and image to a complementary constructed space by using an LSTM model and a pairwise ranking loss to constrain the feature space. Socher et al. (2014) dealt with the same task by extending the language model to a dependency tree RNN to incorporate compositional semantics.

Furthermore, inspired by the success of DeViSE, (Norouzi et al. 2014) proposed a convex combination of semantic embedding model (ConSE) for mapping images into continuous semantic embedding spaces. Unlike DeViSE, this ConSE model keeps the softmax layer of the convolutional net intact. Given a test image, ConSE simply runs the convolutional classifier and considers the convex combination of the semantic embedding vectors from the top T predictions as its corresponding semantic embedding vector. Fang et al. (2015) developed a deep multimodal similarity model that learns two neural networks to map image and text fragments to a common vector representation.

With the development of multimodal distributed semantic models (Fang et al. 2015; Feng and Lapata 2010; Bruni et al. 2014; Yih et al. 2014), deep cross-modal mapping is naturally exploited to learn the improved multimodal distributed semantic representation. Lazaridou and Baroni (2015) introduced multimodal skip-gram models to extend the skip-gram model (Mikolov et al. 2013) by taking visual information into account. In this extension, for a subset of the target words, relevant visual evidence from natural images is presented together with the corpus contexts. In particular, the model is designed to encourage the propagation of visual information to all words via a deep cross-modal ranking so that it can improve image labeling and retrieval in the zero-shot setup, where the test concepts are never seen during

model training. Dong et al. (2016) presented Word2VisualVec, a deep neural network architecture that learns to predict a deep visual encoding of textual input, and thus enables cross-media retrieval in a visual space.

8.3.3.3 Joint Video-Language Model

Deep cross-view embedding models have become increasingly popular in the applications including cross-media retrieval (Frome et al. 2013) and multimodal distributed semantic learning (Kiros et al. 2014a). Xu et al. (2015b) proposed a unified framework of similarity space that jointly models video and the corresponding text sentence. In this joint architecture, the goal is to learn a function $f(\mathscr{V}) : \mathscr{V} \rightarrow \mathscr{T}$, where $\mathscr{V}$ represents the low-level features extracted from video, and $\mathscr{T}$ is the high-level text description of the video. A joint model $\mathscr{P}$ is designed to connect these two levels of information. It consists of three parts: compositional language model $M_L : T \rightarrow T_f$, deep video model $M_V : V \rightarrow V_f$, and a joint embedding model $E(V_f, T_f)$, such that

$$\mathscr{P} : M_V(V) \longrightarrow V_f \leftrightarrow E(V_f, T_f) \leftrightarrow T_f \longleftarrow M_L(T) \tag{8.46}$$

where V_f and T_f are the output of the deep video model and the compositional language model, respectively. In this joint embedding model, the distance between the outputs of the deep video model and those of the compositional language model in the joint space is minimized to make them aligned.

The compositional semantics language model (Xu et al. 2015b) captures high-level semantics information that can help constrain the visual model, and the visual model on the contrary, provides video evidence to support word selection. In joint embedding model, an objective function is defined to take into account video-language embedding error E_{embed} and language model reconstruction error E_{rec}. E_{embed} is based on least squares to implement both bottom-up and top-down paths simultaneously:

$$E_{\text{embed}}(V, T) = \|W_1 f(W_2 x_i) - \text{CLM}(m_{s,i}, m_{v,i}, m_{o,i} | W_m)\|_2^2$$

where $m_{s,i}$ represents the Sth word vector of the ith video, and $\text{CLM}(\cdot)$ is a novel compositional language model with recursive neural network. The objective function is:

$$J(V, T) = \sum_{i=1}^{N} \left(E_{\text{embed}}(V, T) + \sum_{p \in \mathbf{NT}} E_{\text{rec}}(p | W_m, W_r) \right) + r$$

where **NT** is the nonterminal set of a tree structure. Suppose the training set contains N videos, each paired with M sentences, and each ⟨Subject, Verb, Object⟩(SVO) triplet has t tree structures. Let θ be a general notation of model W_1, W_2, W_m or W_r, the regularization term $r = \frac{\lambda}{2}\|\theta\|^2$. In practice, the mean word vector over all sentences can be used as ground truth in each video to represent the training sentence.

According to this embedding model, the visual model and the compositional language model is updated jointly. One this method can implement both bottom-up and top-down paths in the embedded model, and thus the model is able to process both video-to-text and text-to-video tasks.

8.4 View Mapping

In this section, we mainly introduce some methods of mutual generation of views. Unlike the previous section exploring for a constraint space, this section emphasizes exploring relatively straightforward mappings between views.

Mathematically, views mapping is expressed as follows:

$$f(\mathbf{x}^1) \overset{\mathscr{K}}{\rightleftharpoons} g(\mathbf{x}^2) \tag{8.47}$$

where $\mathscr{K}$ can be an end-to-end neural networks or a dictionary. This type of methods look for $\mathscr{K}$ to build a direct connection between views.

8.4.1 Generative Models

8.4.1.1 Deep Conditional Generative Adversarial Networks

Recently, the generation of confrontation networks has received more and more attention in the field of machine learning. Because using a generator model to build a sequence (for example, a sentence or a image) or to generate a consistent sequence with time and structure is a challenging task. This has to led many early multiview mapping methods to rely on dictionary-based for generation. However, this situation has been changing with the advent of deep learning models that are capable of generating images and sequences.

More recently, a large amount of methods has been made in generating images using generative adversarial networks (Goodfellow et al. 2014), which have been used as an alternative to RNNs for image generation from the text. Reed et al. (2016c) and Reed et al. (2016b) trained a deep convolutional generative adversarial network (DC-GAN) conditioned on text features encoded by a hybrid character-level convolutional-recurrent neural network to generate images.

Generative adversarial networks (GANs) consist of a generator G and a discriminator D that compete in a two-player minimax game. The discriminator tries to distinguish real training data from synthetic images, and the generator tries to fool the discriminator. To learn a generator distribution p_g over data x, the generator builds a mapping function from a prior noise distribution $p_z(z)$ to data space as $G(z;\theta_g)$. And the discriminator, $D(x;\theta_d)$, outputs a single scalar representing the

probability that x comes from training data rather than p_g. Concretely, D and G play the following game on $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z)))] \quad (8.48)$$

where z is a noise vector drawn from, e.g., a Gaussian or uniform distribution. Goodfellow et al. (2014) proved that this minimax game has a global optimium precisely when generator distribution $p_g = p_{\text{data}}$, and that under mild conditions (e.g., G and D have enough capacity) p_g converges to p_{data}. In practice, in the start of training samples from D are extremely poor and rejected by D with high confidence. It has been found to work better in practice for the generator to maximize $\log(D(G(z)))$ instead of minimizing $\log(1 - D(G(z)))$.

Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y, such as class labels or data from other views. The objective function of a two-player minimax game would be as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z|y)))] \quad (8.49)$$

To encode visual content from text descriptions, (Reed et al. 2016c) used a convolutional and recurrent text encoder to learn a correspondence function between images and text features, following the approach of (Reed et al. 2016a) (and closely related to (Kiros et al. 2014b)). Sentence embeddings are learned by optimizing the following structured loss:

$$loss = \frac{1}{N} \sum_{n=1}^{N} \triangle(y_n, f_v(v_n)) + \triangle(y_n, f_t(t_n)) \quad (8.50)$$

where $\{(v_n, t_n, y_n),\ n = 1, \ldots, N\}$ is the training data set containing image information $v \in \mathscr{V}$, text descriptions $t \in \mathscr{T}$ and class labels $y \in \mathscr{Y}$, and $\triangle : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ is the 0–1 loss. Classifiers $f_v : \mathscr{V} \to \mathscr{Y}$ and $f_t : \mathscr{T} \to \mathscr{Y}$ are defined as

$$f_v(v) = \arg\max_{y \in \mathscr{Y}} E_{t \sim \mathscr{T}(y)}[\phi(v)^\intercal \psi(t))], \quad f_t(t) = \arg\max_{y \in \mathscr{Y}} E_{v \sim \mathscr{V}(y)}[\phi(v)^\intercal \psi(t))] \quad (8.51)$$

where ϕ is the image encoder (e.g., a deep convolutional network), ψ is the text encoder (e.g., a character-level CNN or LSTM), $\mathscr{T}(y)$ is the subset of $\mathscr{T}$ from class y, $\mathscr{V}(y)$ is the subset of $\mathscr{V}$ from class y, and the expectation is over text descriptions sampled uniformly from these subsets. Intuitively, the text encoder learns to produce a higher compatibility score with images of the correspondong class compared to any other class, and vice versa. Text encoding $\psi(t)$ is used by both generator $G(z, \psi(t))$ and discriminator $D(G(z, \psi(t)), \psi(t))$.

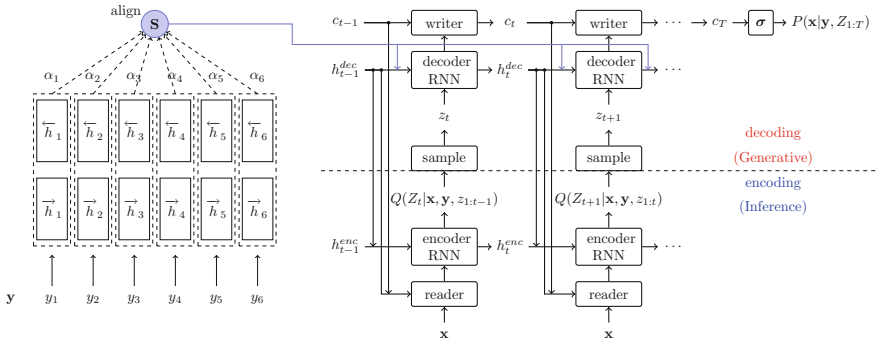


Fig. 8.5 AlignDRAW model for generating images by learning an alignment between the input captions and generating canvas. The caption is encoded using the Bidirectional RNN (left). The generative RNN used to generate the final image x (right). The inference RNN is used to compute approximate posterior Q over the latent sequence

8.4.1.2 Multiview Attention RNN

Mansimov et al. (2016) proposed a model that defines a generative process of images conditioned on captions (Fig. 8.5). In particular, captions are represented as a sequence of consecutive words and images are represented as a sequence of patches drawn on a canvas c_t over time $t = 1, \ldots, T$. The proposed model consists of four main parts: language model, image model, alignment model and post-processing model.

In the language model, bidirectional LSTM (Hochreiter and Schmidhuber 1997) computes a sequence of forward $\overrightarrow{h}^{lang}_{1...N}$ and backward $\overleftarrow{h}^{lang}_{1...N}$ hidden states, respectively, which are concatenated together into the sentence representation $h^{lang} = [\overrightarrow{h}^{lang}_{1...N}, \overleftarrow{h}^{lang}_{1...N}]$.

In the image model, to generate an image $\mathbf{x}$ conditioned on the caption information $\mathbf{y}$, they extended the DRAW network (Gregor et al. 2015) to include caption representation h^{lang} at each step. It iteratively computes the following set of equations over time $t = 1, \ldots, T$.

At each time step t, similar to the DRAW model, the encoder receives input from both the image x and from the previous decoder hidden vector h^{gen}. The inference recurrent network produces an approximate posterior $Q(Z_{1:T}|\mathbf{x}, \mathbf{y})$ via a *read* operator, which reads a patch from an input image $\mathbf{x}$ at each time step t. Specifically,

$$\hat{\mathbf{x}}_t = \mathbf{x} - \boldsymbol{\sigma}(c_{t-1}), \tag{8.52}$$

$$r_t = read(\mathbf{x}, \hat{\mathbf{x}}, h^{gen}_{t-1}), \tag{8.53}$$

$$h^{infer}_t = LSTM^{infer}(h^{infer}, [r_t, h^{gen}_{t-1}]), \tag{8.54}$$

$$Q(Z_t|\mathbf{x}, \mathbf{y}, Z_{1:t-1}) = \mathcal{N}\left(\mu(h^{infer}_t), \sigma(h^{infer}_t)\right), \tag{8.55}$$

where $\hat{\mathbf{x}}$ is the error image and h_0^{infer} is initialized to the learned bias b. Note that the inference $LSTM^{infer}$ takes as its input both the output of the read operator $r_t \in \mathbb{R}^{p \times p}$, which depends on the original input image $\mathbf{x}$, and the previous state of the generative decoder h^{gen}, which depends on the latent sample history $z_{1:t-1}$ and dynamic sentence representation s_{t-1}. Hence, the approximate posterior Q will depend on the input image $\mathbf{x}$, the corresponding caption $\mathbf{y}$, and the latent history $Z_{1:t-1}$, except for the first step $Q(Z_1|x)$, which depends only on $\mathbf{x}$.

At each time step, a sample $z_t \sim Q(Z_t|Z_{t-1})$ drawn from the latent distribution is passed as input to the decoder. The output h^{gen} of the decoder is added (via a *write* operation) to a cumulative canvas matrix c_t, which is ultimately used to reconstruct the image. The total number of time steps T consumed by the network before performing the reconstruction is a free parameter that must be specified in advance. The decoder is formulated as follows:

$$z_t \sim P(Z_t|Z_{1:t-1}) = \mathcal{N}\left(\mu(h_{t-1}^{gen}), \sigma(h_{t-1}^{gen})\right), \tag{8.56}$$

$$s_t = align(h_{t-1}^{gen}, h^{lang}), \tag{8.57}$$

$$h_t^{gen} = LSTM^{gen}(h_{t-1}^{gen}, [z_t, s_t]), \tag{8.58}$$

$$c_t = c_{t-1} + write(h_t^{gen}), \tag{8.59}$$

$$\tilde{\mathbf{x}} \sim P(\mathbf{x}|\mathbf{y}, Z_{1:T}) = \prod_i P(x_i|\mathbf{y}, Z_{1:T}) = \prod_i \text{Bern}(\boldsymbol{\sigma}(c_{T,i})). \tag{8.60}$$

The align model is used to compute the alignment between the input caption and intermediate image generative steps (Bahdanau et al. 2015). Given the caption representation from the language model, $h^{lang} = [h_1^{lang}, h_2^{lang}, \ldots, h_N^{lang}]$, the align operator outputs a dynamic sentence representation s_t at each step through a weighted sum using alignment probabilities $\alpha_{1,\ldots,N}^t$:

$$s_t = align(h_{t-1}^{gen}, h^{lang}) = \alpha_1^t h_1^{lang} + \alpha_2^t h_2^{lang} + \cdots + \alpha_N^t h_N^{lang}. \tag{8.61}$$

The corresponding alignment probability α_k^t for the k^{th} word in the caption is obtained using the caption representation h^{lang} and the current hidden state of the generative model h_{t-1}^{gen}:

$$\alpha_k^t \propto \exp\left(v^\intercal \tanh(U h_k^{lang} + W h_k^{gen} + b)\right). \tag{8.62}$$

Finally, images are generated by discarding the inference network and by sampling latent variables $Z_{1:t}$ from prior distribution. Images are sharpened using a deterministic adversarial network (Goodfellow et al. 2014; Denton et al. 2015) trained on residuals of a Laplacian pyramid.

This model is trained to maximize a variational lower bound $\mathscr{L}$ on the marginal likelihood of the correct image $\mathbf{x}$ given the input caption $\mathbf{y}$ using the stochastic gradient variational bayes (SGVB) (Kingma and Welling 2014) algorithm:

$$\mathscr{L} = \sum_{Z} Q(Z|\mathbf{x}, \mathbf{y}) \log P(\mathbf{x}|\mathbf{y}, Z) - D_{\mathrm{KL}}(Q(Z|\mathbf{x}, \mathbf{y}) \| P(Z|\mathbf{y})) \leq \log P(\mathbf{x}|\mathbf{y}) \tag{8.63}$$

The terms in the variational lower bound (8.63) can be rearranged using the law of total expectation. Therefore, the variational bound $\mathscr{L}$ is calculated as follows:

$$\mathscr{L} = \mathbb{E}_{Q(Z_{1:T}|\mathbf{y},\mathbf{x})} \left[\log p(\mathbf{x}|\mathbf{y}, Z_{1:T}) - \sum_{t=2}^{T} D_{\mathrm{KL}}(Q(Z_t|Z_{1:t-1}, \mathbf{y}, \mathbf{x}) \| P(Z_t|Z_{1:t-1}, \mathbf{y})) \right] - D_{\mathrm{KL}}(Q(Z_1|\mathbf{x}) \| P(Z_1)). \tag{8.64}$$

The expectation can be approximated by $\mathscr{L}$ Monte Carlo samples $\tilde{z}_{1:T}$ from $Q(Z_{1:T}|\mathbf{y}, \mathbf{x})$:

$$\mathscr{L} \approx \frac{1}{L} \sum_{l=1}^{L} \left[\log p(\mathbf{x}|\mathbf{y}, \tilde{z}^l_{1:T}) - \sum_{t=2}^{T} D_{\mathrm{KL}}(Q(Z_t|\tilde{z}^l_{1:t-1}, \mathbf{y}, \mathbf{x}) \| P(Z_t|\tilde{z}^l_{1:t-1}, \mathbf{y})) \right] - D_{\mathrm{KL}}(Q(Z_1|\mathbf{x}) \| P(Z_1)) \tag{8.65}$$

8.4.2 Retrieval-Based Methods

Retrieval-based methods tend to perform better in the semantic space than their single-view methods because they contain retrieved examples in a more meaningful and richer space. These search spaces reflect the commonality of the two views and are usually optimized for retrieval. In addition, they allow bidirectional mapping, which is not straightforward for single-view methods. However, retrieval-based methods need to manually build or learn such a semantic space, which usually depends on the existence of large training data sets of paired samples.

8.4.2.1 Deep Fragment Alignment

Karpathy et al. (2014) introduced a model for bidirectional retrieval of images and sentences through a deep, multiview embedding of visual and natural language data. Unlike previous models that directly map images or sentences into a common embedding space, the model works on a finer level and embeds fragments of images (objects) and fragments of sentences (typed dependency tree relations) into a common space. The proposed model consists of two parts: fragment embeddings and fragment alignment objective.

We first describe the neural networks that compute the image and sentence fragment embeddings. After detecting objects in every image, in order to extract and describe the set of entities that images are composed of, the embedding vectors

(representations) are computed based on the pixels I_b inside each bounding box as follows:

$$v = W_m[CNN_c(I_b)] + b_m, \tag{8.66}$$

where $CNN(I_b)$ transforms the pixels inside bounding box I_b into 4096-dimensional activations of the fully connected layer immediately before the classifier.

To extract and represent the set of visually identifiable entities described in a sentence and establish the intermodal relationships, the simplest approach might be to project every individual word directly into this embedding. However, this approach does not consider any ordering and word context information in the sentence. An extension to this idea is to use dependency tree relations to represent individual sentence fragments. Thus, they represent every word using 1-of-k encoding vector $\mathbf{w}$ with a dictionary of 400,000 words and map every dependency triplet $(R, \mathbf{w}_1, \mathbf{w}_2)$ into the embedding space as follows:

$$s = f\left(W_R \begin{bmatrix} W_e\mathbf{w}_1 \\ W_e\mathbf{w}_2 \end{bmatrix} + b_R\right). \tag{8.67}$$

Here, W_e is a $d \times 400,000$ matrix that encodes a 1-of-k vector into a d-dimensional word vector representation.

The image-sentence similarities should be consistent with the ground truth correspondences. That is, corresponding image-sentence pairs should have a higher score than all other image-sentence pairs. This will be enforced by the global ranking objective. Second, the fragment alignment objective explicitly learns the appearance of sentence fragments in the visual domain. The full objective is the sum of these two objectives (the global ranking objective and the fragment alignment objective) and a regularization term:

$$C(\theta) = C_F(\theta) + \beta C_G(\theta) + \alpha\|\theta\|_2^2, \tag{8.68}$$

where θ is a shorthand for parameters of the neural network ($\theta = \{W_e, W_R, b_R, W_m, b_m, \theta_c\}$) and α and β are hyperparameters.

Recall that the global ranking objective ensures that the computed image-sentence similarities are consistent with the ground truth annotations. Using a structured max-margin objective can explicitly associate these fragments across views. The image-sentence alignment score is defined to be the average thresholded score of their pairwise fragment scores:

$$S_{kl} = \frac{1}{|g_k|(|g_l| + n)} \sum_{i \in g_k} \sum_{j \in g_l} \max(0, v_i^{\mathsf{T}} s_j). \tag{8.69}$$

Here, g_k is the set of image fragments in image k and g_l is the set of sentence fragments in sentence l. The global ranking objective then becomes:

$$C_G(\theta) = \sum_k \left[\underbrace{\sum_l \max(0, S_{kl} S_{kk} + \triangle)}_{\text{rank images}} + \underbrace{\sum_l \max(0, S_{lk} S_{kk} + \triangle)}_{\text{rank sentences}} \right]. \quad (8.70)$$

Here, $\triangle$ is a hyperparameter. The objective stipulates that the score for true image-sentence pairs S_{kk} should be higher than S_{kl} or S_{lk} for any $l \neq k$ by at least a margin of $\triangle$.

Consider an (incomplete) fragment alignment objective that assumes a dense alignment between every corresponding image and sentence fragments:

$$C_0(\theta) = \sum_i \sum_j \max(0, 1 - y_{ij} v_i^\mathsf{T} s_j). \quad (8.71)$$

Here, the sum is over all pairs of image and sentence fragments in the training set. The quantity $v_i^\mathsf{T} s_j$ can be interpreted as the alignment score of visual fragment v_i and sentence fragment s_j. Inspired by the mi-SVM (Andrews et al. 2002), (8.71) is a simple and natural extension of a support vector machine to a multiple learning setting. Instead of treating the y_{ij} as constants, there is an alternative to minimize over them by wrapping (8.71) as follows (Karpathy et al. 2014):

$$\begin{aligned} & C_F(\theta) = \min_{y_{ij}} C_0(\theta) \\ s.t. \quad & \sum_{i \in p_i} \frac{y_{ij} + 1}{2} \geq 1, \quad \forall j \\ & y_{ij} = -1 \quad \forall i, j \quad s.t. \quad m_v(i) \neq m_s(j) \quad \text{and} \quad y_{ij} \in \{1, 1\}. \end{aligned} \quad (8.72)$$

Here, p_j is the set of image fragments in the positive bag for sentence fragment j. $m_v(i)$ and $m_s(j)$ return the index of the image and sentence (an element of $\{1, \ldots, N\}$) that the fragments v_i and s_j belong to.

Generally, this approach decomposes images and sentences into fragments and infers their intermodal alignment using a ranking objective. A multiview similarity metric has been proposed to internally align image fragments (visual objects) together with sentence fragments (dependency tree relations).

References

Andrew G, Arora R, Bilmes J, Livescu K (2013) Deep canonical correlation analysis. In: Proceedings of the 30th international conference on machine learning, pp 1247–1255

Andrews S, Hofmann T, Tsochantaridis I (2002) Multiple instance learning with generalized support vector machines. In: Proceedings of the 8th international association for the advancement of artificial intelligence, pp 943–944

Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Zitnick CL, Parikh D (2015) Vqa: Visual question answering. In: Proceedings of the international conference on computer vision, pp 2425–2433

AP SC, Lauly S, Larochelle H, Khapra M, Ravindran B, Raykar VC, Saha A (2014) An autoencoder approach to learning bilingual word representations. In: Advances in neural information processing systems, pp 1853–1861

Ba J, Mnih V, Kavukcuoglu K (2015) Multiple object recognition with visual attention. In: Proceedings of the 3rd international conference on learning representations

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Proceedings of the 3rd international conference on learning representations

Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166

Bruni E, Tran NK, Baroni M (2014) Multimodal distributional semantics. J Artif Intell Res 49(1):1–47

Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078

Denton EL, Chintala S, Fergus R, et al (2015) Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems, pp 1486–1494

Donahue J, Hendricks LA, Guadarrama S, Rohrbach M, Venugopalan S, Darrell T, Saenko K (2015) Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the 28th IEEE conference on computer vision and pattern recognition, pp 2625–2634

Dong J, Li X, Snoek CGM (2016) Word2visualvec: Cross-media retrieval by visual feature prediction

Ehrlich M, Shields TJ, Almaev T, Amer MR (2016) Facial attributes classification using multi-task representation learning. In: Proceedings of the 29th IEEE conference on computer vision and pattern recognition workshops, pp 47–55

Fang H, Gupta S, Iandola FN, Srivastava RK, Deng L, Dollar P, Gao J, He X, Mitchell M, Platt J, et al (2015) From captions to visual concepts and back. In: Proceedings of the 28th IEEE conference on computer vision and pattern recognition pp 1473–1482

Feng F, Wang X, Li R (2014) Cross-modal retrieval with correspondence autoencoder. In: Proceedings of the 22nd ACM international conference on multimedia, ACM, pp 7–16

Feng F, Wang X, Li R, Ahmad I (2015) Correspondence autoencoders for cross-modal retrieval. ACM Trans Multimed Comput Commun Appl (TOMM) 12(1s):26

Feng Y, Lapata M (2010) Visual information in semantic representation. In: The annual conference of the North American chapter of the association for computational linguistics, association for computational linguistics, pp 91–99

Frome A, Corrado GS, Shlens J, Bengio S, Dean J, Mikolov T, et al (2013) DeViSE: a deep visual-semantic embedding model. In: Advances in neural information processing systems, pp 2121–2129

Ge L, Gao J, Li X, Zhang A (2013) Multi-source deep learning for information trustworthiness estimation. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 766–774

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

Gregor K, Danihelka I, Graves A, Rezende DJ, Wierstra D (2015) Draw: a recurrent neural network for image generation. In: Proceedings of the 32nd international conference on machine learning, pp 1462–1471

Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Comput 16(12):2639–2664

Hinton GE (2009) Deep belief networks. Scholarpedia 4(5):5947

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Hotelling H (1936) Relations between two sets of variates. Biometrika 28(3/4):321–377
Hu H, Liu B, Wang B, Liu M, Wang X (2013) Multimodal DBN for predicting high-quality answers in CQA portals. In: Proceedings of the 51st annual meeting of the association for computational linguistics, vol 2, pp 843–847
Huang J, Kingsbury B (2013) Audio-visual deep learning for noise robust speech recognition. In: Proceedings of the 38th international conference on acoustics, speech, and signal processing, IEEE, pp 7596–7599
Jiang YG, Wu Z, Wang J, Xue X, Chang SF (2018) Exploiting feature and class relationships in video categorization with regularized deep neural networks. IEEE Trans Pattern Anal Mach Intell 40(2):352–364
Karpathy A, Fei-Fei L (2015) Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the 28th IEEE conference on computer vision and pattern recognition, pp 3128–3137
Karpathy A, Joulin A, Fei-Fei LF (2014) Deep fragment embeddings for bidirectional image sentence mapping. In: Advances in neural information processing systems, pp 1889–1897
Kim Y, Lee H, Provost EM (2013) Deep learning for robust feature generation in audiovisual emotion recognition. In: Proceedings of the 38th international conference on acoustics, speech and signal processing, IEEE, pp 3687–3691
Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: Proceedings of the 2nd international conference on learning representations
Kiros R, Salakhutdinov R, Zemel R (2014a) Multimodal neural language models. In: Proceedings of the 31st international conference on machine learning, pp 595–603
Kiros R, Salakhutdinov R, Zemel RS (2014b) Unifying visual-semantic embeddings with multimodal neural language models. arXiv:Learning
Larochelle H, Bengio Y (2008) Classification using discriminative restricted Boltzmann machines. In: Proceedings of the 25th international conference on machine learning, ACM, pp 536–543
Lazaridou A, Baroni M (2015) Combining language and vision with a multimodal skip-gram model. In: The annual conference of the north american chapter of the association for computational linguistics, pp 153–163
Lu A, Wang W, Bansal M, Gimpel K, Livescu K (2015) Deep multilingual correlation for improved word embeddings. In: Proceedings of the North American chapter of the association for computational linguistics: human language technologies, pp 250–256
Mansimov E, Parisotto E, Ba J, Salakhutdinov R (2016) Generating images from captions with attention. In: Proceedings of the 4th international conference on learning representations
Mao J, Xu W, Yang Y, Wang J, Huang Z, Yuille AL (2015) Deep captioning with multimodal recurrent neural networks (m-rnn). In: Proceedings of the 3rd international conference on learning representations
Mikolov T, Karafiát M, Burget L, Černockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: Proceedings of the 11th Annual conference of the international speech communication association
Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781
Mnih V, Heess N, Graves A, et al (2014) Recurrent models of visual attention. In: Advances in neural information processing systems, pp 2204–2212
Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: Proceedings of the 28th international conference on machine learning, pp 689–696

Nojavanasghari B, Gopinath D, Koushik J, Baltrušaitis T, Morency LP (2016) Deep multimodal fusion for persuasiveness prediction. In: Proceedings of the 18th ACM international conference on multimodal interaction, ACM, pp 284–288

Norouzi M, Mikolov T, Bengio S, Singer Y, Shlens J, Frome A, Corrado G, Dean J (2014) Zero-shot learning by convex combination of semantic embeddings. In: Proceedings of the 2nd international conference on learning representations

Ouyang W, Chu X, Wang X (2014) Multi-source deep learning for human pose estimation. In: Proceedings of the 27th IEEE conference on computer vision and pattern recognition, pp 2329–2336

Palangi H, Deng L, Shen Y, Gao J, He X, Chen J, Song X, Ward RK (2016) Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. IEEE/ACM Trans Audio Speech Lang Process 24(4):694–707

Pang L, Ngo CW (2015) Mutlimodal learning with deep boltzmann machine for emotion prediction in user generated videos. In: Proceedings of the 5th ACM on international conference on multimedia retrieval, ACM, pp 619–622

Reed S, Akata Z, Lee H, Schiele B (2016a) Learning deep representations of fine-grained visual descriptions. In: Proceedings of the 29th IEEE conference on computer vision and pattern recognition, pp 49–58

Reed SE, Akata Z, Mohan S, Tenka S, Schiele B, Lee H (2016b) Learning what and where to draw. In: Advances in neural information processing systems, pp 217–225

Reed SE, Akata Z, Yan X, Logeswaran L, Schiele B, Lee H (2016c) Generative adversarial text to image synthesis. In: Proceedings of the 33rd international conference on machine learning, pp 1060–1069

Salakhutdinov R, Larochelle H (2010) Efficient learning of deep boltzmann machines. In: Proceedings of the 13th international conference on artificial intelligence and statistics, pp 693–700

Silberer C, Lapata M (2014) Learning grounded meaning representations with autoencoders. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, vol 1, pp 721–732

Socher R, Karpathy A, Le QV, Manning CD, Ng AY (2014) Grounded compositional semantics for finding and describing images with sentences. Trans Assoc Comput Linguist 2(1):207–218

Sohn K, Shang W, Lee H (2014) Improved multimodal deep learning with variation of information. In: Advances in neural information processing systems, pp 2141–2149

Song Y, Morency LP, Davis R (2012) Multi-view latent variable discriminative models for action recognition. In: Proceedings of the 25th IEEE conference on computer vision and pattern recognition, IEEE, pp 2120–2127

Srivastava N, Salakhutdinov R (2012a) Learning representations for multimodal data with deep belief nets. In: Proceedings of the 25th IEEE conference on computer vision and pattern recognition workshops, vol 79

Srivastava N, Salakhutdinov R (2012b) Multimodal learning with deep boltzmann machines. In: Advances in neural information processing systems, pp 2222–2230

Suk HI, Lee SW, Shen D, Initiative ADN et al (2014) Hierarchical feature representation and multimodal fusion with deep learning for ad/mci diagnosis. NeuroImage 101:569–582

Sun S, Liu Q (2018) Multi-view deep gaussian processes. In: Proceedings of the international conference on neural information processing

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112

Usunier N, Buffoni D, Gallinari P (2009) Ranking with ordered weighted pairwise classification. In: Proceedings of the 26th international conference on machine learning, ACM, pp 1057–1064

Venugopalan S, Xu H, Donahue J, Rohrbach M, Mooney R, Saenko K (2014) Translating videos to natural language using deep recurrent neural networks. arXiv preprint arXiv:14124729

Venugopalan S, Rohrbach M, Donahue J, Mooney R, Darrell T, Saenko K (2015) Sequence to sequence-video to text. In: Proceedings of the international conference on computer vision, pp 4534–4542

Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, ACM, pp 1096–1103

Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11(Dec):3371–3408

Wang D, Cui P, Ou M, Zhu W (2015a) Deep multimodal hashing with orthogonal regularization. In: Proceedings of the 24th international joint conference on artificial intelligence, vol 367, pp 2291–2297

Wang W, Arora R, Livescu K, Bilmes J (2015b) On deep multi-view representation learning. In: Proceedings of the 32nd international conference on machine learning, pp 1083–1092

Weston J, Bengio S, Usunier N (2010) Large scale image annotation: learning to rank with joint word-image embeddings. Mach Learn 81(1):21–35

Weston J, Bengio S, Usunier N (2011) WSABIE: Scaling up to large vocabulary image annotation. In: Proceedings of the 20th international joint conference on artificial intelligence, vol 11, pp 2764–2770

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015a) Show, attend and tell: Neural image caption generation with visual attention. In: Proceedings of the 32nd international conference on machine learning, pp 2048–2057

Xu R, Xiong C, Chen W, Corso JJ (2015b) Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In: Proceedings of the 29th international association for the advancement of artificial intelligence, vol 5, p 6

Yan F, Mikolajczyk K (2015) Deep correlation for matching images and text. In: Proceedings of the 28th IEEE conference on computer vision and pattern recognition, pp 3441–3450

Yih Wt, He X, Meek C (2014) Semantic parsing for single-relation question answering. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, vol 2, pp 643–648

Chapter 9
View Construction

Abstract In most real applications, data are represented by a single view, which is difficult to apply in multiview learning. In this Chapter, we introduce six view construction methods to generate new views from the originate view. All of them will meet the assumptions of multiview models. Ideally, the hypotheses from two views ought to agree on the same example corresponding to the view consensus principle. Views are thought to be conditionally independent with each other. The simplest method is to partition feature set into disjoint subsets, each of which represents one view. The second method is to purify the high-dimension data to small sets of features as new views. By contrast, one can also generate a new view by adding noise into the originate data. The next three methods are based on neural networks. Reversed sequence can be regarded as another view in sequential models. New views can also be constructed using different modules such as kernel functions, neural networks, filters and other structures which can extract specific features from the original data. Finally, we introduce how to generate a new view conditioned on auxiliary information by conditional generative models.

9.1 Introduction

Multiview data arise naturally in our daily life. For example, a video is composed of the visual parts and the audial parts, each of which can be taken as a view. Another example is the web page, where the texts and images in it form two views. Recent progress in multiview learning makes it appealing to apply multiview learning methods and exploit the multiview nature of data. On the other hand, multiview data usually come from specific applications and require more resources than single-view data to store and process. In most of the cases, there are only single-view data available. Even though, one can expect to benefit from multiview learning by generating multiple views from the single-view data.

The construction of views is heuristic somewhat at a glance. Therefore, there should be some guiding principles to generate views suitable for learning. Since the motivation of view construction is to benefit from the multiview learning algorithms, it is natural to consider generating views that meet the assumptions of multiview

S. Sun et al., *Multiview Machine Learning*,
https://doi.org/10.1007/978-981-13-3029-2_9

models. On the one hand, it is useful to ensure that the views can sufficiently represent the data in correspondence with the model assumptions. Ideally, the hypotheses from two views ought to agree on the same example corresponding to the view consensus principle. On the other hand, multiview learning algorithms often assume the views to be conditionally independent with each other. This assumption can be replaced with a weaker view complementary principle in most of the cases. These two principles lay the base of many multiview learning algorithms, which means the generated views should take them into consideration to obtain a promised performance with these algorithms.

In this Chapter, we will introduce some methods for view construction. To begin with, we introduce three simple methods of view construction with concrete practice. In Sect. 9.2, we first demonstrate feature set partition. In Sect. 9.3, we state the principal component analysis as an example of purifying, which purifies the high-dimension data to small sets of features. In Sect. 9.4, we consider the method of noising where a new view is generated by adding noise into the original data. In the remainder of the Chapter, we introduce three neural networks based methods for view generating. In Sect. 9.5, we briefly review sequential models and focus on view construction by sequence reversing. In Sect. 9.6, we introduce how to generate distinctive views from the same original data using different modules. Finally, in Sect. 9.7, we introduce conditional generative models such as conditional generative adversarial networks and conditional variational autoencoder that can be used to generate the second view conditioned on the first view.

9.2 Feature Set Partition

One natural way to generate multiple views from single-view data is to partition feature set into disjoint subsets, each of which represents one view. When the features have their own practical meanings, the partition can be easily carried on.

For instance, in an experiment of human action recognition, the laboratory staff is equipped with fifty sensors around the body. The sensors provide a 50-dimensional feature set describing human poses and actions. According to the positions of the sensors equipped, they can be naturally partitioned into groups, each of which monitoring the motion of a part of the human body. These groups are taken as different views.

In general, suppose we have a single-view dataset $\mathscr{D}$ includes D examples. Every example has m attributes, i.e., $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$. Denote the index set of attributes as $\mathscr{A} = \{1, 2, .., m\}$. In order to generate V views, we partition the index set into V disjoint subsets, i.e., $\mathscr{A} = \bigcup_i \mathscr{A}_i$ and $\mathscr{A}_i \cap \mathscr{A}_j = \emptyset$, for $i \neq j$. Then the representation of $\mathbf{x}$ in the k-th view can be written as $\mathbf{x}^k = \{x_{i_k}\}$ where $\{i_k\} = \mathscr{A}_i$. For example, if we put all odd attributes into $\mathscr{A}_1$ and put all even attributes into $\mathscr{A}_2$, then it appears that $\mathscr{A}_1 = \{1, 3, 5, \ldots\}$ and $\mathscr{A}_2 = \{2, 4, 6, \ldots\}$. So the example in $\mathscr{A}_1$ is $\mathbf{x}^1 = \{x_1, x_3, x_5, \ldots\}$, while the example in $\mathscr{A}_2$ is $\mathbf{x}^2 = \{x_2, x_4, x_6, \ldots\}$.

9.2.1 Random Split

Traditionally, when there is not a principle to partition the feature set, random split is sometimes considered. Although sounds nonsense, random split may work well, and sometimes outperforms some reasonable partition scheme. For example, text sequences can be described by high-dimensional vectors consisting of word features, letter n-grams and many other features. In Brefeld et al. (2005)'s work, they constructed two views of a text sequence. The first view consists of the token itself with letter 2, 3 and 4-grams, the second view contains surface clues like initial upper case("Msp"), Roman numeral ("xii"), digit-dot-digit ("5.78"), and others as documented in Hakenberg et al. (2005). They performed discriminative sequence learning on data with the two generated views and data with views generated by random split. The experiments showed that random split outperforms splitting the features into a token view and a view of surface clues in terms of error rates (Brefeld et al. 2005). A similar good performance of random split was also showed in Bickel and Scheffer (2004)'s work in terms of entropy.

However, it is not clear whether random split works and why in general. It depends on both the model and the data domain. The random split way is kind of passive, which ignores the important task of feature selection. Thus, it is definitely not the most effective approach for view construction.

9.2.2 Genetic Algorithms

Splitting feature set in a random way cannot guarantee to obtain satisfying views. Thus, (Sun et al. 2011) turned to genetic algorithms (GAs) for help. In this case, each partition of the feature set is encoded in a binary string, where each binary bit is associated with a feature. If the ith bit is 1, the ith feature is selected, otherwise the feature is abandoned. The initial individual set is randomly generated. Suppose the size of the population is n, then after one generation, the size will be doubled as a result of growing offspring. Also, during the procedure of evolution, there is a certain probability of crossover and mutation in each iteration, among which the best n individuals will be selected as the next generation.

After a number of iterations, each individual in the final genetic population corresponds to a candidate feature subset, where the features corresponding to a bit with value 1 are added to it. In order to generate two views, we can select two feature subsets on which two classifiers agree most for a particular classification task. As the setting of multiple views changes, the number of individuals selected makes a difference. Diverse views would be helpful to improve the performance of original learning methods.

Sun et al. (2011) also applied GAs to generate views in multiview semi-supervised learning combined with multitask learning, which outperforms the random split baseline.

9.2.3 Pseudo Multiview Co-training

Pseudo multiview co-training (PMC) is a novel feature decomposition algorithm, which can automatically divide the features of a single view dataset into two mutually exclusive subsets (Chen et al. 2011). Although it is proposed to generate views for co-training, its idea is generalizable. Basically, the decomposed views must satisfy two conditions: first, both of them are sufficient within the given hypothesis class with low errors; second, they need to be class-conditionally independent. In PMC, it adds Balcan et al. (2005)'s condition of ε-expandability as a constraint during co-training.

For simplicity, we consider the linear classifier case, i.e., $f(x) = \mathbf{w}^\intercal \mathbf{x} + b$ parametrized by the weight vector $\mathbf{w}$. The objective function of PMC is given by,

$$\min_{\mathbf{w}_1, \mathbf{w}_2} \log(e^{l(\mathbf{w}_1; \mathscr{L})} + e^{l(\mathbf{w}_2; \mathscr{L})}),$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are weight vectors for two classifiers, respectively, and $l(\mathbf{w}; \mathscr{L})$ is the log loss over a labeled dataset $\mathscr{L}$. This objective function is constructed to satisfy the first condition aforementioned. To select mutually exclusive subsets of features, it assumes that at least one of the two classifiers have a zero weight in the i-th dimension, which can be formulated as

$$\mathbf{w}_1^i \mathbf{w}_2^i = 0, \quad 1 \leq i \leq d.$$

To avoid numerical instabilities, it squares both sides and sums over all the features, leading to the following constraint:

$$\sum_{i=1}^{d} (\mathbf{w}_1^i)^2 (\mathbf{w}_2^i)^2 = 0.$$

For the two classifiers to be able to teach each other, they must satisfy Balcan's condition of ε-expandability so as to ensure the predictions confidently on different subsets of the unlabeled data.

This optimization procedure is named pseudo multiview decomposition (PMD), which is the key step in PMC. This step can promise to find the optimal partition of the features.

9.3 Purifying

There are many redundant and duplicate information in real data. For a given example of n dimension, there are 2^n ways of selections of subspaces to construct views. Purifying is proposed to decrease the dimension from n to m, where m is far less

than n. By purifying the high-dimension data to small sets of features, the curse of dimensionality will be alleviated.

Purifying can not only reduce the redundant dimensions, but also ensure the independences of the resultant views. In the following, we introduce a frequently used purifying approach, which is based on principal component analysis (PCA).

PCA is a technique by projecting the data onto the principal subspace orthogonally and minimizing the average projection cost between the data points and their projections. For different projective directions, the data will be projected with diverse variance.

Suppose we have a data set $\mathscr{L} = \{\mathbf{x}_i\}_{i=1,\ldots,L}$ where every example has M dimensions. The goal of PCA is to reduce the number of dimensions to D, where $D < M$ is the desired dimension of examples. To simplify the illustration, we centralize the data set and update every example as the distance to the example mean $\bar{\mathbf{x}}$,

$$\bar{\mathbf{x}} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{x}_i$$

and

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}}.$$

Then the data covariance matrix is defined as

$$\mathbf{S} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}},$$

which helps to depict the variance of the projected data as

$$\frac{1}{L}\sum_{i=1}^{L}(\mathbf{u}^{\mathsf{T}}\mathbf{x}_i)^2 = \mathbf{u}^{\mathsf{T}}\mathbf{S}\mathbf{u}$$

where $\mathbf{u}$ is the unit projective vector such that $\mathbf{u}^{\mathsf{T}}\mathbf{u} = 1$. To maximize the variance and avoid overfitting, an objective function is set as

$$\mathbf{u}^{\mathsf{T}}\mathbf{S}\mathbf{u} + \lambda(1 - \mathbf{u}^{\mathsf{T}}\mathbf{u}).$$

By setting the derivative with respect to $\mathbf{u}$ equal to zero, the stationary point is obtained by solving the following generalized eigenvalue problem (Bishop 2006),

$$\mathbf{S}\mathbf{u} = \lambda\mathbf{u}.$$

Thus, it can be solved by sorting out all the eigenvalues and eigenvectors, among which the largest eigenvector is the best projective vector. This eigenvector is also known as the first principal component. The top D eigenvectors are selected as

the principal components, with the help of which, the M-dimensional data can be projected onto the corresponding D dimensional subspaces.

Zhang and Zheng (2009) proposed to decompose the feature space by first applying PCA and then greedily dividing the orthogonal components to minimize the energy diversity of the two feature sets. With respect to view generation, there are several ways to preserve the original dimensional information. One way to do this is to partition the eigenvector set into two parts as the projective vectors for two views. Due to the fact that the eigenvectors are linearly independent, the data in two views will satisfy the complementary principle for multiview learning.

9.4 Noising

As some data collected from real world are contaminated with white noise, the observations have the probability of deviating from the truth in the distance of a little noise. This inspires the generation of a new view by adding Gaussian white noise for instance, with which the deviation of the original view may be compensated in the observation space. In the case that one wants to utilize multiview learning algorithms but with only single-view data available, adding noise to the original data is the most convenient way of constructing another view as the noise can be easily generated.

In Michaeli et al. (2016)'s work, they generated the first view by randomly rotating images from the original dataset, and constructed the second view by randomly adding uniform pixel noise to the same identity. This generation of two views satisfies the assumption of view conditional independence given the original images, so that the most correlated subspaces could retain class information and exclude noise.

The performance of noising depends on the data. If the observation is full of strong noises already, the additive noise would make the situation more complicated and lose the useful information in the original data.

9.5 Sequence Reversing

Sequential models are designed for processing data in temporal order based on the previous context, which are widely used in natural language processing (NLP), protein structure prediction and speech processing. However, the unidirectional flow may either ignore the future data or lack backward dependencies. To overcome this problem, one can build another view through sequence reversing to process backward information.

In Schuster and Paliwal (1997)'s work, they proposed bidirectional recurrent neural nets (BRNNs), which feed each sequence to two RNNs forward and backward, respectively. In the view of multiview learning, BRNNs construct a new view by reversing the original sequence. Thus each point has complete information about all points before and after it. Both of the views share the same output layer and

interchange information during learning. Each iteration of the BRNNs algorithm consists of two passes and an afterward update. During the forward pass, it feeds the whole sequence into the BRNNs and determines all predicted outputs. In the process of backward pass, it calculates the error function derivative for the sequence used in the forward pass. Then it compares the distinction between the two passes and updates weights. With the help of the reversed sequence, BRNNs can better recover the original sequence, especially for the subsequence at the beginning end.

The long short term memory (LSTM) architecture was motivated by an analysis of error flows in traditional RNNs, where the single cell in LSTM is accompanied with three multiplicative units, i.e., the input, output, and forget gates. Unidirectional LSTM is much faster and more accurate than standard RNNs, so well as bidirectional LSTM (Bi-LSTM) compared to BRNNs. By constructing another view through sequence reversing, Bi-LSTM has two hidden layers, i.e., the forward layer and the backward layer. Both of them directly connect to the same input layers and output layers. Each iteration of Bi-LSTM also has two passes and one update. Given any cell in the hidden layer, the three gates are related to the previous moment. Take the input gate for example,

$$x_l = \sum_{j \in N} w_{lj} y_j(\tau - 1) + \sum_{c \in C} w_{lc} s_c(\tau - 1)$$
$$y_l = f(x_l)$$

where $y_j(\tau - 1)$ is the activation of time $\tau - 1$, $s_c(\tau - 1)$ is the state value of cell c at time $\tau - 1$, and f is the squashing function of the gate. In the backward pass, it updates the cell value through the next moment, time $\tau + 1$. In most cases, the forward subnet usually dominates the outputs.

In the trend of generating the second view through sequence reversing, many sequential models like RNN and LSTM have updated their topologies by adding another hidden layer. For example, (Huang et al. 2015) have applied bidirectional LSTM CRF (Bi-LSTM-CRF) for sequence tagging. The function of the second view depends on the concrete applications. In some online tasks where the model is required to output every learned message instantly after melting the current information, the backward view is helpless because the future data are invisible.

9.6 Multi-module

In many learning scenarios, the available data might not originate from diverse sources. To obtain distinctive views from homologous data, we can project the same data onto different subspaces using diverse modules. A module can be a kernel function, a neural network, a filter, and another structure which plays the role on extracting specific features from the original data. Similar to the shooting of videos with multiple cameras, the actor time has multiple pictures from different views at the same time. Audiences can figure out the actor through multiple views by local features. The picture in each view can be regarded as the result of feature mapping, reflecting

different characteristics contained in the original data. Additionally, to ensure the conditional independence between different views, it ought to set some constraints avoiding that repeatable features are selected.

To extract the features from original data, there are several kernel functions to describe the data by highlighting the corresponding features, such as the linear kernel, the polynomial kernel and the Gaussian kernel. After the transformation of these functions, multiple views can be generated by processing the same source data. Neural networks also have the same effects as kernel functions because their output layers display hidden features through complex computation. Several commonly used networks can be combined together, each of which represents one kind of characteristic. A network is equivalent to a module, which constitutes a view in multiview learning.

In the inner structure of neural networks, multiple units in the hidden layers are generated by different mapping functions, which can also be regarded as multiple views. In this section, we will introduce convolutional neural networks (CNN) as an example.

CNN is one kind of ordinary neural networks, which have network structure from the input layer to the output layer with many neurons. Each neuron has learnable weight and bias, receives the information from the direction of the input layer, performs a dot product, then passes the value to the next layer towards the output layer. The main architecture of CNN includes the convolutional layer, the pooling layer and the fully connected layer. The convolutional layer is the core building block of a CNN, which contains a number of neurons saving the convolution results of multiple filters. To promise a quick computation in the pass of layers, the size of filters is limited. Different filters have multiple designs respectively for the sake of digging out specific shapes. In visual recognition, the objects are detected if they have certain shapes considered to be recognized. Thus, each filter is equivalent to a module, which implies a specific feature of the target, respectively.

9.7 Conditional Generative Model

Traditional ways of generating new views are inactive because they are lack of redundant information contained in original data. For example, feature set partition in Sect. 9.2 only selects the subspace of total feature set, purifying extracts principle components and noising contaminates the observation. To preserve more details in source data and make full use of knowledge base, we can generate new views through conditional generative models, such as conditional generative adversarial nets (CGAN) and conditional variational autoencoders (CVAE).

9.7.1 Conditional Generative Adversarial Nets

Since generative adversarial networks (GAN) was proposed by Goodfellow et al. (2014), there have been many extensions to GAN so as to generate synthetic images

of exceptional visual fidelity. Basic GAN consists of two "adversarial" models: a generative model G that generate data given noise $\mathbf{z}$, and a discriminative model D that discriminates whether the data is true from training data or generated by G. G and D are both trained simultaneously to follow the two-player min-max game with value function $V(G; D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))],$$

where we adjust parameters for G to minimize $\log(1 - D(G(\mathbf{z})))$ and adjust parameters for D to minimize $\log D(\mathbf{x})$.

GAN can be extended to a conditional model (CGAN) if both the generator and discriminator are conditioned on some auxiliary information $\mathbf{y}$, such as class labels or data from other modalities (Mirza and Osindero 2014). Thus, the discriminator and the generator are learned as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] - \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))].$$

GAN is introduced to train a generative model. By conditioning on $\mathbf{y}$, the data generation process can be directed in CGAN.

CGAN helps to generate convincing images or labels given initial noise. In Antipov et al. (2017)'s work, they designed age-conditional generative adversarial network (Age-cGAN) based on encoding person's age into six age categories. So the conditions of Age-cGAN are six-dimensional one-hot vectors. As shown in Fig. 9.1, given an input face image $\mathbf{x}$ of age y_0, it will find an optimal latent vector $\mathbf{z}^*$ which has a close reconstructed face $\bar{\mathbf{x}} = G(\mathbf{z}^*, y_0)$. Then it will generate the resultant face image of target age y_{target}, $\mathbf{x}_{target} = G(\mathbf{z}^*, y_{target})$ by simply switching the age at the input of the generator. After the reconstruction, we similarly obtain two face images of the same person at two different periods. When the conditions change, one can obtain the images of the same person with different facial expressions, poses, and actions, which can be seen as multiple views.

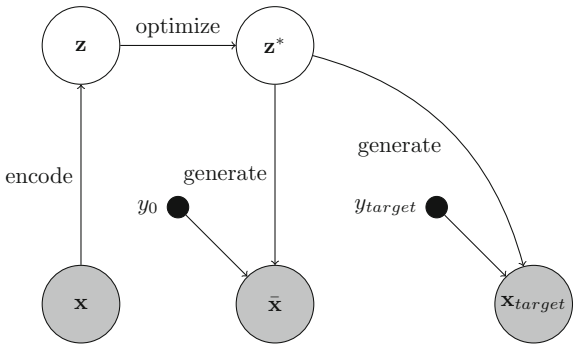


Fig. 9.1 Age-cGAN

9.7.2 Conditional Variational Autoencoders

The variational autoencoder (VAE) (Kingma and Welling 2013) is one of the most popular frameworks for image generation. The basic idea of VAE is to encode the input $\mathbf{x}$ into a probability distribution $\mathbf{z}$ instead of a point encoding in the autoencoder.

In VAE framework, there is a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ with variational parameters ϕ from $\mathbf{x}$ to $\mathbf{z}$, which is seen as an encoder. In another aspect, a generative model $p_\theta(\mathbf{x}|\mathbf{z})$ with generative parameters θ is proposed from $\mathbf{z}$ to $\mathbf{x}$, which is seen as a decoder. Given $i.i.d$ dataset $\mathscr{D} = \{\mathbf{x}_i\}_{i=1,\ldots,D}$, the goal of VAE is to maximum the log-likelihood,

$$\log p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_D) = \sum_{i=1}^{D} \log p_\theta(\mathbf{x}_i)$$

by optimizing the lower bound of log-likelihood,

$$\begin{aligned}\mathscr{L}(\theta, \phi; \mathbf{x}_i) &= -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log \frac{q_\phi(\mathbf{z}|\mathbf{x}_i)}{p_\theta(\mathbf{z}, \mathbf{x}_i)} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log p_\theta(\mathbf{z}, \mathbf{x}_i) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log q_\phi(\mathbf{z}|\mathbf{x}_i)\end{aligned}$$

where

$$\log p_\theta(\mathbf{x}_i) = KL(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p_\theta(\mathbf{z}|\mathbf{x}_i)) + \mathscr{L}(\theta, \phi; \mathbf{x}_i).$$

As the gradient of $\mathscr{L}(\theta, \phi; \mathbf{x}_i)$ with respect to ϕ exhibits very high variance, $\mathscr{L}$ is estimated through Monte Carlo method. To generate images, VAE obtains a sample of $\mathbf{z}$ from the prior distribution, e.g., $N(0; I)$, and then produces an image via the decoder network.

Similar to CGAN by conditioning on auxiliary information, conditional VAE (CVAE) is a recent modification of VAE to generate diverse images conditioned on certain attributes. Given $\mathbf{y}$ as conditioning variable, CVAE generates image $\mathbf{x}$ from $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$. Thus, the conditional log-likelihood is learned as follows:

$$\log p_\theta(\mathbf{x}|\mathbf{y}) = KL(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})) + \mathscr{L}(\theta, \phi; \mathbf{x}, \mathbf{y}),$$

where the variational lower bound

$$\mathscr{L}(\theta, \phi; \mathbf{x}, \mathbf{y}) = -KL(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} \log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$$

is maximized for learning the model parameters. In Yan et al. (2016)'s work, they generated human faces given skin color as the conditional information.

References

Antipov G, Baccouche M, Dugelay JL (2017) Face aging with conditional generative adversarial networks. In: Proceedings of IEEE international conference on image processing, IEEE, pp 2089–2093

Balcan MF, Blum A, Yang K (2005) Co-training and expansion: towards bridging theory and practice. In: Advances in neural information processing systems, pp 89–96

Bickel S, Scheffer T (2004) Multi-view clustering. In: Proceedings of the 4th IEEE international conference on data mining, pp 19–26

Bishop CM (2006) Pattern recognition and machine learning, Springer, chap 12, pp 561–565

Brefeld U, Büscher C, Scheffer T (2005) Multi-view discriminative sequential learning. In: Proceedings of the 16th European conference on machine learning, Springer, pp 60–71

Chen M, Chen Y, Weinberger KQ (2011) Automatic feature decomposition for single view co-training. In: Proceedings of the 28th international conference on machine learning, pp 953–960

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

Hakenberg J, Bickel S, Plake C, Brefeld U, Zahn H, Faulstich LC, Leser U, Scheffer T (2005) Systematic feature evaluation for gene name recognition. BioMed Cent Bioinform 6(1):1–11

Huang Z, Xu W, Yu K (2015) Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:150801991

Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:13126114

Michaeli T, Wang W, Livescu K (2016) Nonparametric canonical correlation analysis. In: Proceedings of the 33rd international conference on machine learning, pp 1967–1976

Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv preprint arXiv:14111784

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Sun S, Jin F, Tu W (2011) View construction for multi-view semi-supervised learning. In: Proceedings of the 8th international symposium on neural networks, Springer, pp 595–601

Yan X, Yang J, Sohn K, Lee H (2016) Attribute2image: Conditional image generation from visual attributes. In: Proceedings of the 14th European conference on computer vision, Springer, pp 776–791

Zhang W, Zheng Q (2009) Tsfs: a novel algorithm for single view co-training. Int Joint Conf Computat Sci Optim 1:492–496