NAIST-IS-DD1361013

Doctoral Dissertation

Latent Variable Models for Bag-of-Words Data Based on Kernel Embeddings of Distributions

Yuya Yoshikawa

September 16, 2015

Department of Information Science
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Yuya Yoshikawa

Latent Variable Models for Bag-of-Words Data Based on Kernel Embeddings of Distributions*

Yuya Yoshikawa

Abstract

In machine learning and its related fields such as natural language processing, kernel methods are studied to perform non-linear prediction. In this thesis, we consider a case where input data are represented as multi-sets of features, i.e., bag-of-words (BoW). Many papers have reported that kernel methods are superior to linear models in terms of prediction accuracy. However, kernel functions based on inner-product such as a Gaussian RBF kernel and polynomial kernel has a common problem that the kernel functions cannot reflect the correlation between related features in a kernel calculation.

To overcome this problem, we propose a general framework of kernel methods for BoW data, which consists of the following two parts: (1) defining a class of kernel functions with latent variables for BoW data, which we call it latent distribution kernel (LDK), and (2) developing models with LDK and their optimization methods.

LDK assumes that each feature has a low-dimensional latent vector, and each of the input data is represented as a multiset (or distribution) of latent vectors for the features associated with the datum. To represent the distributions nonparametrically and efficiently, we employ kernel embeddings of distributions, which can represent the moment information of the distributions e.g., the mean, covariance and higher-order moments as an element in a reproducing kernel Hilbert space. By this method, LDK can use all the information of the latent vectors for the kernel calculation between data, while overcoming the problem of the inner-product kernels.

*Doctoral Dissertation, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1361013, September 16, 2015.

Then, we propose models with LDK to solve three machine learning problems: classification, regression and cross-domain matching, and derive their optimization methods. In our experiments, we demonstrate the quantitative and qualitative effectiveness of the proposed methods compared to various linear and non-linear methods.

Keywords:

machine learning, latent variable models, kernel methods, classification, regression, cross-domaian matching

分布のカーネル埋め込みに基づく
Bag-of-Words データのための潜在変数モデル [*]

吉川 友也

内容梗概

機械学習や自然言語処理等の関連分野において，カーネル法は非線形予測を行うために研究されている．本論文では，入力データが特徴の多重集合，すなわちBag-of-Words（BoW）として表現される場合を考える．多くの論文では，予測性能の観点から，カーネル法は線形予測モデルよりも優れていることが報告されている．しかしながら，ガウシアンカーネルや多項式カーネル等の内積に基づくカーネル関数は，関連ある特徴間の相関をカーネルの計算に反映できないという共通の問題点がある．この問題点を解決するために，BoW データのためのカーネル法の新しい枠組みを提案する．この枠組みは (1) Latent Distribution Kernel（LDK）と呼ぶ BoW データのための潜在変数を含むカーネル関数のクラスの定義，(2) LDK に基づくモデルとそれらの最適化法の構築から構成される．LDK では，各特徴が低次元潜在ベクトルで表現されると仮定し，入力データはそのデータに含まれる特徴を表す潜在ベクトルの集合（分布）によって表現される．その分布をノンパラメトリックかつ効率的に表現するために，分布のカーネル埋め込みを用いる．この方法を用いることで，LDK は潜在ベクトルのすべての情報をデータ間のカーネルの計算のために利用するとともに，内積に基づくカーネル関数の問題点を解決することができる．LDK の有効性を確認するために，機械学習における基本的な問題である分類，回帰，異種データ間マッチングに対して，LDK を組み込んだモデルを提案し，それらの最適化法を導出する．実験では，各問題について，既存の線形や非線形手法と比較して，LDK を内包する提案モデルは予測性能が優れていることを示す．また，特徴の潜在ベクトルを可視化することにより，提案モデルの定量的有効性を示す．

キーワード

機械学習，潜在変数モデル，カーネル法，分類，回帰，異種データ間マッチング

[*]奈良先端科学技術大学院大学 情報科学研究科 情報科学専攻 博士論文, NAIST-IS-DD1361013, 2015 年 9月 16 日.

Acknowledgments

主指導教員の松本裕治教授には，研究指導だけでなく，研究生活や進路に関して様々な助言を頂きました．深く感謝致します．

池田和司教授には，お忙しい中審査委員を引き受けて頂き，公聴会・最終審査で研究に関する助言を頂きました．深く感謝致します．

新保仁准教授には，日頃から気さくに声を掛けて頂き，有意義な学生生活を送れました．また，審査委員を引き受けて頂き，有意義なコメントを多く頂きました．深く感謝致します．

コミュニケーション学研究室の教員である NTT コミュニケーション科学基礎研究所の山田武士教授，NTT サービスエボリューション研究所の澤田宏准教授には，会社業務でお忙しい中，研究指導を行っていただいたり，助言を頂きました．深く感謝致します．引き続き，コミュニケーション学研究室の学生を宜しくお願い致します．

NTT コミュニケーション科学基礎研究所の岩田具治さんには，博士前期課程の頃から現在まで，研究指導をしていただきました．研究の始め方，実験のやり方，論文の書き方等，研究の進め方を 1 から教えていただき，そのおかげで博士論文を書くことができたと言っても過言ではありません．有難うございました．また，今後も宜しくお願い致します．

研究生活の多くの時間は，NTT コミュニケーション科学基礎研究所の知能創発環境研究グループでお世話になりました．第一線で活躍する尊敬できる研究者の皆様と同じ環境で研究できて，とても有意義な時間を過ごすことが出来ました．有難うございました．

松本研究室の Kevin Duh 助教，進藤裕之助教，学生の皆さんには，定例の研究会で有益なコメントを多く頂きました．また，研究室で雑談したり，食事会をしたりして，楽しい時間を過ごすことが出来ました．加えて，秘書の北川祐子さん

には，研究生活が円滑に進むように，様々なことを手伝って頂きました．有難うございました．

最後に，私が好きな事をできるように支えてくれた父・和久，母・さゆりに感謝致します．

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, in various research fields such as computer vision, speech processing, natural language processing (NLP), and bioinformatics, *machine learning* is one of the most necessary techniques to automatically classify data, predict unseen behaviors, and extract latent structures from the data. In the field of machine learning, *kernel methods* are used as a framework for performing non-linear analysis; research on these methods has been in progress since the 1990s. Currently, the best known kernel method is support vector machines (SVMs) [9], which were originally used for non-linear classification, but have recently also been used for non-linear regression [11], anomaly detection [46], and ranking prediction [23]. Further, in NLP, SVMs are successfully used for text classification [22], named entity recognition [18], dependency structure analysis [28], etc.

The core idea of kernel methods is to map data into a high-dimensional (potentially infinite-dimensional) space. By treating the data in the high-dimensional space rather than in the original space, one can easily analyze the complex behaviors appearing in the data. An example that classification works well in the mapped space is shown in Figure 1.1. In this example, there are two classes of data: data denoted by red circle and data denoted by blue cross. Since the data denoted by the red circles are located inside those indicated by blue crosses, we cannot accurately classify the data by using a linear function. On the other hand, by representing the data in a high-dimensional space, we can find a linear function that can classify the data more accurately than in the original space. Here,

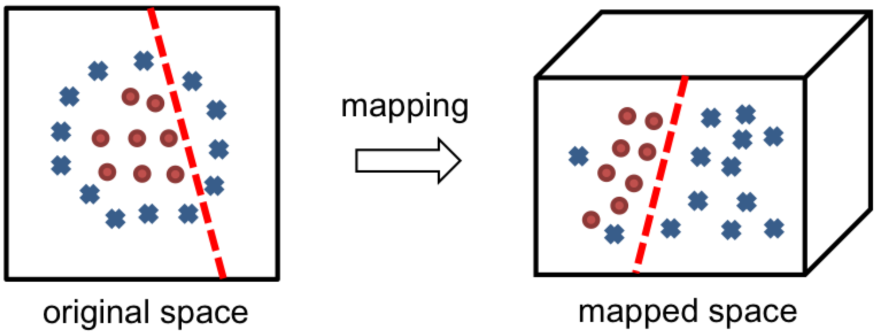


Figure 1.1: Linear discriminative functions (red dotted lines) in an original space (left) and a mapped space (right).

kernel methods do not expand the data explicitly to a high-dimensional space. Instead, they use the similarity between the data, which is calculated by using a non-linear similarity function called *kernel function*. Let $\mathbf{v}$ and $\mathbf{v}'$ be the vectors representing the input data. For example, the typical kernel functions include linear, Gaussian radial basis function (RBF) and polynomial kernels, which are defined as follows:

$$\text{Linear kernel:} \quad K_{\mathrm{LIN}}(\mathbf{v}, \mathbf{v}') = \mathbf{v}^{\top}\mathbf{v}', \tag{1.1}$$

$$\text{Gaussian RBF kernel:} \quad K_{\mathrm{RBF}}(\mathbf{v}, \mathbf{v}') = \exp\left(-\frac{\gamma}{2}||\mathbf{v} - \mathbf{v}'||^2\right), \tag{1.2}$$

$$\text{Polynomial kernel:} \quad K_{\mathrm{POLY}}(\mathbf{v}, \mathbf{v}') = \left(\mathbf{v}^{\top}\mathbf{v}' + a\right)^b, \tag{1.3}$$

where, $\gamma > 0$, $a \in \mathbb{R}$, and $b \in \mathbb{N}$. Basically, what non-linear structure kernel-based models can capture is determined by the form of the kernel function chosen. Thus, it is important to choose a kernel function appropriate for one's goal and dataset.

1.1.1 Bag-of-Words Representation and Its Problem

In this thesis, we consider a case where data are represented as *multisets* of features. Here, a multiset is a generalization of the concept of a set that allows multiple instances of the multiset's elements.

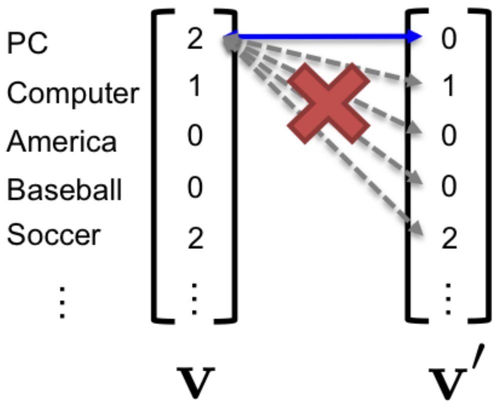


Figure 1.2: A problem occurring in kernel functions based on the inner-product.

Definition 1.1.1 (Multiset) *Let $\mathcal{U}$ be a universal set, $\mathcal{A}$ be a subset of $\mathcal{U}$, and $m : \mathcal{A} \to \mathbb{N}_+$ be a multiplicity function that counts the multiplicity of each element included in $\mathcal{A}$, where $\mathbb{N}_{\geq 1}$ denotes a positive integer set $\{1, 2, \cdots\}$, and $m(a) = 0$ if $a \notin \mathcal{A}$. Then, a multiset is defined as a 2-tuple $(\mathcal{A}, m)$.*

Such a data representation is called *the bag-of-words (BoW) representation* [14] and is a typical representation way of data in the fields of NLP, data mining, and computer vision. For example, in NLP, the universal set $\mathcal{U}$ is a vocabulary set, and each datum is represented as a multiset of vocabulary terms in $\mathcal{U}$. For notation simplicity, in this thesis, we denote the BoW data by the set notation, rather than the multiset notation.

Each of BoW data can also be represented as a feature-frequency vector whose element corresponds to the frequency of a feature associated with the datum. In such a case, previous studies have reported that kernel methods are superior to linear models in terms of the prediction accuracy. However, kernel functions based on the inner-product, such as linear, polynomial, and Gaussian RBF kernels, have a common problem that they cannot reflect the correlation between the related features in a kernel calculation. Here, the Gaussian RBF kernel Eq. (1.2) can be expanded as follows:

$$K_{\mathrm{RBF}}(\mathbf{v}, \mathbf{v}') = \exp\left(-\frac{\gamma}{2}\left[\mathbf{v}^\top \mathbf{v} + \mathbf{v}'^\top \mathbf{v}' - 2\mathbf{v}^\top \mathbf{v}\right]\right). \tag{1.4}$$

Therefore, the Gaussian RBF kernel can be regarded as a type of inner-product-based kernel. Fig. 1.2 illustrates a problem that occurs in inner-product kernels when two BoW documents are given. The inner-product between vectors $\mathbf{v}$ and $\mathbf{v}'$ is given by

$$\mathbf{v}^\top \mathbf{v}' = \sum_{l=1}^{d} v_l v'_l, \tag{1.5}$$

where d denotes the dimensionality of $\mathbf{v}$ and $\mathbf{v}'$, and v_l and v'_l indicate the lth elements of $\mathbf{v}$ and $\mathbf{v}'$, respectively. That is, the inner-product only considers the correlation of the same dimension in the two vectors. However, this is a counterintuitive result. Because although people know that the features 'PC' and 'Computer' indicate almost the same thing, the correlation between the frequency of these two features cannot be considered in the inner-product. Thus, the existing machine learning systems based on kernel methods are expected to be improved by development of a general framework to overcome this problem.

1.2 Contribution

The contribution of this thesis is to develop a general framework of kernel methods for the BoW data. This framework consists of the following two parts: (1) defining a class of kernel functions with latent variables appropriate for the BoW data, which we call *latent distribution kernel (LDK)*, and (2) developing models with LDKs and their optimization methods.

LDK is a class of kernel functions that can capture the relationship between features by incorporating latent variables for the features into the kernel functions. Each of the latent variables is represented as a vector in a lower-dimensional space than an observed BoW data space. In this thesis, we refer to the vector as *latent vector* and the space as *latent space*.

LDK assumes that each feature has a low-dimensional latent vector, and each of the input data is represented as a multiset (or distribution) of latent vectors for the features associated with the datum. To represent the distributions non-parametrically and efficiently, we employ a framework of kernel embeddings of distributions, which can represent the moment information of the distributions, e.g., the mean, covariance, and higher-order moments, as elements in a reproducing kernel Hilbert space. By this method, LDK can use all the information of the latent vectors for the kernel calculation between data, while overcoming the problem of the existing kernel functions based on the inner-product.

LDK can be used by incorporating it into existing kernel-based algorithms such as SVMs, or by developing new kernel-based algorithms. Note that although LDK can reflect the relationship between features in kernel calculations between data, the relationship varies with the problem that one wants to solve. For example, when classifying a web page into the “Dog” or “Cat” categories, the relationship between the two features (words in this case) “Dog” and “Cat” should be weak. On the other hand, when classifying it into the “Animal” or “Economic” categories, the relationship should be strong because both “Dog” and “Cat” are characteristic words in the “Animal” category. Thus, to obtain the latent vectors of features are appropriate for solving solve a given problem, we need to develop an optimization algorithm for the problem.

In this thesis, we consider three machine learning problems: classification, regression, and cross-domain matching. With classification and regression, we attempts to solve these problems by incorporating LDK into SVMs and Gaussian processes (GPs) [44], which are well-studied kernel-based algorithms. Then, we derive optimization methods for the latent vectors of features on the basis of their objective functions. With cross-domain matching, we solve this problem by using an entirely new model with LDK.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2: Preliminaries. In this chapter, we first provide fundamental knowledge about kernel methods. Then, we introduce the framework of kernel embeddings of distributions, which is a key technique to define LDK.

Chapter 3: Latent Distribution Kernel for Bag-of-Words Data. In this chapter, we present LDK, particularly its definition, measurement of the similarity and the distance between data, and the computation of the gradients of LDK, which are often used in the later chapters.

Chapter 4: Classification. This chapter presents a method to solve the classification problem by incorporating LDK into SVMs.

Chapter 5: Regression. In this chapter, we propose a GP-based non-linear regression model with LDK. Then, we show that the proposed model can also be used for two applications: predicting group behaviors and developing information diffusion models on social networks.

Chapter 6: Cross-Domain Matching. In this chapter, we consider predicting the correspondence between the data in different domains. To solve this problem, we propose an entirely new model with LDK, which matches the data in different domains on a shared latent space.

Chapter 7: Conclusion. This chapter summarizes the thesis and discusses the direction of further research.

Chapter 2

Preliminaries

This chapter provides the background knowledge that is required to understand this thesis, particularly about kernel methods and kernel embeddings of distributions. Kernel methods form the basis of the proposed methods throughout this thesis. Kernel embeddings of distributions are necessary to define the proposed kernel presented in Chapter 3.

2.1 Kernel Methods

2.1.1 Overview

First, we will briefly overview kernel methods. In machine learning and its related fields, kernel methods are used for non-linear prediction and for capturing non-linear structures appearing in the data. A naive way to perform such non-linear analyses is to map vectors representing data into those with a higher dimensionality via a *feature map function* ϕ, and then, to perform linear prediction in the high-dimensional space. In this section, we call the vectors representing data *original vectors*, and the vectors mapped into a high-dimensional space by ϕ *feature vectors*. As a feature map function, the following can be considered:

$$\phi : (v_1, v_2) \to (v_1^2, v_2^2, \sqrt{2}v_1 v_2). \tag{2.1}$$

That is, the feature map function transforms a given original vector with variables v_1 and v_2 into a three-dimensional feature vector by calculating the square root of the multiplication of the two variables and their squares. Now, we consider

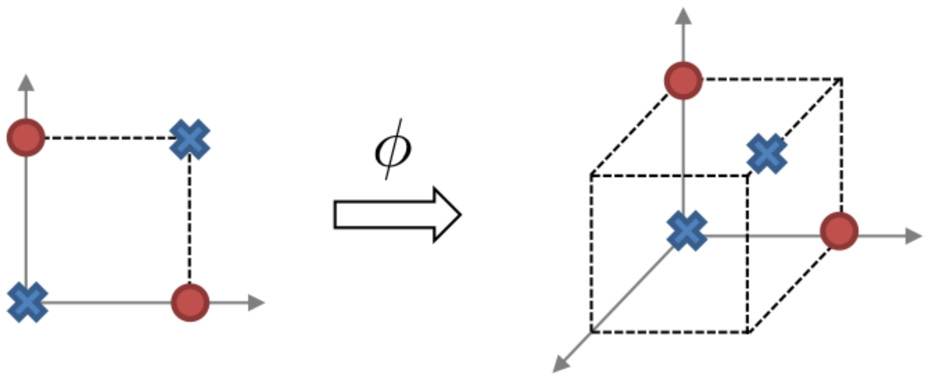


Figure 2.1: Mapping 2D original vector onto 3D space for XOR data. Red circles indicate the vectors with “True”, and blue crosses indicate the ones with “False”.

applying the feature map function to an exclusive-or (XOR) problem. In the XOR problem, our goal is to construct a classifier that upon receiving two binary inputs, outputs “True” if the two inputs have different values, and “False” otherwise, just like an XOR logic gate. Figure 2.1 shows the 2D original vectors and their corresponding feature vectors mapped by Eq. (2.1) for the XOR problem. In this problem, it is impossible to construct a linear separating hyperplane in the original space. On the other hand, by mapping the original vectors onto a 3D feature space, we can find the hyperplane.

More generally, let us consider a case where d variables $v_1, v_2, \cdots, v_d$ are given as observations. Since the number of multiplications of two variables increases with an increase in the number of variables, the dimensionality of the feature vector also increases. Thus, when the dimensionality of an original vector is large, such a naive method requires an enormous computational overhead.

Kernel methods owe their name to the use of *kernel functions*, which enable them to operate in a high-dimensional feature space without explicitly mapping the original vectors into that space, but rather by simply computing the inner-products between the original vectors. This operation is often computationally cheaper than computing a feature map function explicitly. This approach is called the *kernel trick*. Now, we show that using a kernel function is equivalent to calculating the inner-product between feature vectors without explicitly treating the feature map function ϕ. Let $\mathbf{u} = [u_1, u_2]\top$ and $\mathbf{v} = [v_1, v_2]^\top$ be the original vectors. Then, the homogeneous second-order polynomial kernel $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}^\top\mathbf{v})^2$ can be expanded as follows:

$$\begin{aligned} K(\mathbf{u},\mathbf{v}) &= (\mathbf{u}^\top\mathbf{v})^2 \\ &= (u_1v_1 + u_2v_2)^2 \\ &= u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1u_2v_2 \\ &= (u_1^2, u_2^2, \sqrt{2}u_1u_2)^\top(v_1^2, v_2^2, \sqrt{2}v_1v_2) \\ &= \phi(\mathbf{u})^\top\phi(\mathbf{v}). \end{aligned} \tag{2.2}$$

Thus, $K(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})^\top\phi(\mathbf{v})$ is proven. Then, the computational cost of the kernel function is cheaper than that of the inner-product, because the kernel needs three multiplications and an addition, while the inner-product needs nine multiplications and three additions.

2.1.2 Dual Representation

What algorithms can kernel methods be applied to? In fact, objective functions for many linear algorithms in machine learning can be expressed by the inner-product between vectors representing data via *dual representation.* As an example, we introduce *ridge regression* [15], a linear regression method with L2 regularization for weights. Suppose that we are given a set of training data $\mathcal{D} = \{(\mathbf{v}_i, y_i) \mid \mathbf{v}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^n$, where $\mathbf{v}_i$ denotes the ith sample and y_i represents its corresponding target variable. Then, a target variable is predicted by using the prediction function $f(\mathbf{v}) = \mathbf{w}^\top\phi(\mathbf{v})$ with the weight vector $\mathbf{w}$. An objective function of ridge regression to estimate the optimal weight vector $\mathbf{w}$ is given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{w}^\top\phi(\mathbf{v}_i) - y_i)^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2, \tag{2.3}$$

where, $\lambda \geq 0$ denotes a regularizer parameter, which is fixed in advance. The optimal weights can be obtained by minimizing $J(\mathbf{w})$ with respect to $\mathbf{w}$. By setting the gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ to zero, we can obtain the following equation:

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{i=1}^{n}\{\mathbf{w}^\top\phi(\mathbf{v}_i) - y_i\}\phi(\mathbf{v}_i) = \sum_{i=1}^{n} a_i\phi(\mathbf{v}_i) = \mathbf{\Phi}^\top\mathbf{a}, \tag{2.4}$$

where, $a_i = \mathbf{w}^\top \phi(\mathbf{v}_i) - y_i$, $\mathbf{a} = [a_1, a_2, \cdots, a_n]^\top$, and $\mathbf{\Phi} = [\phi(\mathbf{v}_1), \phi(\mathbf{v}_2), \cdots, \phi(\mathbf{v}_n)]^\top$. Then, by substituting $\mathbf{w} = \mathbf{\Phi}^\top \mathbf{a}$ into Eq. (2.3), we obtain the following equation:

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{a} - \mathbf{a}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{a}, \tag{2.5}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top$. Since the (i, j) element of $\mathbf{\Phi}\mathbf{\Phi}^\top$ is calculated by using the inner-product between the ith and the jth feature vectors $\phi(\mathbf{v}_i)^\top \phi(\mathbf{v}_j)$, we can replace the inner-product with kernel function $K(\mathbf{v}_i, \mathbf{v}_j)$. Thus, by defining $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ whose element is expressed as $K_{ij} = K(\mathbf{v}_i, \mathbf{v}_j)$, we can rewrite Eq. (2.5) as follows:

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^\top \mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^\top \mathbf{K}\mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{K}\mathbf{a}. \tag{2.6}$$

In Eq. (2.6), we notice that the parameters to be estimated are $\mathbf{a}$ rather than $\mathbf{w}$. By setting the gradient of Eq. (2.6) with respect to $\mathbf{a}$ to zero, we can obtain the optimal $\mathbf{a}$ by using the following equation:

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y}, \tag{2.7}$$

where $\mathbf{I}_n$ denotes the identity matrix of size n. Then, the prediction function f is given by

$$f(\mathbf{v}) = \mathbf{w}^\top \phi(\mathbf{v}) = \mathbf{a}^\top \mathbf{\Phi}\phi(\mathbf{v}) = \mathbf{k}(\mathbf{v})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y}, \tag{2.8}$$

where $\mathbf{k}(\mathbf{v}) = [K(\mathbf{v}, \mathbf{v}_1), K(\mathbf{v}, \mathbf{v}_2), \cdots, K(\mathbf{v}, \mathbf{v}_n)]^\top$ denotes a vector whose element is the kernel value between the test sample $\mathbf{v}$ and one of the training samples $\mathbf{v}_i$. By thus deriving a dual representation, one can perform non-linear prediction without explicitly using any non-linear functions.

Interestingly, the resulting prediction function Eq. (2.8) depends on the values of the kernel function. Thus, it is important to choose a kernel function appropriate to one's goal and own dataset.

2.2 Kernel Embeddings of Distributions

In this section, we introduce the framework of kernel embeddings of distributions, which is a key technique to define the proposed kernel presented in Chapter 3. The kernel embeddings of distributions are used for representing probabilistic distributions nonparametrically. For a similar purpose, kernel density estimation (KDE) [45] can be used. The main difference between the kernel embeddings of distributions and KDE is that the former is used for estimating the moment information of a distribution, while the latter is used for estimating the density of the distribution. Thus, one should choose between these methods depending on one's goal.

2.2.1 Definition

The kernel embeddings of distributions are used for embedding any probability distribution $\mathbb{P}$ on space $\mathcal{X}$ into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ specified by kernel k, and the distribution is represented as element $m^*(\mathbb{P})$ in the RKHS. More precisely, when given distribution $\mathbb{P}$, the kernel embedding of the distribution or *kernel mean* $m^*(\mathbb{P})$ is defined as follows:

$$m^*(\mathbb{P}) := \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[k(\cdot,\mathbf{x})] = \int_{\mathcal{X}} k(\cdot,\mathbf{x})d\mathbb{P} \in \mathcal{H}_k, \tag{2.9}$$

where kernel k is referred to as the *embedding kernel*. It is known that the kernel mean $m^*(\mathbb{P})$ preserves the properties of probability distribution $\mathbb{P}$, such as the mean, covariance, and higher-order moments by using *characteristic kernels* (e.g., Gaussian RBF kernel) [50].

In practice, there are often cases where the distribution $\mathbb{P}$ is unknown but a set of samples $\mathbf{X} = \{\mathbf{x}_l\}_{l=1}^n$ generated from the unknown distribution is observed. For such cases, by interpreting sample set $\mathbf{X}$ as empirical distribution $\hat{\mathbb{P}} = \frac{1}{n}\sum_{l=1}^n \delta_{\mathbf{x}_l}(\cdot)$, where $\delta_{\mathbf{x}}(\cdot)$ denotes the Dirac delta function at point $\mathbf{x} \in \mathcal{X}$, the estimator of kernel mean $m(\mathbf{X})$ can be calculated by

$$m(\mathbf{X}) = \frac{1}{n}\sum_{l=1}^{n} k(\cdot,\mathbf{x}_l), \tag{2.10}$$

which is approximated with an error rate of $||m(\mathbf{X}) - m^*(\mathbb{P})||_{\mathcal{H}_k} = O_p(n^{-\frac{1}{2}})$ [48]. Unlike that in the case of kernel density estimation, the error rate of the kernel embeddings is independent of the dimensionality of the given distribution. As with the standard kernel methods, the kernel mean is not calculated explicitly.

2.2.2 Kernel Function for Distributions

On the basis of the kernel embeddings of distributions, a kernel function that measures the similarity between distributions can be defined efficiently. This is a benefit of using the kernel embeddings of distributions. The kernel function can be used for applying the existing kernel-based algorithms to the distribution data.

Suppose that two distributions $\mathbb{P}$ and $\mathbb{Q}$ on space $\mathcal{X}$ are given. Then, the kernel function is defined as the inner-product between two kernel means $m^*(\mathbb{P})$ and $m^*(\mathbb{Q})$ in the RKHS $\mathcal{H}_k$, which is given by

$$K(m^*(\mathbb{P}), m^*(\mathbb{Q})) = \langle m^*(\mathbb{P}), m^*(\mathbb{Q})\rangle_{\mathcal{H}_k} = \mathbb{E}_{\mathbf{x}\sim\mathbb{P},\mathbf{y}\sim\mathbb{Q}}[k(\mathbf{x},\mathbf{y})]. \tag{2.11}$$

When given only the samples generated from distributions $\mathbb{P}$ and $\mathbb{Q}$, the similarity between the distributions can also be measured by using the estimators of the kernel means. Let $\mathbf{X} = \{\mathbf{x}_l\}_{l=1}^{n}$ be a set of samples generated from distribution $\mathbb{P}$, and $\mathbf{Y} = \{\mathbf{y}_{l'}\}_{l'=1}^{n'}$ be a set of samples generated from distribution $\mathbb{Q}$. The estimators of the kernel means of $\mathbf{X}$ and $\mathbf{Y}$ can be obtained by using Eq. (2.10). Then, a kernel function between two distributions $\mathbb{P}$ and $\mathbb{Q}$ is given by

$$\begin{aligned} K(\mathbf{X},\mathbf{Y}) &= \langle m(\mathbf{X}), m(\mathbf{Y})\rangle_{\mathcal{H}_k} \\ &= \left\langle \frac{1}{n}\sum_{l=1}^{n} k(\cdot,\mathbf{x}_l), \frac{1}{n'}\sum_{l'=1}^{n'} k(\cdot,\mathbf{y}_{l'}) \right\rangle_{\mathcal{H}_k} \\ &= \frac{1}{nn'}\sum_{l=1}^{n}\sum_{l'=1}^{n'} k(\mathbf{x}_l,\mathbf{y}_{l'}). \end{aligned} \tag{2.12}$$

The kernel in Eq. (2.12) can be used for classifying the distribution data [40]. This method will be reviewed in Section 4.2.2.

The kernel in Eq. (2.12) is regarded as a linear kernel between the distributions $\mathbb{P}$ and $\mathbb{Q}$ as this kernel is calculated using the inner-product. Kernels that define the similarity between $\mathbb{P}$ and $\mathbb{Q}$ are called *level-2 kernels*. Non-linear level-2 kernels can also be defined analogous to standard kernel methods, which will be described in Section 3.5.

2.2.3 Measuring Distance

By using the estimator of the kernel mean in Eq. (2.10), one can measure the distance between two distributions. Given two sets of samples $\mathbf{X} = \{\mathbf{x}_l\}_{l=1}^{n}$ and $\mathbf{Y} = \{\mathbf{y}_{l'}\}_{l'=1}^{n'}$ where $\mathbf{x}_l$ and $\mathbf{y}_{l'}$ belong to the same space but are generated from different distributions $\mathbb{P}$ and $\mathbb{Q}$, one can obtain the estimators of the kernel means by using Eq. (2.10), which are respectively denoted as $m(\mathbf{X})$ and $m(\mathbf{Y})$. Then, the distance between $m(\mathbf{X})$ and $m(\mathbf{Y})$ is given by

$$D(\mathbf{X}, \mathbf{Y}) = ||m(\mathbf{X}) - m(\mathbf{Y})||^2_{\mathcal{H}_k}. \tag{2.13}$$

Intuitively, this distance reflects the difference in the moment information of the distributions. It is equivalent to the square of the *maximum mean discrepancy (MMD)*, which is used for a statistical test of independence of two distributions [13]. The distance can be calculated by expanding Eq. (2.13) as follows:

$$||m(\mathbf{X}) - m(\mathbf{Y})||^2_{\mathcal{H}_k} = \langle m(\mathbf{X}), m(\mathbf{X})\rangle_{\mathcal{H}_k} + \langle m(\mathbf{Y}), m(\mathbf{Y})\rangle_{\mathcal{H}_k} - 2\langle m(\mathbf{X}), m(\mathbf{Y})\rangle_{\mathcal{H}_k}, \tag{2.14}$$

where, $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ denotes the inner-product in RKHS $\mathcal{H}_k$, which is given by Eq. (2.12).

Chapter 3

Latent Distribution Kernel for Bag-of-Words Data

In this chapter, we present a new class of kernel functions for measuring the similarity between bag-of-words (BoW) data, which we call the *latent distribution kernel (LDK)*.

3.1 Motivation

The kernel embeddings of distributions described in Section 2.2 implicitly assume that an unknown distribution $\mathbb{P}$ is a continuous distribution and a sample generated from $\mathbb{P}$ is represented as a dense vector. If an unknown distribution $\mathbb{P}$ is a discrete distribution, a sample generated from $\mathbb{P}$ is represented as a *one-hot vector* in which the value of a single dimension is one, while the values of the other dimensions are zero. In such a case, the use of Eq. (2.12) is inappropriate, because all the inner-product terms in the embedding kernel k become zero except for cases where two vectors are identical. Thus, even if we apply the kernel embeddings of distributions to the BoW data, we cannot overcome the problem in the inner-product kernels, which is described in Section 1.1.1. Therefore, it is inappropriate to directly apply the kernel embeddings of distributions to discrete data, including the BoW data.

A naive method to apply the kernel embeddings of distributions to the BoW data is to adopt a two-stage approach as follows: (1) learning a low-dimensional vector representation for each feature in an unsupervised manner, and (2) representing each datum as a kernel mean of the set of vectors for features associated with the datum. There are many algorithms to learn low-dimensional vector representations for features, which include matrix factorization-based methods such as NMF [33] and manifold learning-based methods such as Isomap [51]. In NLP, in particular, many algorithms to obtain vector representations such that they reflect the semantics of words, such as word2vec [39], have recently been developed. However, since the vector representations are learned independently of the problem that one wants to solve, they would not be appropriate for solving the problem. The proposed class of kernel functions, latent distribution kernel (LDK), treats the vector representations for features as latent variables, which are then optimized to solve a given problem accurately.

3.2 Definition

Let D_i be the ith observed datum, which is represented as BoW, that is, D_i is a multiset of features associated with the ith datum, and D_i consists of elements in the unique feature set $\mathcal{V}$,

LDK assumes that each feature f included in the unique feature set $\mathcal{V}$ has a latent vector $\mathbf{x}_f \in \mathbb{R}^q$, where q denotes a constant parameter for determining the dimensionality of the latent vector, which we decide in advance. Then, the observed datum D_i is represented as a multiset of latent vectors of features associated with the ith datum, which is denoted by $\mathbf{X}_i = \{\mathbf{x}_f\}_{f \in D_i}$. $\mathbf{X}_i$ can be regarded as a multiset of samples obtained from an unknown distribution. To represent the distribution efficiently and nonparametrically according to the samples $\mathbf{X}_i$, we employ the framework of the kernel embeddings of distributions, which is introduced in Section 2.2. From Eq. (2.10), the estimator of the kernel mean of the ith multiset of latent vectors $\mathbf{X}_i$, $m(\mathbf{X}_i)$, is given by

$$m(\mathbf{X}_i) = \frac{1}{|D_i|} \sum_{f \in D_i} k(\cdot, \mathbf{x}_f) = \frac{1}{|\mathbf{X}_i|} \sum_{\mathbf{x} \in \mathbf{X}_i} k(\cdot, \mathbf{x}), \tag{3.1}$$

where $|\cdot|$ denotes the number of elements in a given multiset.

3.3 Measuring Similarity and Distance

Like the similarity and distance between distributions described in Sections 2.2.2 and 2.2.3, LDK can measure the similarity and distance between BoW data.

Let D_i and D_j be the observed BoW data, and $\mathbf{X}_i$ and $\mathbf{X}_j$ be multisets of latent vectors of the features included in D_i and D_j, respectively. Then, the kernel mean estimators of $\mathbf{X}_i$ and $\mathbf{X}_j$, $m(\mathbf{X}_i)$ and $m(\mathbf{X}_j)$, respectively, can be obtained by using Eq. (3.1). According to Eq. (2.12), the similarity between two multisets of latent vectors $\mathbf{X}_i$ and $\mathbf{X}_j$ is given by

$$\begin{aligned} K(\mathbf{X}_i, \mathbf{X}_j) &= \langle m(\mathbf{X}_i), m(\mathbf{X}_j) \rangle_{\mathcal{H}_k} \\ &= \left\langle \frac{1}{|D_i|} \sum_{s \in D_i} k(\cdot, \mathbf{x}_s), \frac{1}{|D_j|} \sum_{t \in D_j} k(\cdot, \mathbf{x}_t) \right\rangle_{\mathcal{H}_k} \\ &= \frac{1}{|D_i||D_j|} \sum_{s \in D_i} \sum_{t \in D_j} k(\mathbf{x}_s, \mathbf{x}_t). \end{aligned} \tag{3.2}$$

Similarly, according to Eq. (2.13), the distance between two multisets of latent vectors $\mathbf{X}_i$ and $\mathbf{X}_j$ is given by

$$\begin{aligned} D(\mathbf{X}_i, \mathbf{X}_j) &= ||m(\mathbf{X}_i) - m(\mathbf{X}_j)||^2_{\mathcal{H}_k} \qquad (3.3) \\ &= \langle m(\mathbf{X}_i), m(\mathbf{X}_i) \rangle_{\mathcal{H}_k} + \langle m(\mathbf{X}_j), m(\mathbf{X}_j) \rangle_{\mathcal{H}_k} - 2\langle m(\mathbf{X}_i), m(\mathbf{X}_j) \rangle_{\mathcal{H}_k}. \end{aligned}$$

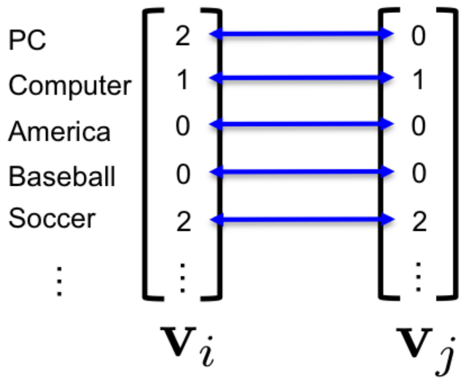
3.4 Interpretation

In this subsection, we discuss the interpretation of why LDK is superior to the existing kernel functions on the basis of the inner-product.

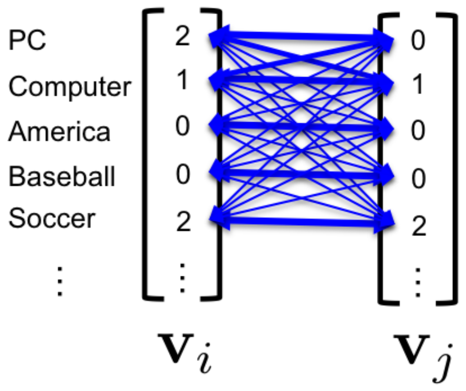
Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be the feature-frequency vectors for observed data D_i and D_j. Then, kernel functions based on the inner-product such as a linear kernel, calculate the similarity between D_i and D_j by using the following equation:

$$K(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i^\top \mathbf{v}_j = \sum_{l=1}^{d} v_{il} v_{jl}, \tag{3.4}$$

where v_{il} indicates the frequency of the lth feature (or dimension) in the ith datum, and d denotes the dimensionality of $\mathbf{v}_i$. Thus, the kernel functions only consider the correlation between the same feature in two items of data.



(a) Inner-product kernel



(b) Latent distribution kernel

Figure 3.1: The difference between the inner-product kernel and the latent distribution kernel in terms of the measurement of feature correlation.

With LDK, Eq. (3.2) can be rewritten as

$$K(\mathbf{X}_i, \mathbf{X}_j) = \frac{1}{|D_i||D_j|} \sum_{l=1}^{d} \sum_{l'=1}^{d} v_{il} v_{jl'} k(\mathbf{x}_l, \mathbf{x}_{l'}). \tag{3.5}$$

As this equation shows, the embedding kernel between latent vectors, $k(\mathbf{x}_l, \mathbf{x}_{l'})$, plays a role in controlling a weight for the correlation between the lth and the l'th features.

Figure 3.1 illustrates the difference between the inner-product kernel and LDK in terms of the weights of feature correlation. In this figure, the line weights represent the kernel values between features, which are regarded as correlation weights. With the inner-product kernel shown in Figure 3.1(a), the correlation weight for the same feature is one and that for the others is zero. On the other hand, LDK shown in Figure 3.1(b) considers the correlations between all the features, which are weighted by the embedding kernel's values between the latent vectors of the features. By learning the latent vectors so as to fit the task that we want to solve, we can automatically control the correlation weights.

3.5 Choice of Kernels

LDK includes various forms of kernel functions depending on the choice of embedding and level-2 kernels. In this section, we introduce the formulations of

LDK when using linear, Gaussian RBF, and polynomial kernels as embedding and level-2 kernels.

3.5.1 Embedding kernels

Embedding kernels determine how highly the moment information of a distribution is embedded in the RKHS. For example, a Gaussian RBF kernel can preserve all the moment information such as the mean, covariance, and higher-order moments, and the bth-order polynomial kernel can preserve all the moment information up to the bth order. We denote linear, Gaussian RBF, and polynomial embedding kernels by k_{LIN}, k_{RBF}, and k_{POLY}, respectively. These are defined as follows:

$$k_{\mathrm{LIN}}(\mathbf{x}_s, \mathbf{x}_t) = \mathbf{x}_s^\top \mathbf{x}_t, \tag{3.6}$$

$$k_{\mathrm{RBF}}(\mathbf{x}_s, \mathbf{x}_t) = \exp\left(-\frac{\gamma}{2}||\mathbf{x}_s - \mathbf{x}_t||^2\right), \tag{3.7}$$

$$k_{\mathrm{POLY}}(\mathbf{x}_s, \mathbf{x}_t) = \left(\mathbf{x}_s^\top \mathbf{x}_t + a\right)^b, \tag{3.8}$$

where $\gamma > 0$ of k_{RBF} denotes a bandwidth parameter, and $a \in \mathbb{R}$ and $b \in \mathbb{N}$ of k_{POLY} represent the bias and degree parameters, respectively.

3.5.2 Level-2 kernels

Level-2 kernels are used for defining the similarity between distributions. We indicate the choice of embedding and level-2 kernels by using the subscript K. For example, we denote LDK consisting of a Gaussian RBF embedding kernel and a linear level-2 kernel by $K_{\mathrm{RBF-LIN}}$. When a kernel formulation is independent of the choice of the embedding or level-2 kernels, we use the subscript $*$ as a wild-card operator for the choice of kernels.

Again, let $\mathbf{X}_i$ and $\mathbf{X}_j$ be the respective multisets of the latent vectors of the features included in observed data D_i and D_j. Then, we can obtain the kernel mean estimators of $\mathbf{X}_i$ and $\mathbf{X}_j$, which are denoted by $m(\mathbf{X}_i)$ and $m(\mathbf{X}_j)$, respectively, on the basis of any of the embedding kernels. A linear level-2 kernel is defined as the inner-product $\langle m(\mathbf{X}_i), m(\mathbf{X}_j)\rangle_{\mathcal{H}_k}$ in RKHS $\mathcal{H}_k$, which is given by

$$\begin{aligned} K_{*-\mathrm{LIN}}(\mathbf{X}_i, \mathbf{X}_j) &= \Big\langle \frac{1}{|D_i|}\sum_{s\in D_i} k_*(\cdot, \mathbf{x}_s), \frac{1}{|D_j|}\sum_{t\in D_j} k_*(\cdot, \mathbf{x}_t)\Big\rangle_{\mathcal{H}_k} \\ &= \frac{1}{|D_i||D_j|}\sum_{s\in D_i}\sum_{t\in D_j} k_*(\mathbf{x}_s, \mathbf{x}_t). \end{aligned} \tag{3.9}$$

The dth-order polynomial level-2 kernel with a bias parameter $c \in \mathbb{R}$ can be defined as

$$K_{*-\mathrm{POLY}}(\mathbf{X}_i, \mathbf{X}_j) = (K_{*-\mathrm{LIN}}(\mathbf{X}_i, \mathbf{X}_j) + c)^d. \tag{3.10}$$

A Gaussian RBF level-2 kernel with a bandwidth parameter $\zeta > 0$ can be defined as

$$\begin{aligned} &K_{*-\mathrm{RBF}}(\mathbf{X}_i, \mathbf{X}_j) \\ &\quad= \exp\left(-\frac{\zeta}{2}||m(\mathbf{X}_i) - m(\mathbf{X}_j)||^2\right) \\ &\quad= \exp\Big(-\frac{\zeta}{2}\big\{K_{*-\mathrm{LIN}}(\mathbf{X}_i, \mathbf{X}_i) - 2K_{*-\mathrm{LIN}}(\mathbf{X}_i, \mathbf{X}_j) + K_{*-\mathrm{LIN}}(\mathbf{X}_j, \mathbf{X}_j)\big\}\Big). \end{aligned} \tag{3.11}$$

3.6 Gradients of Kernels

To learn the latent vectors of features so as to optimize the objective function of the task that we want to solve, we will use gradient-based optimization methods such as L-BFGS [36] in the later chapters. In this section, we list the gradients of the kernels with respect to a latent vector of feature m, $\mathbf{x}_m$.

$$\frac{\partial K_{\mathrm{LIN-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = \frac{1}{|D_i||D_j|}\sum_{s\in D_i}\sum_{t\in D_j}\begin{cases} \mathbf{x}_t & (m = s \wedge m \neq t) \\ \mathbf{x}_s & (m = t \wedge m \neq s) \\ 2\mathbf{x}_m & (m = t \wedge m = s) \\ \mathbf{0} & (m \neq t \wedge m \neq s). \end{cases} \tag{3.12}$$

$$\begin{aligned} &\frac{\partial K_{\mathrm{LIN-RBF}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} \\ &\quad= -\frac{\zeta}{2}K_{\mathrm{LIN-RBF}}(\mathbf{X}_i, \mathbf{X}_j) \\ &\quad\times \left(\frac{\partial K_{\mathrm{LIN-LIN}}(\mathbf{X}_i, \mathbf{X}_i)}{\partial \mathbf{x}_m} + \frac{\partial K_{\mathrm{LIN-LIN}}(\mathbf{X}_j, \mathbf{X}_j)}{\partial \mathbf{x}_m} - 2\frac{\partial K_{\mathrm{LIN-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m}\right). \end{aligned} \tag{3.13}$$

$$\frac{\partial K_{\mathrm{LIN-POLY}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = d\left(K_{\mathrm{LIN-LIN}}(\mathbf{X}_i, \mathbf{X}_j) + c\right)^{d-1} \frac{\partial K_{\mathrm{LIN-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m}. \quad (3.14)$$

$$\frac{\partial K_{\mathrm{RBF-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = \frac{1}{|D_i||D_j|} \sum_{s \in D_i} \sum_{t \in D_j} \begin{cases} k_{\mathrm{RBF}}(\mathbf{x}_s, \mathbf{x}_t)\gamma(\mathbf{x}_t - \mathbf{x}_s) & (m = s \wedge m \neq t) \\ k_{\mathrm{RBF}}(\mathbf{x}_s, \mathbf{x}_t)\gamma(\mathbf{x}_s - \mathbf{x}_t) & (m = t \wedge m \neq s) \\ \mathbf{0} & (m = t \wedge m = s) \\ \mathbf{0} & (m \neq t \wedge m \neq s). \end{cases} \quad (3.15)$$

$$\begin{aligned} &\frac{\partial K_{\mathrm{RBF-RBF}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} \quad (3.16) \\ &= -\frac{\zeta}{2} K_{\mathrm{RBF-RBF}}(\mathbf{X}_i, \mathbf{X}_j) \\ &\times \left(\frac{\partial K_{\mathrm{RBF-LIN}}(\mathbf{X}_i, \mathbf{X}_i)}{\partial \mathbf{x}_m} + \frac{\partial K_{\mathrm{RBF-LIN}}(\mathbf{X}_j, \mathbf{X}_j)}{\partial \mathbf{x}_m} - 2\frac{\partial K_{\mathrm{RBF-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} \right). \end{aligned}$$

$$\frac{\partial K_{\mathrm{RBF-POLY}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = d\left(K_{\mathrm{RBF-LIN}}(\mathbf{X}_i, \mathbf{X}_j) + c\right)^{d-1} \frac{\partial K_{\mathrm{RBF-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m}. \quad (3.17)$$

$$\frac{\partial K_{\mathrm{POLY-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = \frac{1}{|D_i||D_j|} \sum_{s \in D_i} \sum_{t \in D_j} \begin{cases} b(\mathbf{x}_s^\top \mathbf{x}_t + a)^{b-1}\mathbf{x}_t & (m = s \wedge m \neq t) \\ b(\mathbf{x}_s^\top \mathbf{x}_t + a)^{b-1}\mathbf{x}_s & (m = t \wedge m \neq s) \\ b(\mathbf{x}_s^\top \mathbf{x}_t + a)^{b-1}2\mathbf{x}_m & (m = t \wedge m = s) \\ \mathbf{0} & (m \neq t \wedge m \neq s). \end{cases} \quad (3.18)$$

$$\begin{aligned} &\frac{\partial K_{\mathrm{POLY-RBF}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} \quad (3.19) \\ &= -\frac{\zeta}{2} K_{\mathrm{POLY-RBF}}(\mathbf{X}_i, \mathbf{X}_j) \\ &\times \left(\frac{\partial K_{\mathrm{POLY-LIN}}(\mathbf{X}_i, \mathbf{X}_i)}{\partial \mathbf{x}_m} + \frac{\partial K_{\mathrm{POLY-LIN}}(\mathbf{X}_j, \mathbf{X}_j)}{\partial \mathbf{x}_m} - 2\frac{\partial K_{\mathrm{POLY-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} \right). \end{aligned}$$

$$\frac{\partial K_{\mathrm{POLY-POLY}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m} = d\left(K_{\mathrm{POLY-LIN}}(\mathbf{X}_i, \mathbf{X}_j) + c\right)^{d-1} \frac{\partial K_{\mathrm{POLY-LIN}}(\mathbf{X}_i, \mathbf{X}_j)}{\partial \mathbf{x}_m}. \quad (3.20)$$

Chapter 4

Classification

4.1 Introduction

Classification is one of the most fundamental problems in machine learning, and the developed algorithms for classification are utilized in a wide variety of research fields such as natural language processing [29], information retrieval [57, 55], data mining [27], etc.

Classification is basically executed through the processes shown in Fig. 4.1. When training of a classifier, training data consisting of pairs of an input sample and a label are given. Here, we consider that the input sample is represented by BoW representation. Our goal is to learn a classifier (or discriminative function) f that outputs the correct label when receiving its corresponding input sample. Then, we can classify newly coming data without label information via the resulting classifier.

One of the methods to learn discriminative function f is support vector machines (SVMs) [9], which are kernel-based non-linear discriminative learning methods. Because SVMs are kernel-based methods, the performance of SVMs generally depends on whether the kernel values between input samples can be defined properly, as with kernel ridge regression described in Section 2.1.

This chapter presents SVM-based discriminative learning methods for BoW data classification, which we call them *latent support measure machine* (latent SMMs). Latent SMMs employ LDK to compute kernel values between BoW input samples. The learning procedure of latent SMMs is performed by alternately finding a separating hyperplane and estimating the latent vectors for features.

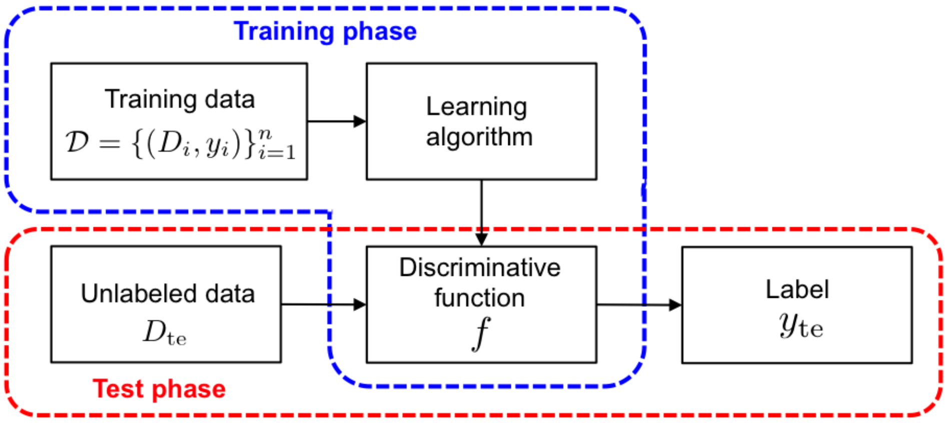


Figure 4.1: Training and test processes in classification.

The learned latent vectors of two related features are located close to each other in the latent space, and we can obtain kernel values that reflect the relationship. As a result, latent SMMs can classify unseen data using a richer and more useful representation than the BoW representation.

In our experiments, we demonstrate the quantitative and qualitative effectiveness of latent SMMs on standard BoW text datasets. The experimental results first indicate that latent SMMs can achieve state-of-the-art classification accuracy. Therefore, we show that the performance of latent SMMs is robust with respect to its own hyper-parameters, and the latent vectors for words in latent SMM can be represented in a two dimensional space while achieving high classification performance. Finally, we show that the characteristic words of each class are concentrated in a single region by visualizing the latent vectors.

Latent SMMs are a general framework of discriminative learning for BoW data. Thus, the idea of latent SMMs can be applied to various machine learning problems for BoW data, which have been solved by using SVMs: for example, novelty detection [47], structure prediction [53], and learning to rank [23].

The rest of this chapter is organized as follows: First, Section 4.2 provides the formulations of SVMs and support measure machines (SMMs), which are basis of latent SMMs. Then, Section 4.3 describes the formulation and the optimization method of latent SMMs. In Section 4.4, we introduce some related works. In Section 4.5, we demonstrate the quantitative and qualitative effectiveness of latent

SMMs. Finally, we summarize this chapter in Section 4.6.

4.2 Preliminaries

In this section, we introduce support vector machines (SVMs) and support measure machines (SMMs). Our proposed method will build upon these techniques.

4.2.1 Support Vector Machines

Support vector machines (SVMs) are discriminative learning methods, which are based on a maximum-margin criteria. Suppose that we are given a set of n training data, $\mathcal{D} = \{(\mathbf{v}_i, y_i) \mid \mathbf{v}_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\}_{i=1}^{n}$. Here, we call $\mathbf{v}_i$ *sample*, and y_i *class*. One wants to learn a discriminative function $f : \mathbb{R}^d \to \{+1, -1\}$ from the training data.

The strategy of SVM to obtain f is to find a separating hyperplane so as to maximize the margin between the samples in different classes. The separating hyperplane is defined as

$$\mathbf{w}^\top \mathbf{v} - b = 0, \tag{4.1}$$

where, $\mathbf{w} \in \mathbb{R}^d$ is a weight vector and $b \in \mathbb{R}$ is an offset parameter. Thus, discriminative function f is defined as

$$f(\mathbf{v}) = \operatorname{sgn}\left(\mathbf{w}^\top \mathbf{v} - b\right), \tag{4.2}$$

where, $\operatorname{sgn}(\cdot)$ is a sign function that returns $+1$ if its argument is positive, and -1 otherwise. Our goal is to learn the optimal $\mathbf{w}$ and b such that the margin is maximized.

In order to describe the separating hyperplane, we introduce the following formulas:

$$\mathbf{w}^\top \mathbf{v}_i - b \geq +1 \qquad (\text{if } y_i = +1) \tag{4.3}$$

$$\mathbf{w}^\top \mathbf{v}_i - b \leq -1 \qquad (\text{if } y_i = -1). \tag{4.4}$$

In these formulas, the equality is satisfied when sample $\mathbf{v}_i$ lies on the hyperplane. For $i = 1, 2, \cdots, n$, these formulas can be rewritten as:

$$y_i(\mathbf{w}^\top \mathbf{v}_i - b) \geq 1 \tag{4.5}$$

Then, the distance between the separating hyperplane Eq. (4.1) and sample $\mathbf{v}_i$ is given by

$$d(\mathbf{w}, b; \mathbf{v}_i) = \frac{|\mathbf{w}^\top \mathbf{v}_i - b|}{||\mathbf{w}||}. \tag{4.6}$$

Thus, the margin between two hyperplanes Eqs. (4.3) and (4.4) can be written as:

$$\min_{\mathbf{v}_i : y_i = +1} d(\mathbf{w}, b; \mathbf{v}_i) + \min_{\mathbf{v}_i : y_i = -1} d(\mathbf{w}, b; \mathbf{v}_i) = \frac{2}{||\mathbf{w}||}, \tag{4.7}$$

because $\min_{\mathbf{v}_i : y_i = +1} |\mathbf{w}^\top \mathbf{v}_i - b| = \min_{\mathbf{v}_i : y_i = -1} |\mathbf{w}^\top \mathbf{v}_i - b| = 1$. Since maximizing $\frac{2}{||\mathbf{w}||}$ is equivalent to minimize $\frac{1}{2}||\mathbf{w}||^2$, the optimal parameters $\mathbf{w}$ and b can be obtained by solving the following problem with constraints:

$$\min_{\mathbf{w}, b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{v}_i - b) \geq 1 \ (i = 1, 2, \cdots, n). \tag{4.8}$$

By introducing Lagrange multipliers $\mathbf{A} = \{a_1, a_2, \cdots, a_n\}$, the constrained problem Eq. (4.8) can be expressed as

$$\min_{\mathbf{w}, b} \max_{\mathbf{A} \geq \mathbf{0}} \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{n} a_i (y_i [\mathbf{w}^\top \mathbf{v}_i - b] - 1). \tag{4.9}$$

According to the gradients of Eq. (4.8) with respect to $\mathbf{w}$ and b, the following conditions are satisfied at the saddle point:

$$\mathbf{w} = \sum_{i=1}^{n} y_i a_i \mathbf{v}_i \tag{4.10a}$$

$$0 = \sum_{i=1}^{n} y_i a_i. \tag{4.10b}$$

Plugging Eq. (4.10) into Eq. (4.9), we can obtain the following dual Lagrangian problem:

$$\max_{\mathbf{A}} L(\mathbf{A}) \quad \text{where} \quad L(\mathbf{A}) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{v}_i^\top \mathbf{v}_j \tag{4.11a}$$

$$\text{subject to} \quad a_i \geq 0 \quad (i = 1, 2, \cdots, n), \quad \sum_{i=1}^{n} a_i y_i = 0. \tag{4.11b}$$

This problem can be solved as a quadratic programming problem. After the optimization, the samples with $\alpha_i > 0, i = 1, 2, \cdots, n$ are called *support vectors*. The support vector samples lie on any of the hyperplanes Eqs. (4.3) and (4.4).

The optimal offset b can be obtained by utilizing the property that the support vectors hold the equalities in Eqs. (4.3) and (4.4), that is, $\mathbf{w}^\top \mathbf{v}_i - b = 1$ if $y_i = +1$ or $\mathbf{w}^\top \mathbf{v}_i - b = -1$ if $y_i = -1$. In practice, b is obtained by the average over the support vectors as follows:

$$b = \frac{1}{n_{SV}} \sum_{i \in SV} (\mathbf{w}^\top \mathbf{v}_i - y_i), \tag{4.12}$$

where, SV is a set of support vector indexes, and n_{SV} is the number of support vectors.

Soft-margin SVMs

SVMs described above assume that all the training samples are linearly separable. Such SVMs are called *hard-margin SVMs*. If there are samples that cannot be classified correctly, the learning of hard-margin SVMs would be unstable. Soft-margin SVMs adopt a modified maximum-margin criterion that allows for misclassified samples. Soft-margin SVMs introduce non-negative slack variables, ξ_i, which measure the degree of misclassification of sample $\mathbf{v}_i$. In this case, Eq. (4.5) can be rewritten as

$$y_i(\mathbf{w}^\top \mathbf{v}_i - b) \geq 1 - \xi_i, \qquad \xi_i \geq 0. \tag{4.13}$$

Under the soft-margin criterion, the dual Lagrangian problem Eq. (4.11) changes as follows:

$$\max_{\mathbf{A}} L(\mathbf{A}) \quad \text{where} \quad L(\mathbf{A}) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{v}_i^\top \mathbf{v}_j \tag{4.14a}$$

$$\text{subject to} \quad 0 \leq a_i \leq C \quad (i = 1, 2, \cdots, n), \quad \sum_{i=1}^{n} a_i y_i = 0, \tag{4.14b}$$

where, C appearing in the constraints is a cost parameter that determines the degree of penalty for misclassification. If $C = \infty$, soft margin SVMs are equivalent to hard-margin SVMs. The value of C is usually decided by cross-validation.

Kernelization

In Eqs. (4.11) and (4.14), one can use kernel $k(\mathbf{v}_i, \mathbf{v}_j)$ instead of the inner-product $\mathbf{v}_i^\top \mathbf{v}_j$. By using a non-linear kernel such as polynomial and Gaussian kernels, non-linear classification can be executed as with the same solution for Eqs. (4.11) and (4.14). By substituting Eq. (4.10) into Eq. (4.2), the discriminative function f can be rewritten as

$$f(\mathbf{v}) = \mathrm{sgn}\left(\sum_{i \in SV} a_i y_i K(\mathbf{v}, \mathbf{v}_i) - b \right). \tag{4.15}$$

4.2.2 Support Measure Machines

Support measure machines (SMMs) are kernel-based discriminative methods for distribution data [40]. Here, each of the distribution data consists of a set of samples.

Suppose that we are given a set of n training data $\mathcal{D} = \{(\mathbf{V}_i, y_i) \mid \mathbf{V}_i = \{\mathbf{v} \in \mathbb{R}^d\}, y_i = \{+1, -1\}\}$. To represent distribution sample $\mathbf{V}_i$ efficiently and nonparametrically, SMMs employ the framework of the kernel embeddings of distributions described in Section 2.2. Since the kernel between distribution samples can be defined by using Eq. (2.12), SMMs can be solved like the standard SVM problem. For example, soft-margin SMMs can be obtained by solving the following problem:

$$\max_{\mathbf{A}} L(\mathbf{A}) \quad \text{where} \quad L(\mathbf{A}) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(\mathbf{V}_i, \mathbf{V}_j) \tag{4.16a}$$

$$\text{subject to} \quad 0 \leq a_i \leq C \quad (i = 1, 2, \cdots, n), \quad \sum_{i=1}^{n} a_i y_i = 0, \tag{4.16b}$$

where, $K(\mathbf{V}_i, \mathbf{V}_j)$ is a kernel between distribution samples $\mathbf{V}_i$ and $\mathbf{V}_j$ defined in Eq. (2.12).

4.3 Latent Support Measure Machines

In this section, we propose latent support measure machines (latent SMMs) that are effective for BoW data classification by learning latent word representation to improve classification performance. For intuitive explanation, in this section, we refer to each datum as a document, and each feature as a word, respectively.

SMMs assume that a set of samples from distribution $\mathbb{P}_i$, $\mathbf{X}_i$, is observed. On the other hand, as described later, latent SMMs assume that $\mathbf{X}_i$ is unobserved. Instead, we consider a case where BoW features are given for each document. More formally, we are given a training set of n pairs of documents and class labels $\{(D_i, y_i)\}_{i=1}^n$, where D_i is the ith document that is represented by a multiset of words appearing in the document and $y_i \in \mathcal{Y}$ is a class variable. Each word is included in vocabulary set $\mathcal{V}$. For simplicity, we consider binary class variable $y_i \in \{+1, -1\}$. The proposed method is also applicable to multi-class classification problems by adopting one-versus-one or one-versus-rest strategies as with the standard SVMs [17].

Latent SMMs adopt the data representation of LDK described in Chapter 3. That is, each word $t \in \mathcal{V}$ is represented by a q-dimensional latent vector $\mathbf{x}_t \in \mathbb{R}^q$, and the ith document is represented as a multiset of latent vectors for words appearing in the document $\mathbf{X}_i = \{\mathbf{x}_t\}_{t\in D_i}$. Then, we can obtain the kernel mean representation of the ith document from $\mathbf{X}_i$ as follows:

$$m(\mathbf{X}_i) = \frac{1}{|D_i|}\sum_{t\in D_i} k(\cdot, \mathbf{x}_t). \tag{4.17}$$

Using latent word vectors $\mathbf{X} = \{\mathbf{x}_t\}_{t\in\mathcal{V}}$ and document representations $\{m(\mathbf{X}_i)\}_{i=1}^n$, the primal optimization problem for latent SMM can be formulated in an analogous but different way from original SMMs as follows:

$$\min_{\mathbf{w},b,\xi,\mathbf{X},\theta} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^n \xi_i + \frac{\rho}{2}\sum_{t\in\mathcal{V}} ||\mathbf{x}_t||_2^2 \tag{4.18a}$$

$$\text{subject to} \quad y_i\left(\langle \mathbf{w}, m(\mathbf{X}_i)\rangle_{\mathcal{H}} - b\right) \geq 1 - \xi_i, \; \xi_i \geq 0, \tag{4.18b}$$

where $\{\xi_i\}_{i=1}^n$ denotes slack variables for handling soft margins. Unlike the primal form of SMMs, that of latent SMMs includes a ℓ_2 regularization term with parameter $\rho > 0$ with respect to latent word vectors $\mathbf{X}$. Latent SMM minimizes Eq. (4.18) with respect to the latent word vectors $\mathbf{X}$ and kernel parameters θ, along with weight parameters $\mathbf{w}$, offset parameter b and $\{\xi_i\}_{i=1}^n$.

It is extremely difficult to solve the primal problem Eq. (4.18) directly because the inner term $\langle \mathbf{w}, m(\mathbf{X}_i)\rangle_{\mathcal{H}}$ in the constrained conditions is in fact calculated in an infinite dimensional space. Thus, we solve this problem by converting it into an another optimization problem in which the inner term does not appear explicitly. Unfortunately, due to its non-convex nature, we cannot derive the dual form for Eq. (4.18) as with standard SVMs. Thus we consider a min-max optimization problem, which is derived by first introducing Lagrange multipliers $\mathbf{A} = \{a_1, a_2, \cdots, a_n\}$ and then plugging $\mathbf{w} = \sum_{i=1}^{n} a_i m(\mathbf{X}_i)$ into Eq (4.18), as follows:

$$\min_{\mathbf{X},\theta} \max_{\mathbf{A}} L(\mathbf{A}, \mathbf{X}, \theta) \quad \text{subject to} \quad 0 \leq a_i \leq C, \ \sum_{i=1}^{n} a_i y_i = 0 \tag{4.19a}$$

$$\text{where} \ \ L(\mathbf{A}, \mathbf{X}, \theta) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(m(\mathbf{X}_i), m(\mathbf{X}_j)) + \frac{\rho}{2} \sum_{t \in \mathcal{V}} ||\mathbf{x}_t||_2^2, \tag{4.19b}$$

where $K(m(\mathbf{X}_i), m(\mathbf{X}_j))$ is a kernel value between kernel mean estimators $m(\mathbf{X}_i)$ and $m(\mathbf{X}_j)$ specified by parameters $\mathbf{X}$ and θ as is shown in Section 3.5.

We solve this min-max problem by separating it into two partial optimization problems: 1) maximization over $\mathbf{A}$ given current estimates $\bar{\mathbf{X}}$ and $\bar{\theta}$, and 2) minimization over $\mathbf{X}$ and θ given current estimates $\bar{\mathbf{A}}$. This approach is analogous to *wrapper methods* in multiple kernel learning [49].

Maximization over A

When we fix $\mathbf{X}$ and θ in Eq. (4.19) with current estimate $\bar{\mathbf{X}}$ and $\bar{\theta}$, the maximization over $\mathbf{A}$ becomes a quadratic programming problem as follows:

$$\max_{\mathbf{A}} \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(m(\bar{\mathbf{X}}_i), m(\bar{\mathbf{X}}_j)) \tag{4.20}$$

$$\text{subject to} \quad 0 \leq a_i \leq C, \ \sum_{i=1}^{n} a_i y_i = 0,$$

which is identical to solving the dual problem of standard SVMs described in Section 4.2.1. Thus, we can obtain optimal $\mathbf{A}$ by employing an existing SVM package.

Minimization over $\mathbf{X}$ and θ

When we fix $\mathbf{A}$ in Eq. (4.19) with current estimate $\bar{\mathbf{A}}$, the min-max problem can be replaced with a simpler minimization problem as follows:

$$\min_{\mathbf{X},\theta} l(\mathbf{X},\theta), \quad \text{where} \quad l(\mathbf{X},\theta) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{a}_i\bar{a}_j y_i y_j K(m(\mathbf{X}_i), m(\mathbf{X}_j)) + \frac{\rho}{2}\sum_{t\in\mathcal{V}}||\mathbf{x}_t||_2^2. \tag{4.21}$$

To solve this problem, we use a quasi-Newton method [36]. The quasi-Newton method needs the gradient of parameters. For each word $m \in \mathcal{V}$, the gradient of latent word vector $\mathbf{x}_m$ is given by

$$\frac{\partial l(\mathbf{X},\theta)}{\partial \mathbf{x}_m} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{a}_i\bar{a}_j y_i y_j \frac{\partial K(m(\mathbf{X}_i), m(\mathbf{X}_j))}{\partial \mathbf{x}_m} + \rho\mathbf{x}_m, \tag{4.22}$$

where the gradient of the kernel with respect to $\mathbf{x}_m$ depends on the choice of kernels. See in Section 3.6. As with the estimation of $\mathbf{X}$, kernel parameters θ can be obtained by calculating gradient $\frac{\partial l(\mathbf{X},\theta)}{\partial\theta}$. By alternately repeating these computations until dual function Eq. (4.19) converges, we can find a local optimal solution to the min-max problem.

The parameters that need to be stored after learning are latent word vectors $\mathbf{X}$, kernel parameters θ and Lagrange multipliers $\mathbf{A}$. Classification for new document D_{te} is performed by computing

$$f(D_{\text{te}}) = \text{sgn}\left(\sum_{i=1}^{n} a_i y_i K(m(\mathbf{X}_i), m(\mathbf{X}_{\text{te}})) - b\right), \tag{4.23}$$

where, $m(\mathbf{X}_{\text{te}})$ is the kernel mean estimator of the set of the latent vectors for D_{te}.

4.4 Related Work

The proposed method is based on the framework of SMMs, which are kernel-based discriminative learning on distributions [40]. Muandet et al. showed that SMMs are more effective than SVMs when the observed feature vectors are numerical and dense in their experiments on handwriting digit recognition and natural scene categorization. On the other hand, when observations are BoW features,

Table 4.1: Dataset specifications.

	# samples	# features	# classes
WebKB	4,199	7,770	4
Reuters-21578	7,674	17,387	8
20 Newsgroups	18,821	70,216	20

SMMs coincide with SVMs as described in Section 4.2.2. To receive the benefits of SMMs for BoW data, the proposed method represents each word as a numerical and dense vector, which is estimated from the given data.

The proposed method aims to achieve a higher classification performance by learning a classifier and feature representation simultaneously. Supervised topic models [2] and maximum margin topic models (MedLDA) [60] have been proposed based on a similar motivation but using different approaches. They outperform classifiers using features extracted by unsupervised LDA. There are two main differences between these methods and the proposed method. First, the proposed method plugs the latent word vectors into a discriminant function, while the existing methods plug the document-specific vectors into their discriminant functions. Second, the proposed method can naturally develop non-linear classifiers based on the kernel embeddings of distributions. We demonstrate the effectiveness of the proposed model by comparing the topic model based classifiers in our text classification experiments.

4.5 Experiments with Bag-of-Words Text Classification

Data description

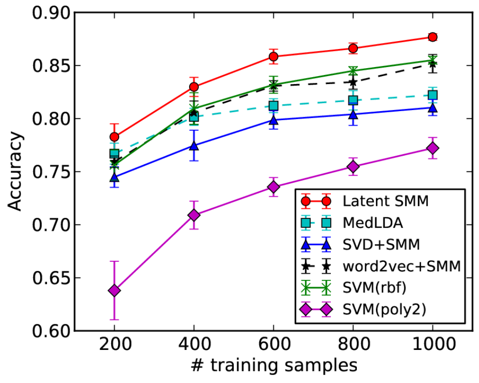
For the evaluation, we used the following three standard multi-class text classification datasets: WebKB, Reuters-21578 and 20 Newsgroups. These datasets, which have already been preprocessed by removing short and stop words, are found in [6] and can be downloaded from the author's website[1]. The specifications of these datasets are shown in Table 4.1. For our experimental setting, we ignored the original training/test data separations.

[1] http://web.ist.utl.pt/acardoso/datasets/

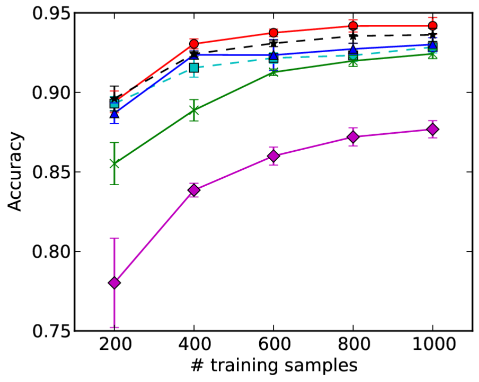
Setting

In our experiments, the proposed method, latent SMM, uses a Gaussian RBF embedding kernel and a linear level-2 kernel. To demonstrate the effectiveness of latent SMM, we compare it with several methods: MedLDA, SVD+SMM, word2vec+SMM and SVMs. MedLDA is a method that jointly learns LDA and a maximum margin classifier, which is a state-of-the-art discriminative learning method for BoW data [60]. We use the author's implementation of MedLDA[2]. SVD+SMM is a two-step procedure: 1) extracting low-dimensional representations of words by using a singular value decomposition (SVD), and 2) learning a support measure machine using the distribution of extracted representations of words appearing in each document with the same kernels as latent SMM. word2vec+SMM employs the representations of words learned by word2vec [39] and uses them for SMM as in SVD+SMM. Here we use pre-trained 300 dimensional word representation vectors from the Google News corpus, which can be downloaded from the author's website[3]. Note that word2vec+SMM utilizes an additional resource to represent the latent vectors for words unlike latent SMM, and the learning of word2vec requires n-gram information about documents, which is lost in the BoW representation. With SVMs, we use a Gaussian RBF kernel with parameter γ and a quadratic polynomial kernel, and the features are represented as BoW. We use LIBSVM[4] to estimate Lagrange multipliers $\mathbf{A}$ in latent SMM and to build SVMs and SMMs. To deal with multi-class classification, we adopt a one-versus-one strategy [17] in latent SMM, SVMs and SMMs. In our experiments, we choose the optimal parameters for these methods from the following variations: $\gamma \in \{10^{-3}, 10^{-2}, \cdots, 10^{3}\}$ in latent SMM, SVD+SMM, word2vec+SMM and SVM with a Gaussian RBF kernel, $C \in \{2^{-3}, 2^{-1}, \cdots, 2^{5}, 2^{7}\}$ in all the methods, regularizer parameter $\rho \in \{10^{-2}, 10^{-1}, 10^{0}\}$, latent dimensionality $q \in \{2, 3, 4\}$ in latent SMM, and the latent dimensionality of MedLDA and SVD+SMM ranges $\{10, 20, \cdots, 50\}$.

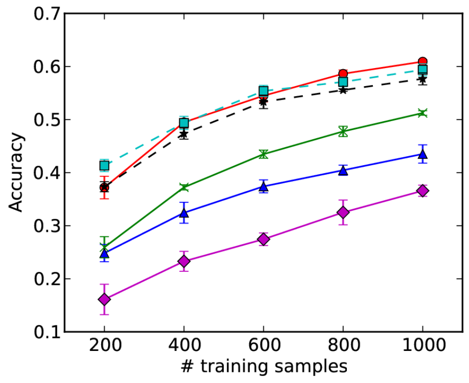
[2]http://www.ml-thu.net/~jun/medlda.shtml
[3]https://code.google.com/p/word2vec/
[4]http://www.csie.ntu.edu.tw/~cjlin/libsvm/



(a) WebKB



(b) Reuters-21578



(c) 20 Newsgroups

Figure 4.2: Classification accuracy over number of training samples.

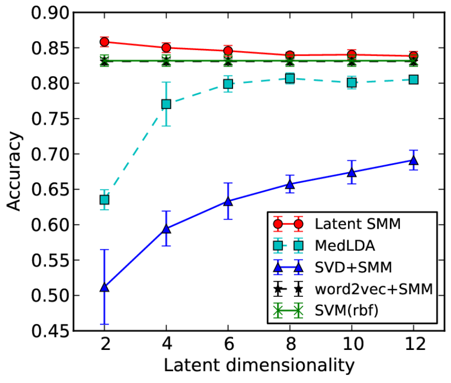
Accuracy over number of training samples

We first show the classification accuracy when varying the number of training samples. Here we randomly chose five sets of training samples, and used the remaining samples for each of the training sets as the test set. We removed words that occurred in less than 1% of the training documents. Below, we refer to the percentage as a word occurrence threshold. As shown in Figure 4.2, latent SMM outperformed the other methods for each of the numbers of training samples in the WebKB and Reuters-21578 datasets. For the 20 Newsgroups dataset, the accuracies of latent SMM, MedLDA and word2vec+SMM were proximate and better than those of SVD+SMM and SVMs.

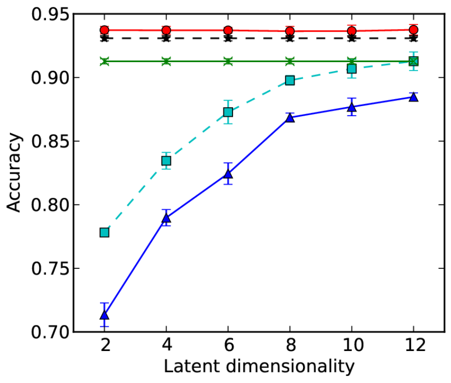
The performance of SVD+SMM changed depending on the datasets: while SVD+SMM was the second best method with the Reuters-21578, it placed fourth with the other datasets. This result indicates that the usefulness of the low rank representations by SVD for classification depends on the properties of the dataset. The high classification performance of latent SMM for all of the datasets demonstrates the effectiveness of learning the latent word representations.

Robustness over latent dimensionality

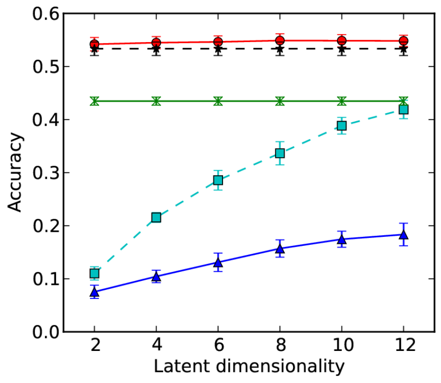
Next we confirm the robustness of the latent SMM over the latent dimensionality. For this experiment, we changed the latent dimensionality of the latent SMM, MedLDA and SVD+SMM within $\{2, 4, \cdots, 12\}$. Figure 4.3 shows the accuracy when varying the latent dimensionality. Here the number of training samples in each dataset was 600, and the word occurrence threshold was 1%. For all the latent dimensionality, the accuracy of the latent SMM was consistently better than the other methods. Moreover, even with two-dimensional latent vectors, the latent SMM achieved high classification performance. On the other hand, MedLDA and SVD+SMM often could not display their own abilities when the latent dimensionality was low. One of the reasons why latent SMM with a very low latent dimensionality q achieves a good performance is that it can use $q|d_i|$ parameters to classify the ith document, while MedLDA uses only q parameters. Since the latent word representation used in SVD+SMM is not optimized for the given classification problem, it does not contain useful features for classification, especially when the latent dimensionality is low.



(a) WebKB



(b) Reuters-21578



(c) 20 Newsgroups

Figure 4.3: Classification accuracy over the latent dimensionality.

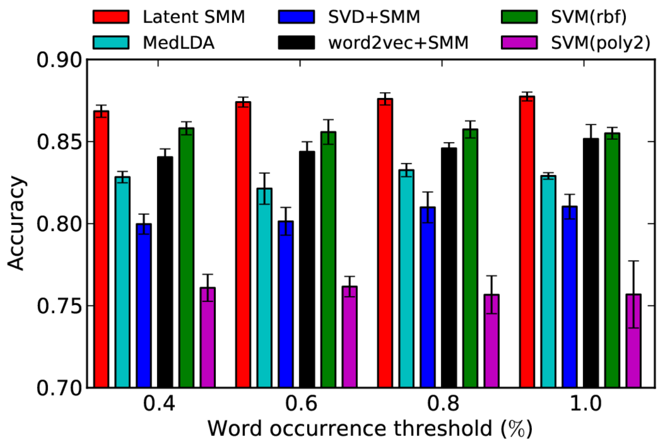


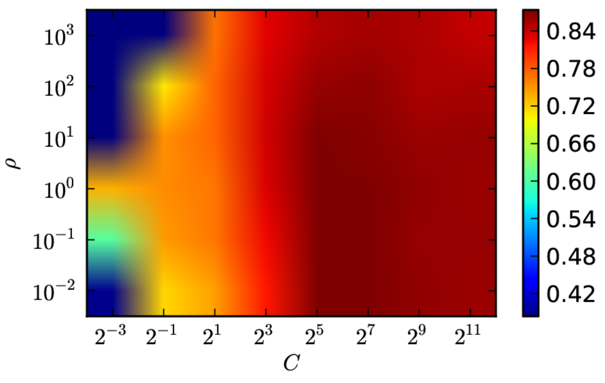
Figure 4.4: Classification accuracy on WebKB when varying word occurrence threshold.

Accuracy over word occurrence threshold

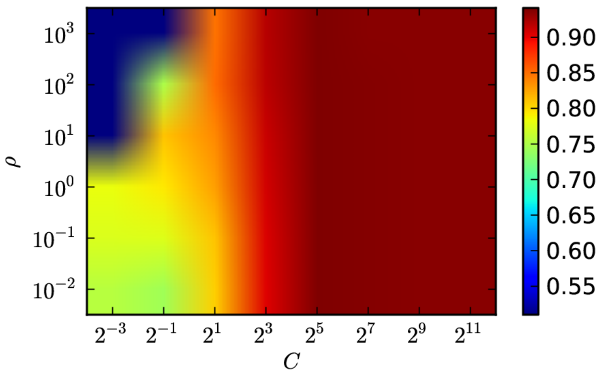
In the above experiments, we omit words whose occurrence accounts for less than 1% of the training document. By reducing the threshold, low frequency words become included in the training documents. This might be a difficult situation for latent SMM and SVD+SMM because they cannot observe enough training data to estimate their own latent word vectors. On the other hand, it would be an advantageous situation for SVMs using BoW features because they can use low frequency words that are useful for classification to compute their kernel values. Figure 4.4 shows the classification accuracy on WebKB when varying the word occurrence threshold within $\{0.4, 0.6, 0.8, 1.0\}$. The performance of latent SMM did not change when the thresholds were varied, and was better than the other methods in spite of the difficult situation.

Parameter sensitivity

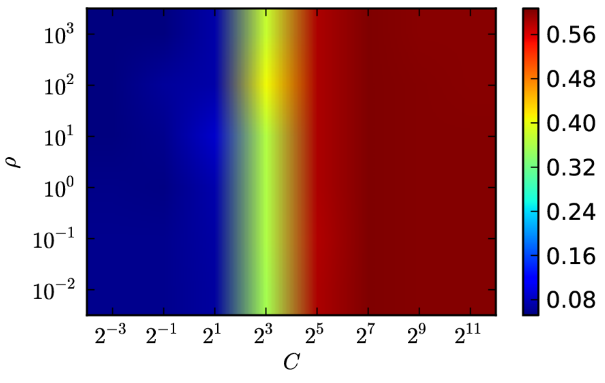
Figure 4.5 shows how the performance of latent SMM changes against ℓ_2 regularizer parameter ρ and C on a Reuters-21578 dataset with 1,000 training samples. Here the latent dimensionality of latent SMM was fixed at $q = 2$ to eliminate the



(a) WebKB



(b) Reuters-21578



(c) 20 Newsgroups

Figure 4.5: Hyper-parameter sensitivity for accuracy.

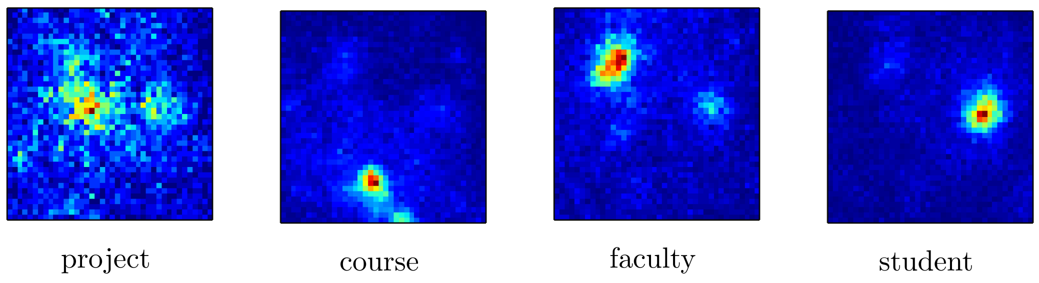


Figure 4.6: Distributions of latent vectors for words appearing in documents of each class on WebKB.

effect of q. The performance is insensitive to ρ except when C is too small. Moreover, we can see that the performance is improved by increasing the C value. In general, the performance of SVM-based methods is very sensitive to C and kernel parameters [7]. Since kernel parameters θ in latent SMM are estimated along with latent vectors $\mathbf{X}$, latent SMM can avoid the problem of sensitivity for the kernel parameters. In addition, Figure 4.3 has shown that latent SMM is robust over the latent dimensionality. Thus, latent SMM can achieve high classification accuracy by focusing only on tuning the best C, and experimentally the best C exhibits a large value, e.g., $C \geq 2^5$.

Visualization of classes

In the above experiments, we have shown that latent SMM can achieve high classification accuracy with low-dimensional latent vectors. By using two- or three-dimensional latent vectors in latent SMM, and visualizing them, we can understand the relationships between classes. Figure 4.6 shows the distributions of latent vectors for words appearing in documents of each class. Each class has its own characteristic distribution that is different from those of other classes. This result shows that latent SMM can extract the difference between the distributions of the classes. For example, the distribution of 'course' is separated from those of the other classes, which indicates that documents categorized in 'course' share few words with documents categorized in other classes. On the other hand, the latent words used in the 'project' class are widely distributed, and its distribution overlaps those of the 'faculty' and 'student' classes. This would be because faculty and students work jointly on projects, and words in both 'faculty' and 'student' appear simultaneously in ‘project’ documents.

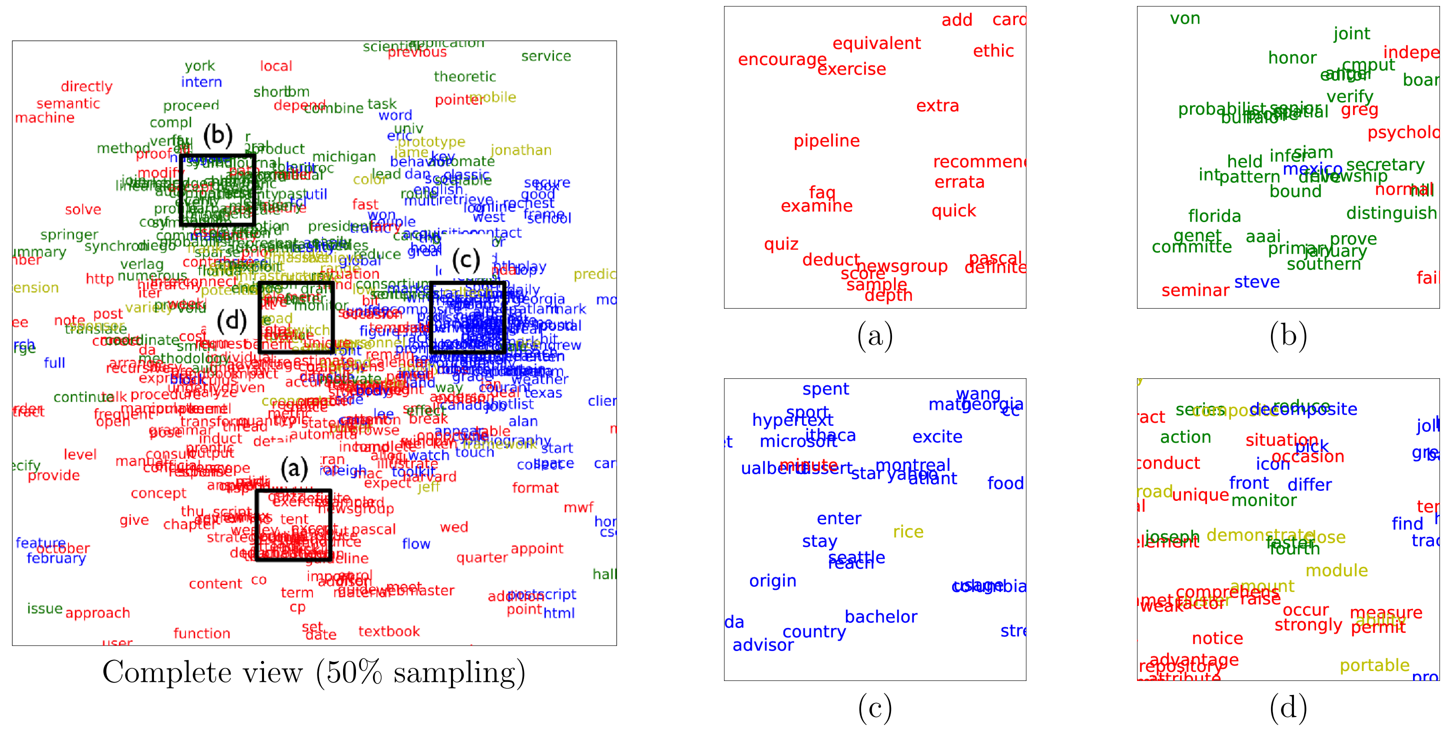


Figure 4.7: Visualization of latent vectors for words on WebKB. The font color of each word indicates the class in which the word occurs most frequently, and ‘project’, ‘course’, ‘student’ and ‘faculty’ classes correspond to yellow, red, blue and green fonts, respectively.

Visualization of words

In addition to the visualization of classes, latent SMM can visualize words using two- or three-dimensional latent vectors. Unlike unsupervised visualization methods for documents, e.g., [20], latent SMM can gather characteristic words of each class in a region. Figure 4.7 shows the visualization result of words on the WebKB dataset. Here we used the same learning result as that used in Figure 4.6. As shown in the complete view, we can see that highly-frequent words in each class tend to gather in a different region. On the right side of this figure, four regions from the complete view are displayed in closeup. Figures (a), (b) and (c) include words indicating 'course', 'faculty' and 'student' classes, respectively. For example, figure (a) includes 'exercise', 'examine' and 'quiz' which indicate examinations in lectures. Figure (d) includes words of various classes, although the 'project' class dominates the region as shown in Figure 4.6. This means that words appearing in the 'project' class are related to the other classes or are general words, e.g., 'occur' and 'differ'.

4.6 Summary

In this chapter, we have attempted to apply LDK to BoW data classification. In particular, we have proposed latent support measure machines (latent SMMs), which are kernel-based discriminative learning methods effective for BoW data. Latent SMMs adopt the data representation by LDK, that is, latent SMMs represent each feature as a latent vector, and each datum to be classified as a distribution of the latent vectors for features appearing in the datum. Then, latent SMMs find a separating hyperplane that maximizes the margins between distributions of different classes while estimating the latent vectors to improve the classification performance. The experimental results can be summarized as follows: First, latent SMMs have achieved state-of-the-art classification accuracy for BoW data. Second, we have shown experimentally that the performance of latent SMMs is robust as regards its own hyper-parameters. Third, since latent SMMs can represent each feature (word in this case) as a two- or three- dimensional latent vector, we have shown that latent SMMs are useful for understanding the relationships between classes and between words by visualizing the latent vectors.

Chapter 5

Regression

5.1 Introduction

In the previous chapter, we have present a non-linear classifier based on LDK. In machine learning, another fundamental problem is regression, which is a problem in which predicts a real value from an input datum. Figure 5.1 shows the training and test processes in regression. Regression is almost the same as classification, but because the target value is real-valued, a regression function f to be learned also outputs a real value, although the function for classification outputs a class label.

In this chapter, we consider performing prediction using a function generated by *Gaussian processes (GPs)*. This approach is called GP regression, which is a widely used method for regression problems in various domains, e.g. natural language processing [8], time series analysis [41], computer vision [26] and data mining [30].

In GP regression, the prediction function f is modeled such that two correlated inputs have correlated outputs. To describe the idea, let us consider a standard linear regression $f(\mathbf{v}) = \mathbf{w}^\top \phi(\mathbf{v})$ where $\mathbf{v}$ is an input vector and $\mathbf{w}$ is a weight vector, and output y is given by $y = f(\mathbf{v}) + \epsilon$ where ϵ is a noise term. For convenience, we define $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top$ as a vector of n outputs. Here, we put an assumption that $\mathbf{w}$ is generated from a Gaussian distribution with zero-mean and precision α, i.e., $\mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$. Note that, since $\mathbf{y}$ is computed by linear summation of the weights, $\mathbf{y}$ is also generated from a Gaussian distribution. Then, with the mean and the covariance of the output vector $\mathbf{y}$, $\mathbb{E}[\mathbf{y}]$ and $\mathrm{cov}[\mathbf{y}]$,

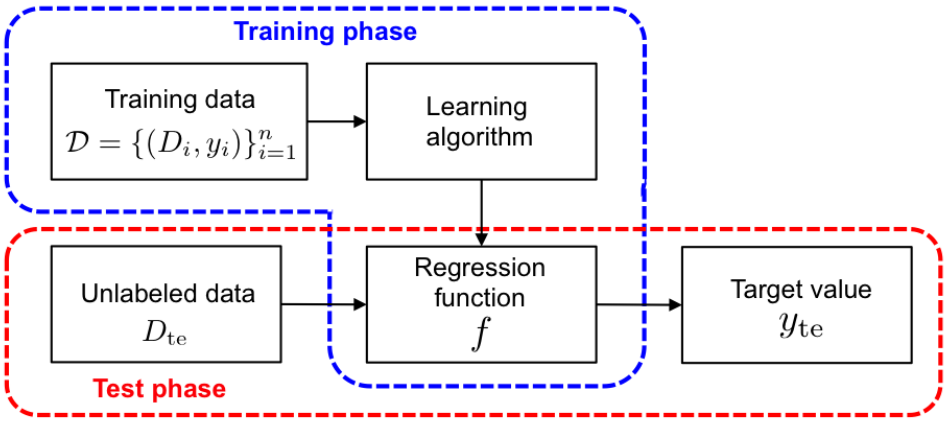


Figure 5.1: Training and test processes in regression.

the following results can be derived:

$$\mathbb{E}[\mathbf{y}] = \mathbf{0}, \tag{5.1}$$

$$\mathrm{cov}[\mathbf{y}] = \frac{1}{\alpha}\mathbf{\Phi}^\top\mathbf{\Phi}, \tag{5.2}$$

where, $\mathbf{\Phi}$ is a feature matrix whose column corresponds to an input. Thus, under this assumption, the covariance (i.e., correlation) of outputs is completely determined by the covariance of inputs. In fact, since this assumption is identical to that of GPs, the form of regression function f in GP regression is determined by the covariance of the inputs.

Since each element of covariance matrix $\mathbf{\Phi}^\top\mathbf{\Phi}$ is calculated by the inner-product between two inputs, a kernel function can be used instead of the inner-product. Thus, as with latent SMMs presented in Chapter 4, it is expected that the prediction performance of regression can be improved by applying LDK to the GP regression.

In this chapter, we propose a *Gaussian process latent variable set model (GP-LVSM)*, which is a non-linear regression model effective for BoW data. Figure 5.2 illustrates GP-LVSM. GP-LVSM adopts the representation for BoW data based on LDK. That is, GP-LVSM assumes that a latent vector is associated with each feature, and each input datum is represented as a distribution of the latent vectors for features appearing in the datum. Then, GP-LVSM generates a regression function from a Gaussian process with the covariance structures calculated by the similarity between input data based on LDK. The learning of GP-LVSM is based on maximizing a posterior (MAP) estimation, which is performed by updating the latent vectors and other kernel parameters.

In the experiments, we demonstrate the quantitative and qualitative effectiveness of GP-LVSM on 25 item review datasets. First, we show that GP-LVSM outperforms standard non-linear and linear regression methods in rating prediction. Then, we show that the performance of GP-LVSM is robust for the dimensionality of the latent vectors for features, and we can obtain vector representations for features on a quite low-dimensional space while achieving high prediction performance. Finally, we show that GP-LVSM is also useful for visualizing words.

GP-LVSM provides a general framework of solving regression problems for BoW data. Thus, the idea of GP-LVSM can be applied to various machine learning problems, which have been solved based on GP regression such as multi-task learning [4] and active learning [25].

The rest of this chapter is organized as follows. In Section 5.2, we review models and techniques related to GP-LVSM. In Section 5.3, we explain the details of GP-LVSM. In Section 5.4, we show the effectiveness and the properties of GP-LVSM experimentally. Finally, we summarize this chapter with future work in Section 5.6.

5.2 Related Work

Topic models such as latent Dirichlet allocation (LDA) [3] finds latent topic structures from BoW data. By learning the LDA, we obtain a low-dimensional and dense vector representation for each document. Supervised topic model [2] is a topic model of predicting target variables from documents, and uses the low-dimensional vectors for documents as features for prediction. We note that there are mainly two differences between GP-LVSM and the supervised topic model, which would show that GP-LVSM is better than the supervised topic model. The first one is that GP-LVSM performs non-linear prediction, while the supervised topic model is linear prediction. The second one is that GP-LVSM uses Vq parameters to represent a document, while the supervised topic model only uses K parameters, where V is the number of words in the document, q is the latent dimensionality for words and K is the number of topics. Generally, because of

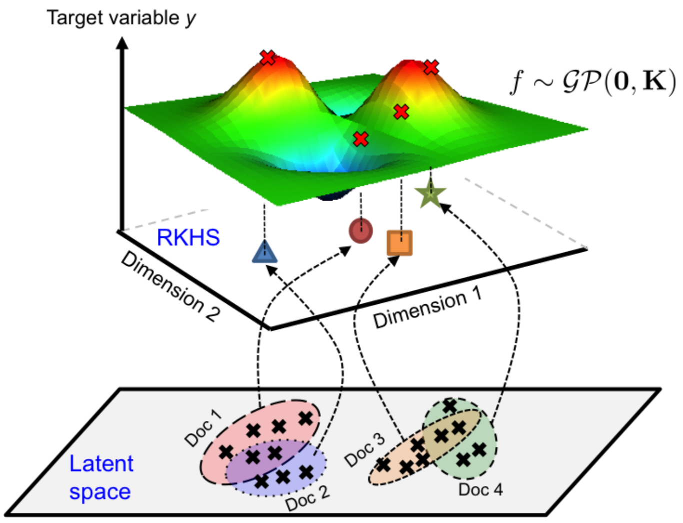


Figure 5.2: Framework of GP-LVSM. Each word is represented as a latent vector denoted by '$\times$' in the latent space. The distributions of the documents are mapped into a reproducing kernel Hilbert space (RKHS). The target variables are expressed by a non-linear regression function generated from a Gaussian process.

$Vq > K$, GP-LVSM can capture the characteristic of the document in more detail than the supervised topic model.

GP-LVSM is related to but different from the Gaussian process latent variable model (GP-LVM), which is used for dimension reduction [31] and matrix factorization [32]. Given BoW data, the GP-LVM learns a single latent vector for each datum. Since the GP-LVM cannot obtain the latent vector of a new datum, we cannot use it as a regression method. On the other hand, since GP-LVSM learns a latent vector for each feature, we can predict the target variable of a new datum by using the representation of the datum calculated from the latent vectors for features.

5.3 Gaussian Process Latent Variable Set Model

In this section, we define the proposed model, the Gaussian Process Latent Variable Set Model (GP-LVSM), in detail. Then, we explain how GP-LVSM is learned, and when a new input is given, how GP-LVSM predicts the target variable of the input. For intuitive explanation, in this section too, we refer to each datum as a document, and each feature as a word, respectively.

5.3.1 Model

Suppose that we are given a set of n training data $\mathcal{D} = \{(D_i, y_i)\}_{i=1}^{n}$, where D_i is a set of words appearing in the ith document and $y_i \in \mathbb{R}$ is its target variable. Here, D_i is bag-of-words with vocabulary set $\mathcal{V}$.

With GP-LVSM, each word $v \in \mathcal{V}$ is represented by a q-dimensional latent vector $\mathbf{x}_v \in \mathbb{R}^q$, and the ith document is represented as a multiset of latent vectors for words appearing in the document $\mathbf{X}_i = \{\mathbf{x}_v\}_{v \in D_i}$. Then, using Eq. (3.1), we can obtain the kernel mean estimator corresponding to the ith document from $\mathbf{X}_i$ by $m(\mathbf{X}_i) = \frac{1}{|D_i|}\sum_{v \in D_i} k(\cdot, \mathbf{x}_v)$.

GP-LVSM assumes the following regression model with Gaussian noise for a document and target pair (D_i, y_i):

$$f(\mathbf{X}_i) = \mathbf{w}^\top m(\mathbf{X}_i), \qquad y_i = f(\mathbf{X}_i) + \epsilon, \tag{5.3}$$

where $\mathbf{w}$ is a weight vector of the regression and ϵ is a noise drawn from a Gaussian distribution with zero mean and precision parameter β, i.e., $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

We consider the probabilistic model for Eq. (5.3). Given a set of latent vectors $\mathbf{X} = \{\mathbf{x}_v\}_{v \in \mathcal{V}}$, weight vector $\mathbf{w}$, and a set of documents $\mathbf{D} = \{D_i\}_{i=1}^{n}$, the likelihood of target variables $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top$ is given by the following Gaussian distribution:

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \mathbf{D}, \beta, \gamma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\beta^{-1}}} \exp\left(-\frac{\beta}{2}(y_i - f(\mathbf{X}_i))^2\right), \tag{5.4}$$

where γ is a parameter of embedding kernel k. We analytically marginalize out weight vector $\mathbf{w}$ by assuming the following Gaussian prior distribution with zero mean and precision parameter α:

$$p(\mathbf{w}|\alpha) = \frac{1}{\sqrt{2\pi\alpha^{-1}}} \exp\left(-\frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}\right). \tag{5.5}$$

By doing the marginalization, we do not need to explore the optimal $\mathbf{w}$ in a potentially infinite dimensional space. The marginal likelihood of target variables $\mathbf{y}$ is also a Gaussian distribution, which can be obtained analytically because likelihood Eq. (5.4) and prior Eq. (5.5) are both Gaussian distributions. As a result, the marginal likelihood is given by

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{D}, \alpha, \beta, \gamma) &= \int p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \mathbf{D}, \beta) p(\mathbf{w}|\alpha) \mathrm{d}\mathbf{w} \\ &= p(\mathbf{y}|\mathbf{0}, \alpha^{-1}\mathbf{M}\mathbf{M}^\top + \beta^{-1}\mathbf{I}), \end{aligned} \tag{5.6}$$

where $\mathbf{M} = [m(\mathbf{X}_1), m(\mathbf{X}_2), \cdots, m(\mathbf{X}_n)]^\top$. The mean and the covariance are derived by using $\mathbb{E}[\mathbf{y}] = \mathbf{M}\mathbb{E}[\mathbf{w}] = \mathbf{0}$ and $\mathbb{E}[\mathbf{y}\mathbf{y}^\top] = \mathbf{M}\mathbb{E}[\mathbf{w}\mathbf{w}^\top]\mathbf{M}^\top = \alpha^{-1}\mathbf{M}\mathbf{M}^\top$, respectively.

The (i, j) element of $\mathbf{M}\mathbf{M}^\top$ is inner product $\langle m(\mathbf{X}_i), m(\mathbf{X}_j)\rangle_{\mathcal{H}_k}$ of the kernel mean estimators for ith and jth documents on RKHS $\mathcal{H}_k$ specified by embedding kernel k. The value of the inner product is the similarity between their documents, which is calculated by one of the kernels introduced in Section 3.5. For example, when one uses linear level-2 kernel, the inner-product is given by

$$\begin{aligned} \langle m(\mathbf{X}_i), m(\mathbf{X}_j)\rangle_{\mathcal{H}_k} &= \left\langle \frac{1}{|D_i|} \sum_{s \in D_i} k(\cdot, \mathbf{x}_s), \frac{1}{|D_j|} \sum_{t \in D_j} k(\cdot, \mathbf{x}_t) \right\rangle_{\mathcal{H}_k} \\ &= \frac{1}{|D_i||D_j|} \sum_{s \in D_i} \sum_{t \in D_j} k(\mathbf{x}_s, \mathbf{x}_t). \end{aligned} \tag{5.7}$$

Using the inner product, we define kernels between documents. For each pair of document indexes (i, j), the kernel value between their documents is calculated as follows:

$$K_{ij} = \alpha^{-1}\langle m(\mathbf{X}_i), m(\mathbf{X}_j)\rangle_{\mathcal{H}_k} + \beta^{-1}\delta_{ij}, \tag{5.8}$$

where δ_{ij} is a function that returns 1 if i is equal to j and 0 otherwise. By defining $\mathbf{K}$ as a Gram matrix such that ith row and jth column is K_{ij}, marginal likelihood Eq. (5.6) can be rewritten as the following Gaussian distribution with zero mean and covariance $\mathbf{K}$.

$$p(\mathbf{y}|\mathbf{X}, \mathbf{D}, \alpha, \beta, \gamma) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det \mathbf{K}}} \exp\left(-\frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}\right). \tag{5.9}$$

5.3.2 Parameter Estimation

We estimate the parameters of GP-LVSM, latent vectors $\mathbf{X}$, precision parameters α and β, and kernel parameter γ.

For latent vectors $\mathbf{X}$, we place a Gaussian prior with zero mean and precision parameter ρ: $p(\mathbf{X}|\rho) \propto \prod_{v\in\mathcal{V}} \exp(-\frac{\rho}{2}||\mathbf{x}_v||_2^2)$. Then, the parameter estimation is performed by maximizing the following logarithm of the posterior of the parameters:

$$\begin{aligned}\mathcal{L}(\Theta) &= \log p(\mathbf{y}|\mathbf{X},\mathbf{D},\alpha,\beta,\gamma) + \log p(\mathbf{X}|\rho) \\ &\propto -\frac{1}{2}\mathbf{y}^\top\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log\det\mathbf{K} - \frac{\rho}{2}\sum_{v\in\mathcal{V}}||\mathbf{x}_v||_2^2,\end{aligned} \tag{5.10}$$

where, $\Theta = \{\mathbf{X},\alpha,\beta,\gamma\}$ is a set of parameters to be estimated.

To maximize Eq. (5.10), we use the quasi-Newton method, which is a gradient-based optimization method [36]. For each word $v \in \mathcal{V}$, the gradient with respect to $\mathbf{x}_v$ can be calculated by

$$\frac{\partial\mathcal{L}(\Theta)}{\partial\mathbf{x}_v} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial\mathcal{L}(\Theta)}{\partial\mathbf{K}}\right)_{ij}\frac{\partial K_{ij}}{\partial\mathbf{x}_v} - \rho\mathbf{x}_v. \tag{5.11}$$

The first factor $\frac{\partial\mathcal{L}(\Theta)}{\partial\mathbf{K}}$ is the gradient of $\mathcal{L}(\Theta)$ with respect to Gram matrix $\mathbf{K}$, which is given by

$$\frac{\partial\mathcal{L}(\Theta)}{\partial\mathbf{K}} = \frac{1}{2}\mathbf{K}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{K}^{-1} - \frac{1}{2}\mathbf{K}^{-1}, \tag{5.12}$$

where we note that the form of the gradient is independent of the choice of the embedding kernel. The second factor in Eq. (5.11), $\frac{\partial K_{ij}}{\partial\mathbf{x}_v}$, is the gradient of the kernel with respect to $\mathbf{x}_v$, which varies by the choice of kernels in LDK (see in Section 3.6). As with the estimation of latent vectors $\mathbf{X}$, α, β and γ can be estimated using the chain rule of Eq. (5.11).

Using these gradients, we can obtain a local solution of the parameters by continuing to update the parameters in order until the improvement of Eq. (5.10) is converged. The computational cost to calculate the gradient for each word vector $\mathbf{x} \in \mathbf{X}$ is $O(N^2W^2q)$, where W is the average number of words in documents. However, when one wants to use large training data, by using stochastic gradient descent, the computational cost can be reduced to $O(W^2q)$.

5.3.3 Prediction

When a prediction is required, we can use the standard formula for prediction by a Gaussian process regression [44]. Given a new document D_* consisting of words in $\mathcal{V}$, the predictive target variable y_* is given by

$$y_* = \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{y}, \tag{5.13}$$

where $\mathbf{k}_*$ is a vector whose element is a kernel value between the new document and a training document, that is,

$$\mathbf{k}_* = \left[K_{*1}, K_{*2}, \cdots, K_{*n}\right]^\top. \tag{5.14}$$

Intuitively, the prediction is given by a weighted sum of training target variables $\mathbf{y}$, where the weights are calculated by kernel values between training documents.

Since GP-LVSM provides the posterior distribution of the predictive target variable, we can calculate the variance of the predictive value, which is given by

$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{k}_*. \tag{5.15}$$

This variance σ_*^2 can be used for measuring the confidence of the prediction: a smaller variance indicates a higher confidence for the prediction.

5.4 Experiments

In this section, we demonstrate the effectiveness of GP-LVSM in prediction and visualization.

5.4.1 Datasets and settings

Table 5.1: Dataset specifications. n_{tr} is the number of training data, n_{te} is the number of test data and $|\mathcal{V}|$ is the maximum number of vocabularies in training data. The number of development data is equal to n_{tr}.

	n_{tr}	n_{te}	$\lvert\mathcal{V}\rvert$
apparel	1,000	7,064	1,449
automotive	200	324	918
baby	800	2,635	1,250
beauty	800	1,274	1,747
books	1,000	9,927	1,953
camera	1,000	5,338	1,434
cell phones & service	300	409	1,501
computer & video games	600	1,550	2,000
dvd	1,000	9,892	2,184
electronics	1,000	9,883	1,341
gourmet food	400	756	1,713
grocery	500	1,612	1,565
health & personal care	1,000	5,154	2,165
jewelry & watches	500	951	1,313
kitchen & housewares	1,000	9,855	1,161
magazines	700	2,745	1,695
music	1,000	9,870	1,716
musical instruments	100	127	542
office products	100	220	569
outdoor living	400	781	1,141
software	500	1,375	1,759
sports & outdoors	900	3,859	1,360
tools & hardware	30	49	155
toys & games	1,000	9,947	1,883
video	1,000	9,878	2,012

Table 5.2: Prediction errors (RMSE) and their standard deviations. Values in bold typeface are better than the others. The 'Average' row indicates the average errors and their standard deviations on all the datasets.

	GP-LVSM	GP regression	Ridge	Lasso	Elastic net
apparel	0.885 ± 0.015	0.936 ± 0.069	0.889 ± 0.018	0.898 ± 0.018	$\mathbf{0.868 \pm 0.019}$
automotive	$\mathbf{0.958 \pm 0.049}$	1.002 ± 0.075	0.972 ± 0.058	1.036 ± 0.059	0.989 ± 0.074
baby	$\mathbf{0.874 \pm 0.034}$	0.985 ± 0.039	0.933 ± 0.025	0.891 ± 0.031	0.919 ± 0.055
beauty	$\mathbf{0.890 \pm 0.030}$	0.938 ± 0.052	0.936 ± 0.019	0.914 ± 0.029	0.935 ± 0.044
books	$\mathbf{0.923 \pm 0.024}$	0.979 ± 0.023	1.111 ± 0.022	0.941 ± 0.027	0.937 ± 0.025
camera & photo	$\mathbf{0.870 \pm 0.013}$	0.962 ± 0.024	0.957 ± 0.010	0.898 ± 0.015	0.894 ± 0.019
cell phones & service	$\mathbf{0.843 \pm 0.034}$	0.942 ± 0.028	0.968 ± 0.038	0.943 ± 0.028	0.905 ± 0.026
computer & video games	$\mathbf{0.847 \pm 0.019}$	0.881 ± 0.021	0.987 ± 0.025	0.898 ± 0.019	0.932 ± 0.023
dvd	$\mathbf{0.942 \pm 0.040}$	0.968 ± 0.044	1.117 ± 0.035	0.945 ± 0.036	0.951 ± 0.038
electronics	$\mathbf{0.854 \pm 0.010}$	0.927 ± 0.045	0.983 ± 0.019	0.890 ± 0.012	0.882 ± 0.016
gourmet food	$\mathbf{0.931 \pm 0.045}$	0.947 ± 0.065	0.983 ± 0.048	0.986 ± 0.041	0.993 ± 0.052
grocery	0.927 ± 0.045	0.925 ± 0.082	0.940 ± 0.036	$\mathbf{0.922 \pm 0.049}$	0.953 ± 0.068
health & personal care	0.883 ± 0.019	$\mathbf{0.877 \pm 0.058}$	0.946 ± 0.015	0.902 ± 0.024	0.888 ± 0.022
jewelry & watches	$\mathbf{0.900 \pm 0.049}$	0.954 ± 0.082	0.924 ± 0.045	0.900 ± 0.043	0.945 ± 0.055
kitchen & housewares	$\mathbf{0.860 \pm 0.017}$	0.922 ± 0.065	0.956 ± 0.011	0.884 ± 0.008	0.873 ± 0.009
magazines	$\mathbf{0.835 \pm 0.022}$	0.882 ± 0.046	0.895 ± 0.019	0.877 ± 0.016	0.898 ± 0.028
music	0.960 ± 0.052	0.977 ± 0.065	1.128 ± 0.045	$\mathbf{0.954 \pm 0.064}$	0.956 ± 0.064
musical instruments	$\mathbf{0.966 \pm 0.144}$	1.025 ± 0.188	0.978 ± 0.128	1.031 ± 0.185	1.023 ± 0.189
office products	1.033 ± 0.108	1.041 ± 0.105	$\mathbf{1.025 \pm 0.114}$	1.108 ± 0.088	1.076 ± 0.113
outdoor living	$\mathbf{0.882 \pm 0.036}$	0.920 ± 0.074	0.952 ± 0.037	0.960 ± 0.050	0.955 ± 0.033
software	$\mathbf{0.806 \pm 0.014}$	0.915 ± 0.064	0.927 ± 0.015	0.883 ± 0.024	0.870 ± 0.025
sports & outdoors	$\mathbf{0.875 \pm 0.015}$	0.949 ± 0.048	0.950 ± 0.013	0.887 ± 0.015	0.896 ± 0.022
tools & hardware	0.892 ± 0.165	$\mathbf{0.884 \pm 0.260}$	0.918 ± 0.273	1.107 ± 0.256	0.974 ± 0.272
toys & games	$\mathbf{0.846 \pm 0.030}$	0.879 ± 0.050	0.908 ± 0.021	0.865 ± 0.019	0.851 ± 0.022
video	$\mathbf{0.844 \pm 0.027}$	0.867 ± 0.027	0.975 ± 0.019	0.891 ± 0.033	0.887 ± 0.029
Average	$\mathbf{0.893 \pm 0.052}$	0.939 ± 0.047	0.970 ± 0.064	0.936 ± 0.068	0.930 ± 0.054

For evaluation, we use 25 item review datasets obtained from Amazon.com, where each dataset corresponds to an item category on Amazon.com. Each review is represented with bag-of-words without short, low-frequency and stop words, and is associated with a rating ranging from $\{1, 2, \cdots, 5\}$. In our experiments, we use the bag-of-words as input document D and the standardized value of the rating as target variable y. Table 5.1 shows the specification of the datasets. For each dataset, we randomly choose five sets of training, development and test data from the whole of the dataset.

For comparison, we use four non-linear and linear regression methods: Gaussian Process (GP) regression, Ridge [15], Lasso [52] and Elastic net [61]. With the GP regression, we use a Gaussian RBF kernel with additive noise term as follows:

$$K_{ij} = \alpha^{-1} \exp\left(-\frac{\gamma}{2}||\text{vec}(D_i) - \text{vec}(D_j)||_2^2\right) + \beta^{-1}\delta_{ij}, \tag{5.16}$$

where $\text{vec}(\cdot)$ is a function that returns a vector with vocabulary length, and vth element of the vector is the frequency of the vth word in the given set. Parameters α, β and γ are estimated so as to maximize the marginal likelihood of the GP regression. Ridge [15], Lasso [52] and Elastic net [61] are standard linear regression models with different regularizers. We choose the parameters for these regularizers so as to minimize the prediction errors on development data. With GP-LVSM, we learned the model with latent dimensionality $q \in \{1, 2, 4, 6, 8, 10\}$ and regularizer parameter $\rho \in \{10^{-2}, 10^{-1}, \cdots, 10^{2}\}$, and chose the optimal q and ρ so as to minimize the prediction errors on development data.

5.4.2 Prediction performance

Table 5.2 shows the prediction errors of ratings on test data. On 19 of 25 datasets, GP-LVSM outperforms the other methods. On average of the prediction errors on all datasets, GP-LVSM is the best method. This result indicates GP-LVSM is robust and can perform better prediction than the other methods.

Next, we investigate how the choice of latent dimensionality q and regularizer parameter ρ of GP-LVSM affects the prediction performance. Figure 5.3 shows the prediction errors of GP-LVSM when varying the latent dimensionality q. Here, the regularizer parameter ρ was fixed at $\rho = 10$ to eliminate the effect of ρ. As shown in the figure, even with a very small latent dimensionality, GP-LVSM achieves low prediction error. Even though q is relatively high, the errors are

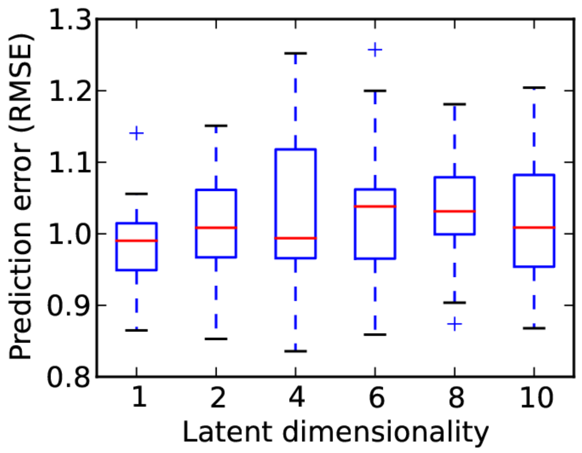


Figure 5.3: Prediction errors of GP-LVSM when varying latent dimensionality. The regularizer parameter is fixed at $\rho = 10$.

nearly unchanged compared to that of the best latent dimensionality. Thus, the performance of GP-LVSM is robust for the dimensionality of the latent vectors for words, and we can obtain vector representations for words on a quite low dimensional space while achieving high prediction performance. Figure 5.4 shows the prediction errors when varying the regularizer parameter ρ. As opposed to the latent dimensionality, the predictive performance is sensitive to the choice of ρ. These results indicate that GP-LVSM can archive the high predictive performance by focusing only on tuning the best ρ.

5.4.3 Visualization

Finally, we show that GP-LVSM can visualize words using two- or three-dimensional latent vectors for words. In our experiments, since we predict the ratings from item reviews, it is expected that positive and negative words for the items are separated from each other. Figure 5.5 shows the visualization result of the latent vectors for words, which are trained on a ‘software’ dataset. Here, the regularizer parameter is fixed at $\rho = 0.1$. For understandability, we selected positive and negative words based on Loughran and McDonald Financial Senti-

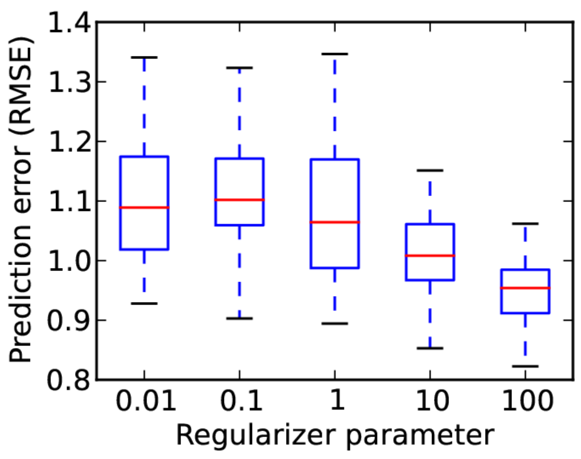


Figure 5.4: Prediction errors of GP-LVSM when varying regularizer parameter. The latent dimensionality is fixed at $q = 2$.

ment Dictionaries[1], and visualized their latent vectors with blue and red colors. As shown in the figure, positive and negative words tend to gather in different regions. Therefore, 'great' and 'cumbersome', which are characteristic words in positive and negative polarity are far away from each other.

5.5 Applications

Since GP-LVSM is a kind of probabilistic generative models, various extended models can be easily constructed based on GP-LVSM. In this section, we discuss two extended models based on GP-LVSM for (1) predicting group behaviors and (2) constructing information diffusion models.

5.5.1 Predicting Group Behaviors

This subsection considers modeling group behaviors, and predicting unseen ones via the model. The group behaviors can be observed in various situations.

[1]http://www3.nd.edu/~mcdonald/Word_Lists.html

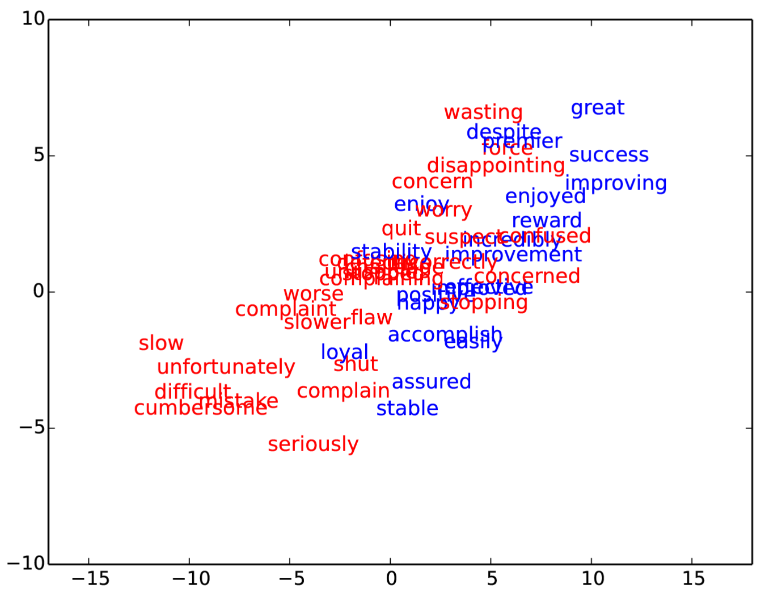


Figure 5.5: Visualization of latent vectors for words trained on 'software' dataset. Words in blue are positive words while words in red are negative words.

For example, a movie is created by collaboration in a group of actors and creators. Then, the contents, the quality and the evaluation of the movie by users correspond to the behaviors of the group. For another example, a scientific paper is written by a research group of researchers. In this case, the contents and the citation behaviors by the paper correspond to the behaviors of the group. By modeling such group behaviors, we can predict the future and the unknown group behaviors.

In this task, we are given a training data $\mathcal{D} = \{(G_i, \mathbf{y}_i)\}_{i=1}^n$, which consists of pairs of a set of members in the ith group G_i and its behavior vector $\mathbf{y}_i \in \mathbb{R}^B$. Here, B is the number of behaviors. Our goal is to construct function f such that $\mathbf{y}_i \approx f(G_i)$ for each $i = 1, 2, \cdots, n$. This problem is a kind of regression problems. Thus, GP-LVSM can be used by regarding each feature as a member, and each datum as a group. However, since GP-LVSM is now a model to predict a single target value for each datum, we need to extend it to treat multiple target variables.

We define behavior matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]^\top$ where the ith row corresponds to the ith behavior vector $\mathbf{y}_i$, while the bth column corresponds to the groups' own values for the bth behavior. Then, according to Eq. (5.9), the likelihood for multiple target variables in GP-LVSM can be rewritten as

$$p(\mathbf{Y}|\mathbf{X}, \mathbf{D}, \alpha, \beta, \gamma) = \prod_{b=1}^{B} \frac{1}{(\sqrt{2\pi})^n \sqrt{\det \mathbf{K}}} \exp\left(-\frac{1}{2}\mathbf{Y}_{:,b}^\top \mathbf{K}^{-1} \mathbf{Y}_{:,b}\right), \tag{5.17}$$

where, $\mathbf{Y}_{:,b}$ indicates the bth column of $\mathbf{Y}$. Coupled with that, the objective function, the logarithm of posterior Eq (5.10), is changed as follows:

$$\begin{aligned} \mathcal{L}(\Theta) &= \log p(\mathbf{Y}|\mathbf{X}, \mathbf{D}, \alpha, \beta, \gamma) + \log p(\mathbf{X}|\rho) \\ &\propto -\frac{1}{2}\sum_{b=1}^{B} \mathbf{Y}_{:,b}^\top \mathbf{K}^{-1} \mathbf{Y}_{:,b} - \frac{1}{2}\log\det \mathbf{K} - \frac{\rho}{2}\sum_{v\in\mathcal{V}} ||\mathbf{x}_v||_2^2. \end{aligned} \tag{5.18}$$

Parameter estimation and prediction in the extended GP-LVSM can be performed as with those of the ogirinal GP-LVSM.

5.5.2 Constructing Information Diffusion Models

An information diffusion model is a model to capture how information diffuses on a social network. We have developed a latent variable model for information diffusion [56], in which each node in the social network has a latent vector, and diffusion probability between two nodes is determined by the inner-product between the latent vectors of the two nodes. Since the model is a linear model, i.e., the model only captures linear relationships between nodes, it is expected that a more accomplished information diffusion model will be developed by applying LDK for capturing non-linear relationships between nodes.

5.6 Summary

In this chapter, we have proposed a non-linear regression model for BoW data, which we call it Gaussian process latent variable set model (GP-LVSM). GP-LVSM performs prediction using a function generated from Gaussian processes (GPs). Since the form of the function is determined kernel values between inputs, we have used LDK in Gaussian processes to generate the function. In our experiments, we have shown that GP-LVSM outperforms conventional linear and non-linear regression methods on the rating prediction using 25 item review datasets, and is useful for visualizing features (words in this case) by using the learned latent vectors for the features. Through Chapter 4 and this chapter, we have shown that LDK works well on the existing algorithms in kernel methods.

Chapter 6

Cross-Domain Matching

6.1 Introduction

The discovery of matched instances in different domains is an important task, which appears in natural language processing, information retrieval and data mining tasks such as finding the alignment of cross-lingual sentences [59], attaching tags to images [12] or text documents [21], and matching user identifications in different databases [34].

When given an instance in a source domain, our goal is to find the instance in a target domain that is the most closely related to the given instance. In this chapter, we focus on a supervised setting, where correspondence information between some instances in different domains is given. To find matching in a single domain, e.g., find documents relevant to an input document, a similarity (or distance) measure between instances can be used. On the other hand, when trying to find matching between instances in different domains, we cannot directly measure the distances since they consist of different types of features. For example, when matching documents in different languages, since the documents have different vocabularies we cannot directly measure the similarities between documents across different languages without dictionaries.

One solution is to map instances in both the source and target domains into a shared latent space. One such method is canonical correspondence analysis (CCA) [16], which maps instances into a latent space by linear projection to maximize the correlation between paired instances in the latent space. However, in practice, CCA cannot solve non-linear relationship problems due to its linear-

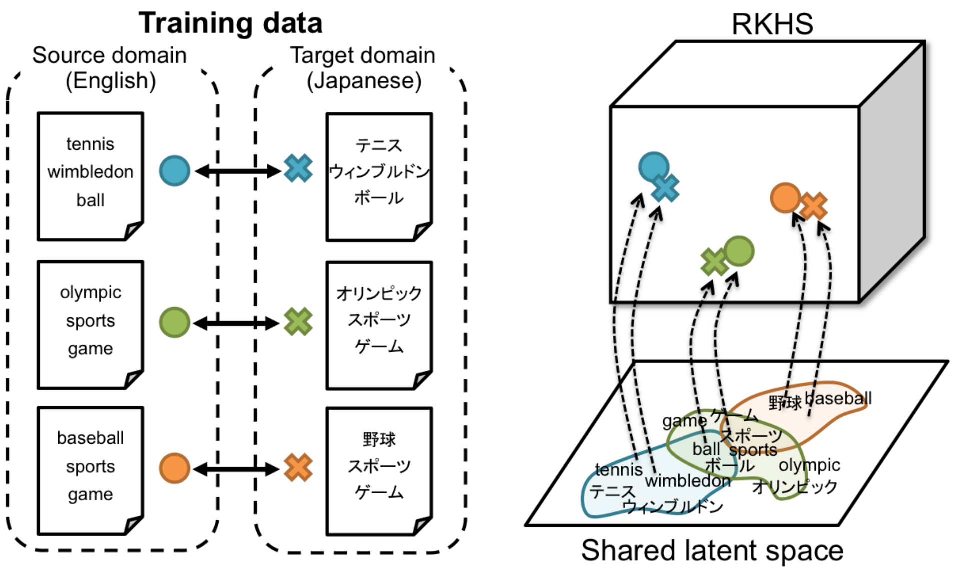


Figure 6.1: An example of the proposed method used on a multilingual document matching task. Correspondences between instances in source (English) and target (Japanese) domains are observed. The proposed method assumes that each feature (vocabulary term) has a latent vector in a shared latent space, and each instance is represented as a distribution of the latent vectors of the features associated with the instance. Then, the distribution is mapped as an element in a reproducing kernel Hilbert space (RKHS) based on the kernel embeddings of distributions. The latent vectors are estimated so that the paired instances are embedded closer together in the RKHS.

ity. To find non-linear correspondence, kernel CCA [1] can be used. It has been reported that kernel CCA performs well as regards document/sentence alignment between different languages [54, 35], when searching for images from text queries [43] and when matching 2D-3D face images [24]. Note that the performance of kernel CCA depends on how appropriately we define the kernel function for measuring the similarity between instances within a domain. Since the existing inner-product kernels have the major weakness as described in Chapter 1, there are cases where kernel CCA does not work better than one expects.

In this chapter, we propose an entirely new kernel-based cross-domain matching method. The proposed method employs LDK to measure the distance between instances in different domains, although latent SMM in Chapter 4 and GP-LVSM in Chapter 5 do that to measure the similarity within a single domain. In particular, it assumes that each feature in source and target domains is associated with a latent vector in a shared latent space. Since all the features are mapped into the latent space, it can measure the similarity between features in different domains. Then, each instance is represented as a set of the latent vectors of features that are contained in the instance, and it is embedded into a reproducing kernel Hilbert space (RKHS) by the framework of the kernel embeddings of distributions. The proposed method assumes that two instances are matched when their distance in the RKHS is the smallest in all possible pairs of instances. Here, the distance is calculated by the distance measurement based on LDK presented in Section 3.3. Thus, in the learning of the proposed method, the latent vectors for features are estimated by minimizing the distances between paired instances while keeping unpaired ones apart. Then, the proposed method predicts the matching of test instances by measuring the distance according to the learned latent vectors. Figure 6.1 shows an example of the proposed method used on a multilingual document matching task.

In our experiments, we demonstrate the effectiveness of our proposed method in tasks that involve finding the correspondence between multi-lingual Wikipedia articles, between documents and tags, and between images and tags, by comparison with existing linear and non-linear matching methods.

The rest of this chapter is organized as follows. Section 6.3 describes the model design, learning and prediction methods of the proposed method. Section 6.4 shows the effectiveness of the proposed method experimentally. Finally, Section 6.5 summarize this chapter.

6.2 Related Work

In this section, we review canonical correlation analysis (CCA) [16], kernel CCA [1] and bilingual topic models [19, 59] that can be used for BoW data, and discuss the difference between the proposed method and these methods.

CCA learns projection vectors so as to maximize the correlation of paired instances in a latent space, and then, by measuring the distance between test instances in the latent space, it can find an unseen matching.

Suppose that we are given two observations $\mathbf{X} \in \mathbb{R}^{n\times u}$ and $\mathbf{Y} \in \mathbb{R}^{n\times v}$ from different domains. Here, there is a one-to-one matching between the ith rows of $\mathbf{X}$ and $\mathbf{Y}$ for each $i = 1, 2, \cdots, n$. Then, CCA finds an optimal projection vectors $\mathbf{a} \in \mathbb{R}^u, \mathbf{b} \in \mathbb{R}^v$ by solving the following problem with a constraint:

$$\max_{\mathbf{a},\mathbf{b}} \mathbf{a}^\top \mathbf{X}^\top \mathbf{Y}\mathbf{b} \quad \text{s.t.} \quad \mathbf{a}^\top \mathbf{X}^\top \mathbf{X}\mathbf{a} = \mathbf{b}^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{b} = 1. \tag{6.1}$$

This corresponds to mapping two instances $\mathbf{X}$ and $\mathbf{Y}$ into a one-dimensional shared latent space. Mapping the instances into a multidimensional shared latent space can be achieved by repeatedly solving Eq. (6.1) such that the correlation between new projection vectors and previously obtained ones is zero. At test phase, matching can be predicted by (1) mapping test instances using learned projection vectors $\mathbf{a}$ and $\mathbf{b}$ into the shared latent space and (2) measuring the distance between the test instance in the shared latent space.

Kernel CCA is a kernel extension version of CCA, and can perform non-linear cross-domain matching. Instead of $\mathbf{X}$ and $\mathbf{Y}$, kernel CCA uses Gram matrices $\mathbf{K}$ and $\mathbf{L}$ with respect to $\mathbf{X}$ and $\mathbf{Y}$, respectively. Here, (i, j) elements of $\mathbf{K}$ and $\mathbf{L}$ are kernel values between the ith and jth rows of $\mathbf{X}$ and $\mathbf{Y}$, respectively. Kernel CCA learns an optimal projection vectors $\mathbf{a}$ and $\mathbf{b}$ by solving the following problem:

$$\max_{\mathbf{a},\mathbf{b}} \mathbf{a}^\top \mathbf{K}\mathbf{L}\mathbf{b} \quad \text{s.t.} \quad \mathbf{a}^\top \mathbf{K}\mathbf{K}\mathbf{a} = \mathbf{b}^\top \mathbf{L}\mathbf{L}\mathbf{b} = 1. \tag{6.2}$$

As with CCA, multidimensional extension and finding matching can be performed.

When we want to match cross-domain instances represented by bag-of-words such as documents, bilingual topic models [19, 59] can also be used. Bilingual topic models assume that words appearing in paired bilingual documents are generated from a shared topic distribution. By learning the model from a bilingual document collection, the topic distribution of each document can be obtained. Then, at test phase, one can find matching by searching pairs of bilingual documents that have similar topic distributions.

The proposed method differs from CCA, kernel CCA and bilingual topic models in two ways. The first difference is that the proposed method is a discriminative matching method, i.e., its objective function is designed to minimize the distances between paired instances while keeping unpaired ones apart. The second one is that the proposed method can use all the information about the latent vectors of features by employing LDK, although the existing methods use the mean of the latent vectors only.

6.3 Proposed Method

Suppose that we are given a training set consisting of n instance pairs $\mathcal{O} = \{(D_i^s, D_i^t)\}_{i=1}^n$, where D_i^s is the ith instance in a source domain and D_i^t is the ith instance in a target domain. These instances D_i^s and D_i^t are represented as multisets of features, i.e., BoW included in source feature set $\mathcal{F}^s$ and target feature set $\mathcal{F}^t$, respectively. The goal of our task is to determine the unseen relationship between instances across source and target domains in test data. The number of instances in the source domain may be different to that in the target domain.

6.3.1 Kernel Embeddings of Distributions in a Shared Latent Space

As described in Section 6.1, the difficulty as regards finding cross-domain instance matching is that the similarity between instances across source and target domains cannot be directly measured. We have also stated that although we can find a latent space that can measure the similarity by using kernel CCA, standard kernel functions, e.g., a Gaussian RBF kernel, cannot reflect the co-occurrence of different but related features in a kernel calculation between instances. To overcome them, we propose to employ LDK for finding cross-domain instance matching. The proposed method assumes that each feature in a source feature set, $f \in \mathcal{F}^s$, has a q-dimensional latent vector $\mathbf{x}_f \in \mathbb{R}^q$ in a shared space. Likewise, each feature in target feature set, $g \in \mathcal{F}^t$, has a q-dimensional latent vector $\mathbf{y}_g \in \mathbb{R}^q$ in the shared space. Since all the features in the source and target domains are mapped into a common shared space, the proposed method can capture the relationship between features both in each domain and across different domains. We define the sets of latent vectors in the source and target domains as $\mathbf{X} = \{\mathbf{x}_f\}_{f \in \mathcal{F}^s}$ and $\mathbf{Y} = \{\mathbf{y}_g\}_{g \in \mathcal{F}^t}$, respectively.

The proposed method assumes that each instance is represented by a distribution (or set) of the latent vectors of the features that are contained in the instance. The ith instance in the source domain D_i^s is represented by a set of latent vectors $\mathbf{X}_i = \{\mathbf{x}_f\}_{f \in D_i^s}$ and the jth instance in the target domain D_j^t is represented by a set of latent vectors $\mathbf{Y}_j = \{\mathbf{y}_g\}_{g \in D_j^t}$.

In the proposed method, we employ the framework of kernel embeddings of distributions to represent the distributions of the latent vectors for the instances. The kernel mean estimators for the ith source and the jth target domain instances are given by

$$m(\mathbf{X}_i) = \frac{1}{|\mathbf{X}_i|} \sum_{f \in D_i^s} k(\cdot, \mathbf{x}_f), \qquad m(\mathbf{Y}_j) = \frac{1}{|\mathbf{Y}_j|} \sum_{g \in D_j^t} k(\cdot, \mathbf{y}_g). \tag{6.3}$$

Then, the distance between the distributions of the latent vectors are measured by using Eq. (3.4), that is, the distance between the ith source and the jth target domain instances is given by

$$d(\mathbf{X}_i, \mathbf{Y}_j) = ||m(\mathbf{X}_i) - m(\mathbf{Y}_j)||^2_{\mathcal{H}_k}. \tag{6.4}$$

6.3.2 Likelihood and Posterior

The proposed method assumes that paired instances have similar distributions of latent vectors and unpaired instances have different distributions. In accordance with the assumption, we define the likelihood of the relationship between the ith source domain instance and the jth target domain instance as follows:

$$p(D_j^t | D_i^s, \mathbf{X}, \mathbf{Y}, \theta) = \frac{\exp\left(-d(\mathbf{X}_i, \mathbf{Y}_j)\right)}{\sum_{j'=1}^{N} \exp\left(-d(\mathbf{X}_i, \mathbf{Y}_{j'})\right)}, \tag{6.5}$$

where, θ is a set of hyper-parameters for the embedding kernel used in Eq. (6.3). Eq. (6.5) is in fact the conditional probability with which the jth target domain instance is chosen given the ith source domain instance. This formulation is more efficient than we consider a bidirectional matching. Intuitively, when distribution $\mathbf{X}_i$ is more similar to $\mathbf{Y}_j$ than other distributions $\{\mathbf{Y}_{j'} \mid j' \neq j\}_{j'=1}^{n}$, the probability has a higher value.

We define the posterior distribution of latent vectors $\mathbf{X}$ and $\mathbf{Y}$. By placing Gaussian priors with precision parameter $\rho > 0$ for $\mathbf{X}$ and $\mathbf{Y}$, that is,

$$p(\mathbf{X}|\rho) \propto \prod_{\mathbf{x} \in \mathbf{X}} \exp\left(-\frac{\rho}{2}||\mathbf{x}||_2^2\right), \quad p(\mathbf{Y}|\rho) \propto \prod_{\mathbf{y} \in \mathbf{Y}} \exp\left(-\frac{\rho}{2}||\mathbf{y}||_2^2\right), \tag{6.6}$$

the posterior distribution is given by

$$p(\mathbf{X}, \mathbf{Y} | \mathcal{O}, \Theta) = \frac{1}{Z} p(\mathbf{X}|\rho) p(\mathbf{Y}|\rho) \prod_{i=1}^{n} p(D_i^t | D_i^s, \mathbf{X}, \mathbf{Y}, \theta), \tag{6.7}$$

where, $\mathcal{O} = \{(D_i^s, D_i^t)\}_{i=1}^n$ is a training set of n instance pairs, $\Theta = \{\theta, \rho\}$ is a set of hyper-parameters and $Z = \int\int p(\mathbf{X}, \mathbf{Y}, \mathcal{O}, \Theta) d\mathbf{X} d\mathbf{Y}$ is a marginal probability, which is constant with respect to $\mathbf{X}$ and $\mathbf{Y}$.

6.3.3 Parameter Estimation

We estimate latent vectors $\mathbf{X}$ and $\mathbf{Y}$ by maximizing the posterior probability of the latent vectors given by Eq. (6.7). Instead of Eq. (6.7), we consider the following negative logarithm of the posterior probability,

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \left\{ d(\mathbf{X}_i, \mathbf{Y}_i) + \log \sum_{j=1}^{n} \exp\left(-d(\mathbf{X}_i, \mathbf{Y}_j)\right) \right\} + \frac{\rho}{2} \left(\sum_{\mathbf{x} \in \mathbf{X}} ||\mathbf{x}||_2^2 + \sum_{\mathbf{y} \in \mathbf{Y}} ||\mathbf{y}||_2^2 \right), \tag{6.8}$$

and minimize it with respect to the latent vectors. Here, maximizing Eq. (6.7) is equivalent to minimizing Eq. (6.8). To minimize Eq. (6.8) with respect to $\mathbf{X}$ and $\mathbf{Y}$, we perform a gradient-based optimization. The gradient of Eq. (6.8) with respect to each $\mathbf{x}_f \in \mathbf{X}$ is given by

$$\frac{\partial \mathcal{L}(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{x}_f} = \sum_{i: f \in D_i^s} \left\{ \frac{\partial d(\mathbf{X}_i, \mathbf{Y}_i)}{\partial \mathbf{x}_f} - \frac{1}{c_i} \sum_{j=1}^{n} e_{ij} \frac{\partial d(\mathbf{X}_i, \mathbf{Y}_j)}{\partial \mathbf{x}_f} \right\} + \rho \mathbf{x}_f \tag{6.9}$$

where,

$$e_{ij} = \exp\left(-d(\mathbf{X}_i, \mathbf{Y}_j)\right), \qquad c_i = \sum_{j=1}^{n} \exp\left(-d(\mathbf{X}_i, \mathbf{Y}_j)\right), \tag{6.10}$$

and the gradient of the distance between distributions $\mathbf{X}_i$ and $\mathbf{Y}_j$ with respect to $\mathbf{x}_f$ is given by

$$\frac{\partial d(\mathbf{X}_i, \mathbf{Y}_j)}{\partial \mathbf{x}_f} = \frac{1}{|\mathbf{X}_i|^2} \sum_{l \in D_i^s} \sum_{l' \in D_i^s} \frac{\partial k(\mathbf{x}_l, \mathbf{x}_{l'})}{\partial \mathbf{x}_f} - \frac{2}{|\mathbf{X}_i||\mathbf{Y}_j|} \sum_{l \in D_i^s} \sum_{g \in D_i^t} \frac{\partial k(\mathbf{x}_l, \mathbf{y}_g)}{\partial \mathbf{x}_f}. \tag{6.11}$$

When the distribution $\mathbf{X}_i$ does not include the latent vector $\mathbf{x}_f$, the gradient consistently becomes a zero vector. $\frac{\partial k(\mathbf{x}_l, \mathbf{x}_{l'})}{\partial \mathbf{x}_f}$ and $\frac{\partial k(\mathbf{x}_l, \mathbf{y}_g)}{\partial \mathbf{x}_f}$ are the gradients of an embedding kernel k. This depends on the choice of the embedding kernel. For example, when the embedding kernel is Gaussian RBF kernel $k_{\mathrm{RBF}}(\mathbf{x}_l, \mathbf{y}_g) = \exp\left(-\frac{\gamma}{2}||\mathbf{x}_l - \mathbf{y}_g||^2\right)$, the gradient is given by

$$\frac{\partial k(\mathbf{x}_l, \mathbf{y}_g)}{\partial \mathbf{x}_f} = k_{\mathrm{RBF}}(\mathbf{x}_l, \mathbf{y}_g) \gamma (\mathbf{y}_g - \mathbf{x}_l). \tag{6.12}$$

Similarly, The gradient of Eq. (6.8) with respect to each $\mathbf{y}_g \in \mathbf{Y}$ is given by

$$\frac{\partial \mathcal{L}(\mathbf{X},\mathbf{Y})}{\partial \mathbf{y}_g} = \sum_{i=1}^{n} \left\{ \frac{\partial d(\mathbf{X}_i,\mathbf{Y}_i)}{\partial \mathbf{y}_g} - \frac{1}{c_i} \sum_{j:g \in D_j^t} e_{ij} \frac{\partial d(\mathbf{X}_i,\mathbf{Y}_j)}{\partial \mathbf{y}_g} \right\} + \rho \mathbf{y}_g, \tag{6.13}$$

where, the gradient of the distance between distributions $\mathbf{X}_i$ and $\mathbf{Y}_j$ with respect to $\mathbf{y}_g$ is given by

$$\frac{\partial d(\mathbf{X}_i,\mathbf{Y}_j)}{\partial \mathbf{y}_g} = \frac{1}{|\mathbf{Y}_j|^2} \sum_{l \in D_j^t} \sum_{l' \in D_j^t} \frac{\partial k(\mathbf{y}_l,\mathbf{y}_{l'})}{\partial \mathbf{y}_g} - \frac{2}{|\mathbf{X}_i||\mathbf{Y}_j|} \sum_{f \in D_i^s} \sum_{l \in D_i^t} \frac{\partial k(\mathbf{x}_f,\mathbf{y}_l)}{\partial \mathbf{y}_g}. \tag{6.14}$$

Learning is performed by alternately updating $\mathbf{X}$ using Eq. (6.9) and updating $\mathbf{Y}$ using Eq. (6.13) until the improvement in the negative log likelihood Eq. (6.8) converges.

6.3.4 Matching Prediction

After the estimation of the latent vectors $\mathbf{X}$ and $\mathbf{Y}$, the proposed method can reveal the matching between test instances. The matching is found by first measuring the distances between a given source domain instance and target domain instances using Eq. (6.4), and then searching for the instance pair with the smallest distance.

6.4 Experiments

In this section, we report our experimental results for three different types of cross-domain datasets: multi-lingual Wikipedia, document-tag and image-tag datasets.

Setup of proposed method

Throughout these experiments, we used a Gaussian RBF kernel as an embedding kernel. The hyper-parameters of the proposed method are the dimensionality of a shared latent space q, a regularizer parameter for latent vectors ρ and a Gaussian RBF embedding kernel parameter γ. After we train the proposed method with various hyper-parameters $q \in \{8, 10, 12\}$, $\rho \in \{0, 10^{-2}, 10^{-1}\}$ and $\gamma \in \{10^{-1}, 10^{0}, \cdots, 10^{3}\}$, we chose the optimal hyper-parameters by using development data. When training the proposed method, we initialized latent vectors $\mathbf{X}$ and $\mathbf{Y}$ by applying principle component analysis (PCA) to a matrix concatenating two feature-frequency matrices in the source and target domains. Then, we employed the L-BFGS method [36] with gradients given by Eqs. (6.9) (6.13) to learn the latent vectors.

Compared methods

We compared the proposed method with the k-nearest neighbor method (KNN), canonical correspondence analysis (CCA), kernel CCA (KCCA), bilingual latent Dirichlet allocation (BLDA), and kernel CCA with the kernel embeddings of distributions (KED-KCCA). For a test instance in the source domain, our KNN searches for the nearest neighbor source instances in the training data, and outputs a target instance in the test data, which is located close to the target instances that are paired with the searched for source instances. CCA and KCCA first learn the projection of instances into a shared latent space using training data, and then they find matching between instances by projecting the test instances into the shared latent space. KCCA used a Gaussian kernel for measuring the similarity between instances and chose the optimal Gaussian kernel parameter and regularizer parameter by using development data. With BLDA, we first learned the same model as [19, 59] and found matching between instances in the test data by obtaining the topic distributions of these instances from the learned model. KED-KCCA uses the kernel embeddings of distributions described in Section 2.2 for obtaining the kernel values between the instances. The vector representations of features were obtained by applying singular value decomposition (SVD) for instance-feature frequency matrices. Here, we set the dimensionality of the vector representations to 100. Then, KED-KCCA learns kernel CCA with the kernel values as with the above KCCA. With CCA, KCCA, BLDA and KED-KCCA, we chose the optimal latent dimensionality (or number of topics) within $\{10, 20, \cdots, 100\}$ by using development data.

Evaluation method

Throughout the experiments, we quantitatively evaluated the matching performance by using the precision with which the true target instance is included in a set of R candidate instances, $\mathcal{S}(R)$, found by each method. More formally, the precision is given by

$$\text{Precision@}R = \frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \delta\left(t_i \in \mathcal{S}_i(R)\right), \tag{6.15}$$

where, N_{te} is the number of test instances in the target domain, t_i is the ith true target instance, $\mathcal{S}_i(R)$ is R candidate instances of the ith source instance and $\delta(\cdot)$ is the binary function that returns 1 if the argument is true, and 0 otherwise.

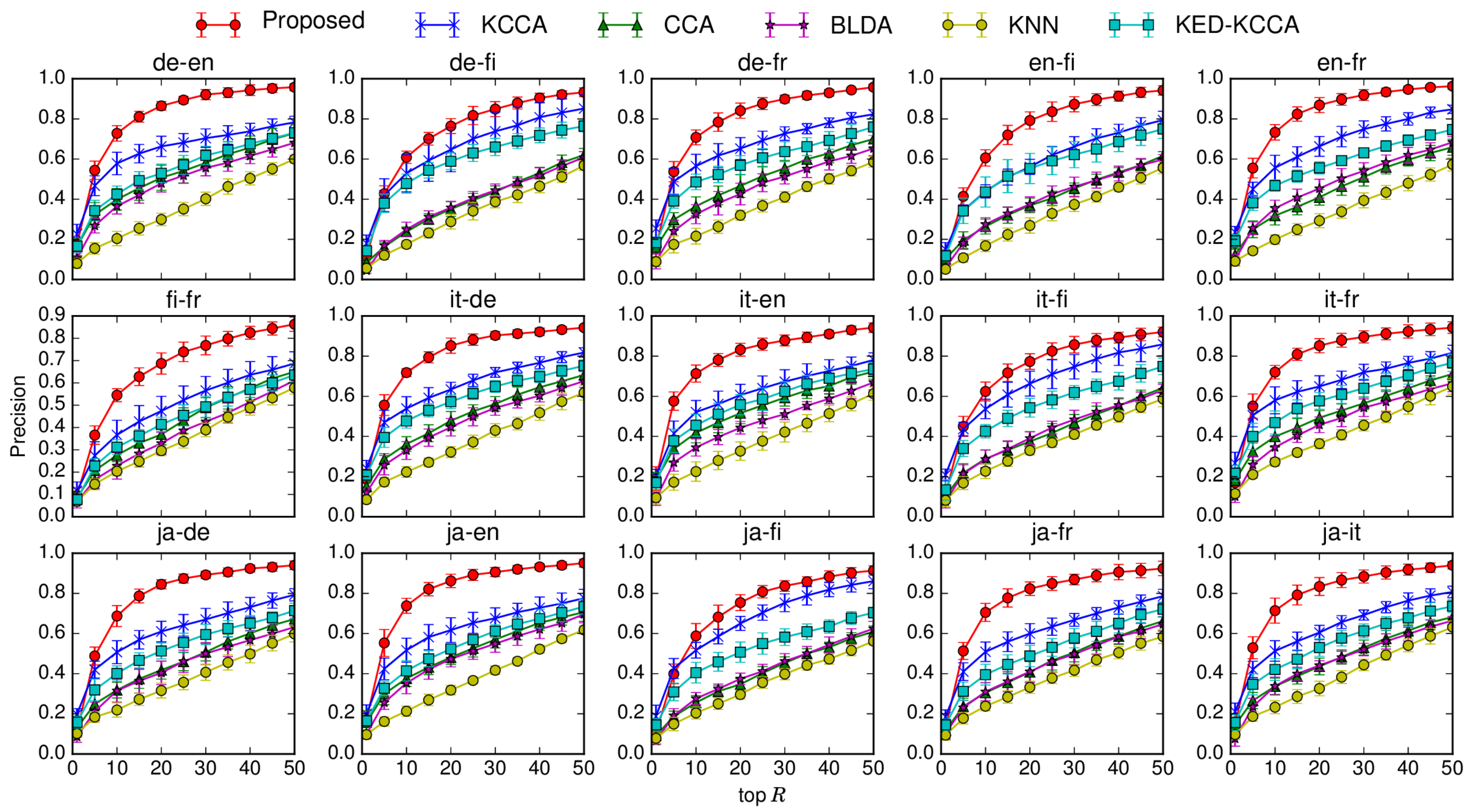


Figure 6.2: Precision of matching prediction and its standard deviation on multi-lingual Wikipedia datasets.

Table 6.1: Top five English documents matched by the proposed method and KCCA given five Japanese documents in the Wikipedia dataset. Titles in bold typeface indicate correct matching.

(a) Japanese Input title: SD カード (SD card)

Proposed	Intel, **SD card**, Libavcodec, MPlayer, Freeware
KCCA	BBC World News, **SD card**, Morocco, Phoenix, 24 Hours of Le Mans

(b) Japanese Input title: 炭疽症 (Anthrax)

Proposed	Psittacosis, **Anthrax**, Dehydration, Isopoda, Cataract
KCCA	Dehydration, Psittacosis, Cataract, Hypergeometric distribution, Long Island Iced Tea

(c) Japanese Input title: ドップラー効果 (Doppler effect)

Proposed	LU deconmposition, Redshift, **Doppler effect**, Phenylalanine, Dehydration
KCCA	Long Island Iced Tea, Opportunity cost, Cataract, Hypergeometric distribution, Intel

(d) Japanese Input title: メキシコ料理 (Mexican cuisine)

Proposed	**Mexican cuisine**, Long Island Iced Tea, Phoenix, Baldr, China Radio International
KCCA	Taoism, Chariot, Anthrax, Digital Millennium Copyright Act, Alexis de Tocqueville

(e) Japanese Input title: フリーウェア (Freeware)

Proposed	BBC World News, Opportunity cost, **Freeware**, NFS, Intel
KCCA	Digital Millennium Copyright Act, China Radio International, Hypergeometric distribution, Taoism, Chariot

6.4.1 Matching between Bilingual Documents

With a multi-lingual Wikipedia document dataset, we examine whether the proposed method can find the correct matching between documents written in different languages. The dataset includes 34,024 Wikipedia documents for each of six languages: German (de), English (en), Finnish (fi), French (fr), Italian (it) and Japanese (ja), and documents with the same content are aligned across the languages. From the dataset, we create ${}_6C_2 = 15$ bilingual document pairs. We regard the first component of the pair as a source domain and the other as a target domain. For each of the bilingual document pairs, we randomly create 10 evaluation sets that consist of 1,000 document pairs as training data, 100 document pairs as development data and 100 document pairs as test data. Here, each document is represented as a bag-of-words without stopwords and low frequency words.

Figure 6.2 shows the matching precision for each of the bilingual pairs of the Wikipedia dataset. With all the bilingual pairs, the proposed method achieves significantly higher precision than the other methods with a wide range of R. Table 6.1 shows examples of predicted matching with the Japanese-English Wikipedia dataset. Compared with KCCA, which is the second best method, the proposed method can find both the correct document and many related documents. For example, in Table 6.1(a), the correct document title is "SD card". The proposed method outputs the SD card's document and documents related to computer technology such as "Intel" and "MPlayer". This is because the proposed method can capture the relationship between words and reflect the distance between documents across different domains by learning the latent vectors of the words.

6.4.2 Matching between Documents and Tags, and between Images and Tags

We performed experiments matching documents and tailgates, and matching images and tailgates with the datasets used in [21]. When matching documents and tailgates, we use datasets obtained from two social bookmarking sites, `delicious`[1] and `hatena`[2], and `patent` dataset. The `delicious` and the `hatena` datasets include pairs consisting of a web page and a tag list labeled by

[1]https://delicious.com/
[2]http://b.hatena.ne.jp/

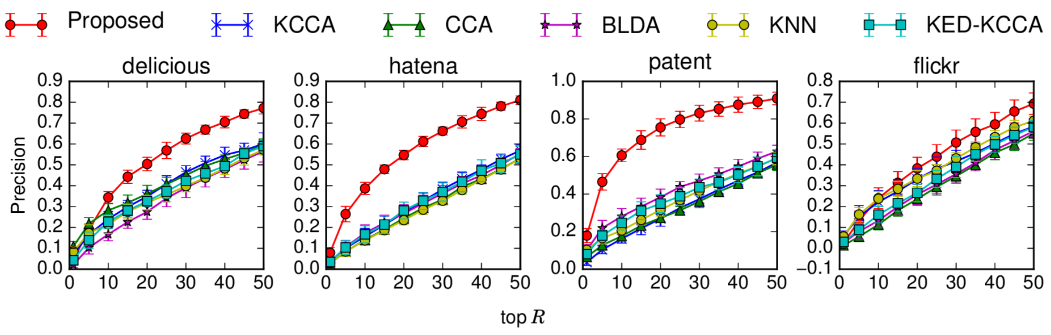


Figure 6.3: Precision of matching prediction and its standard deviation on `delicious`, `hatena`, `patent` and `flickr` datasets.

users, and the patent dataset includes pairs consisting of a patent description and a tag list representing the category of the patent. Each web page and each patent description are represented as a bag-of-words as with the experiments using the Wikipedia dataset, and the tag list is represented as a set of tags. With the matching of images and tag lists, we use the `flickr` dataset, which consists of pairs of images and tag lists. Each image is represented as a bag-of-visual-words, which is obtained by first extracting features using SIFT [38], and then applying K-means clustering with 200 components to the SIFT features. For all the datasets, the numbers of training, test and development pairs are 1,000, 100 and 100, respectively.

Figure 6.3 shows the precision of the matching prediction of the proposed and comparison methods for the `delicious`, `hatena`, `patent` and `flickr` datasets. The precision of the comparison methods with these datasets was much the same as the precision of random prediction. Nevertheless, the proposed method achieved very high precision particularly for the `delicious`, `hatena` and `patent` datasets. Figure 6.4 shows examples of input tag lists and the top five images matched by the proposed method with the `flickr` dataset. In the examples, the proposed method found the correct images and similar related images from given tag lists.

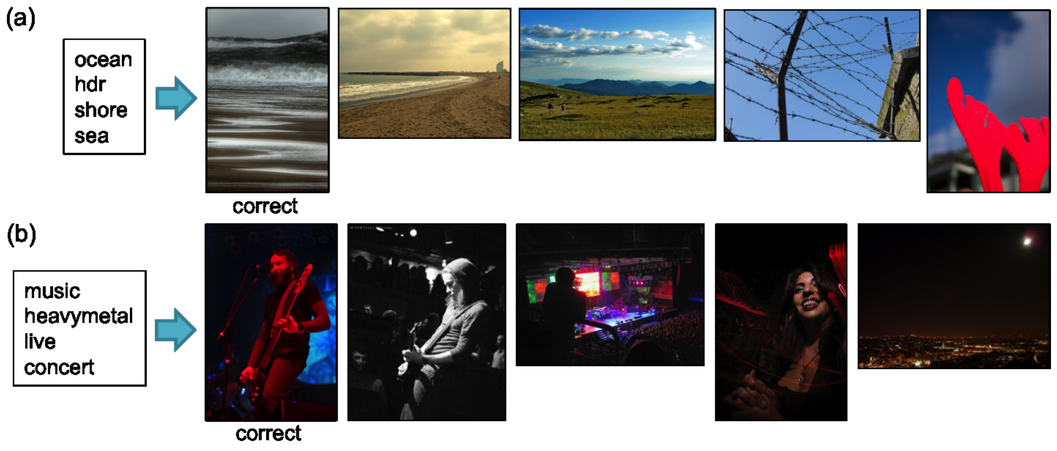


Figure 6.4: Two examples of input tag lists and the top five images matched by the proposed method on the `flickr` dataset.

6.5 Summary

We have proposed a novel kernel-based method for addressing cross-domain instance matching tasks for BoW data. In the proposed method, we have employed LDK to measure the distance between instances in different domains, although latent SMM in Chapter 4 and GP-LVSM in Chapter 5 do that in a single domain. Experiments on various types of cross-domain datasets confirmed that the proposed method significantly outperforms the existing methods including kernel CCA and topic model-based approach for cross-domain matching.

Chapter 7

Conclusion

This thesis has proposed a general framework of kernel methods for bag-of-words (BoW) data. This framework consists of the following two parts: (1) defining a class of kernel functions with latent variables appropriate for BoW data, which we call it *latent distribution kernel (LDK)*, and (2) developing models with LDK and their optimization methods.

In LDK, each feature has a latent vector, and each datum is represented as a set of the latent vectors of the features associated with the datum. Here, the latent vectors are learned according to the problem we want to solve. To represent the set of the latent vectors for each datum, we have employed the framework of kernel embeddings of distributions. The benefits that we employ the framework are as follows:

- According to the latent vectors, the moment information of the distribution that generates the latent vectors for each datum can be obtained nonparametrically.
- Since the framework is based on kernel methods, the existing strong machine learning methods such as support vector machines (SVMs) and Gaussian processes (GPs) can be utilized.
- The problem that the inner-product kernels cannot capture the correlation between different features in a kernel calculation can be overcome naturally.

LDK can be used by incorporating itself into the existing kernel-based algorithms such as SVMs, or to develop new kernel-based algorithms. To demonstrate the effectiveness of LDK, we have developed three methods based on LDKs for adressing their corresponding machine learning problems: classification, regression and cross-domain matching. First, we have developed a novel non-linear discriminative learning method for BoW data classification, which is formulated based on SVMs. In the experiments, we have showed that the proposed method achieves the state-of-the-art accuracy on BoW text categorization tasks. Second, we have developed a non-linear regression method for BoW data by incorporating LDKs into GPs. In the experiments, we have showed that the proposed method outperforms the existing linear and non-linear regression methods on item review score prediction. Third, we have developed a cross-domain matching method for BoW data, in which LDKs are used for matching data in different domains. The experimental results have showed that the proposed matching method outperforms the existing methods on multi-lingual document, document-tag, and image-tag matching.

7.1 Future Work

There are two future directions to advance our research.

The first direction is to make the learning of the proposed methods efficient. Although the proposed methods based on LDK are superior to the existing ones in terms of prediction accuracy, the time to learn the proposed methods is basically longer than the existing kernel methods. The most time-consuming part in the learning is the calculation of the gradients with respect to the latent vectors. In this thesis, we have presented straightforward implementations for the learning, i.e., batch learning to evaluate whether the proposed methods are valid or not. Since this thesis have showed that the proposed methods are superior to the existing methods in terms of prediction accuracy, in future work, we will develop faster learning methods to apply to larger datasets. In particular, two approaches can be considered. The first approach is to use stochastic gradient descent (SGD) [5]. SGD is a gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. In each iteration of SGD, parameters are updated by optimizing the objective function for a single sample that is randomly chosen from training samples. In practice, the SGD-based learning is more efficient in terms of time and space complexity than a batch learning that uses the whole training samples in each iteration. Since the objective functions of the proposed methods described in Chapters 4 and 5 are a sum of objective functions for two sample pairs, these methods can use the SGD-based learning without any change of their formulation. The second one is to use kernel approximation such as Nyström method [10] and random feature method [42]. Kernel approximation speeds up learning by approximating a kernel matrix, and is successfully used for SVM [58] and kernel CCA [37]. By applying this for the proposed methods in this thesis, it is expected that their learning is more efficient.

The second direction is to develop new methods for other machine learning problems and applications based on LDK. LDK is a general framework for supervised learning. Basically, LDK can be applied to the existing kernel-based supervised learning methods as described in Chapter 4 and 5. Thus, we will confirm the effectiveness of LDK by applying to other supervised learning such as learning to rank and structured learning. Additinally, since BoW data appears in various tasks in computer vision and data mining, we will explore to apply LDK in these other fields.

Bibliography

[1] S. Akaho. A Kernel Method for Canonical Correlation Analysis. *Proceedings of International Meeting on Psychometric Society*, (4), 2001.

[2] D. M. Blei and J. D. McAuliffe. Supervised Topic Models. *Advances in Neural Information Processing Systems*, 2010.

[3] D. M. Blei, A. Y. Ng, and M. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[4] E. Bonilla, K. Chai, and C. Williams. Multi-task Gaussian Process Prediction. In *Advances in Neural Information Processing Systems*, 2008.

[5] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT'2010*, pp. 177–186, 2010.

[6] A. Cardoso-Cachopo. *Improving Methods for Single-label Text Categorization*. PhD thesis, 2007.

[7] V. Cherkassky and Y. Ma. Practical Selection of SVM Parameters and Noise Estimation for SVM Regression. *Neural Networks*, 17(1):113–26, Jan. 2004.

[8] T. Cohn and L. Specia. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[9] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, Sept. 1995.

[10] P. Drineas and M. W. Mahoney. On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[11] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support Vector Regression Machines. *Advances in Neural Information Processing Systems*, pp. 155–161, 1996.

[12] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik. A Multi-View Embedding Space for Modeling Internet Images, Tags, and Their Semantics. *International Journal of Computer Vision*, 106(2):210–233, Oct. 2013.

[13] A. Gretton, K. Fukumizu, C. Teo, L. Song, B. Schölkopf, and A. Smola. A Kernel Statistical Test of Independence. In *Advances in Neural Information Processing Systems*, 2008.

[14] Z. S. Harris. Distributional Structure. *Word*, 10:146–162, 1954.

[15] A. Hoerl and R. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

[16] H. Hotelling. Relations Between Two Sets of Variants. *Biometrika*, 28:321–377, 1936.

[17] C.-W. Hsu and C.-J. Lin. A Comparison of Methods for Multi-class Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[18] H. Isozaki and H. Kazawa. Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th International Conference on Computational linguistics*, 2002.

[19] T. Iwata, S. Watanabe, and H. Sawada. Fashion Coordinates Recommender System Using Photographs from Fashion Magazines. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.* AAAI Press, July 2011.

[20] T. Iwata, T. Yamada, and N. Ueda. Probabilistic Latent Semantic Visualization: Topic Model for Visualizing Documents. In *ACM SIGKDD Interna-*

tional Conference on Knowledge Discovery and Data Mining, pp. 363–371, 2008.

[21] T. Iwata, T. Yamada, and N. Ueda. Modeling Social Annotation Data with Content Relevance using a Topic Model. *Advances in Neural Information Processing Systems*, 2009.

[22] T. Joachims. *Text categorization with support vector machines: Learning with many relevant features.* 1998.

[23] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *International Conference on Knowledge Discovery and Data Mining*, New York, New York, USA, 2002. ACM Press.

[24] P. Kamencay, R. Hudec, M. Benco, and M. Zachariasov. 2D-3D Face Recognition Method Based on a Modified CCA-PCA Algorithm. *International Journal of Advanced Robotic Systems*, 2014.

[25] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active Learning with Gaussian Processes for Object Categorization. *IEEE 11th International Conference on Computer Vision*, 2007.

[26] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian Processes for Object Categorization. *International Journal of Computer Vision*, 88(2):169–188, July 2009.

[27] P. Kolari, T. Finin, and A. Joshi. SVMs for the Blogosphere: Blog Identification and Splog Detection. *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pp. 92–99, 2006.

[28] T. Kudo and Y. Matsumoto. Japanese Dependency Structure Analysis Based on Support Vector Machines. *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 18–25, 2000.

[29] T. Kudo and Y. Matsumoto. Chunking with Support Vector Machines. *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, 816, 2001.

[30] V. Lampos and N. Aletras. Predicting and Characterising User Impact on Twitter. In *Proceedings of the 52st Annual Meeting of the Association for Computational Linguistics*, pp. 405–413, 2014.

[31] N. Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In *Advances in Neural Information Processing Systems*, 2004.

[32] N. Lawrence and R. Urtasun. Non-linear Matrix Factorization with Gaussian Processes. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[33] D. D. Lee and H. S. Seung. Algorithms for Non-Negative Matrix Factorization. *Advances in Neural Information Processing Systems*, pp. 556–562, 2001.

[34] B. Li, Q. Yang, and X. Xue. Transfer Learning for Collaborative Filtering via a Rating-Matrix Generative Model. *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[35] Y. Li and J. Shawe-Taylor. Using KCCA for Japanese – English cross-language information retrieval and document classification. *Journal of Intelligent Information Systems*, 27(2):117–133, Sept. 2006.

[36] D. C. Liu and J. Nocedal. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(1-3):503–528, Aug. 1989.

[37] D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schölkopf. Randomized Nonlinear Component Analysis. In *Proceeding of International Conference of Machine Learning*, Vol. 32, 2014.

[38] D. G. Lowe. Object Recognition fromLocal Scale-Invariant Features. *The proceedings of the seventh IEEE international conference on Computer vision*, 2:1150–1157, 1999.

[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, 2013.

[40] K. Muandet, K. Fukumizu, F. Dinuzzo, and B. Schölkopf. Learning from Distributions via Support Measure Machines. *Advances in Neural Information Processing Systems*, 2012.

[41] D. Preotiuc-Pietro and T. Cohn. A Temporal Model of Text Periodicities Using Gaussian Processes. In *The 2013 Conference on Empirical Methods on Natural Language Processing.*, pp. 977–988, 2013.

[42] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. *Advances in Neural Information Processing Systems*, 2007.

[43] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos. A New Approach to Cross-Modal Multimedia Retrieval. In *Proceedings of the International Conference on Multimedia*, 2010.

[44] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, Dec. 2005.

[45] M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, Sept. 1956.

[46] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, a. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural computation*, 13(7):1443–1471, 2001.

[47] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and P. John. Support Vector Method for Novelty Detection. *Advances in Neural Information Processing Systems*, 12:582–588, 1999.

[48] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert Space Embedding for Distributions. In *Algorithmic Learning Theory.* 2007.

[49] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[50] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. G. Lanckriet. Hilbert Space Embeddings and Metrics on Probability Measures. *The Journal of Machine Learning Research*, 11:1517–1561, 2010.

[51] J. B. Tenenbaum, V. D. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(December):2319–2323, 2000.

[52] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1996.

[53] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[54] A. Vinokourov, J. Shawe-Taylor, and N. Cristianini. Inferring a Semantic Representation of Text via Cross-Language Correlation Analysis. In *Advances in Neural Information Processing Systems*. 2003.

[55] C. Yang, K. H.-Y. Lin, and H.-H. Chen. Emotion Classification Using Web Blog Corpora. *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 275–278, Nov. 2007.

[56] Y. Yoshikawa, T. Iwata, and H. Sawada. Information Diffusion Model with Latent Features of Users on Social Networks (in Japanese). *IPSJ Transactions on Databases*, 6(5):85–94, 2013.

[57] D. Zhang and W. S. Lee. Question Classification Using Support Vector Machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, New York, New York, USA, 2003. ACM Press.

[58] K. Zhang, L. Lan, Z. Wang, and F. Moerchen. Scaling up Kernel SVM on Limited Resources: A Low-rank Linearization Approach. *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, pp. 1425–1434, 2012.

[59] T. Zhang, K. Liu, and J. Zhao. Cross Lingual Entity Linking with Bilingual Topic Model. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[60] J. Zhu, A. Ahmed, and E. Xing. MedLDA: Maximum Margin Supervised Topic Models for Regression and Classification. *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[61] H. Zou and T. Hastie. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B*, 67:301–320, 2005.

List of Publications

Journal Papers (refereed)

- 吉川 友也, 岩田 具治, 澤田 宏, “ソーシャルメディア上の協働: ソーシャルコーディングにおける成功するプロジェクトの要因分析,” 日本データベース学会論文誌, Vol.12, No.3, Feb. 2014.
- 吉川 友也, 岩田 具治, 澤田 宏, “ユーザの潜在特徴を考慮したソーシャルネットワーク上の情報拡散モデル,” 情報処理学会論文誌 データベース (TOD), Vol.6, No.5, pp.85-94, Dec. 2013.

Letters (refereed)

- 吉川 友也, 岩田 具治, 澤田 宏, “グループ推薦のための非線形行列分解,” 人工知能学会論文誌, Vol.30, No.2, Mar. 2015.

International Conferences (refereed)

- Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada, Takeshi Yamada, “Cross-Domain Matching for Bag-of-Words Data via Kernel Embeddings of Latent Distributions,” Advances in Neural Information Processing Systems (NIPS2015), Montreal, Canada, Dec. 2015.
- Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada, “Non-linear Regression for Bag-of-Words Data via Gaussian Process Latent Variable Set Model,”

The 28th AAAI Conference on Artificial Intelligence (AAAI2015), Austin, Texas, USA, Jan. 2015.

- Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada, “Latent Support Measure Machines for Bag-of-Words Data Classification,” Advances in Neural Information Processing Systems (NIPS2014), Montreal, Canada, Dec. 2014.
- Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada, “Latent Feature Independent Cascade Model for Social Propagation,” International Conference on Parallel and Distributed Processing Techniques and Application (PDPTA2013), Las Vegas, USA, July 2013.

Preprint Paper

- Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada, “Collaboration on Social Media: Analyzing Successful Projects on Social Coding,” Arxiv:1408.6012, Aug. 2014.

Domestic Conferences (refereed)

- 吉川 友也, 岩田 具治, 澤田 宏, “ソーシャルコーディングにおけるプロジェクト成功の法則,” 第6回 Web とデータベースに関するフォーラム (WebDB Forum 2013), 京都, Nov. 2013.
- 吉川 友也, 岩田 具治, 澤田 宏, “ユーザの潜在特徴を考慮した情報拡散モデルの学習,” 第5回 Web とデータベースに関するフォーラム (WebDB Forum 2012), 東京, Nov. 2012.

Domestic Conferences (non-refereed)

- 吉川 友也, 岩田 具治, 澤田 宏, 山田 武士, “潜在分布のカーネル埋め込みによる異種データ間マッチング,” 第29回人工知能学会全国大会 (JSAI2015), 函館, May 2015.
- 澤井 裕一郎, 吉川 友也, 松本 裕治, “文書分類におけるサポートメジャーマシンの有効性,” 言語処理学会第21回年次大会 (NLP2015), 京都, Mar. 2015.

- 吉川 友也, 岩田 具治, 澤田 宏, "Bag-of-Words データ分類のための潜在サポートメジャーマシン," 第17回情報論的学習理論ワークショップ (IBIS2014), 愛知, Nov. 2014.
- 吉川 友也, 岩田 具治, 澤田 宏, "グループ推薦のための非線形行列分解," 第28回人工知能学会全国大会 (JSAI2014), 愛媛, May 2014.
- 吉川 友也, 岩田 具治, 澤田 宏, "ネットワーク上の情報に対する拡散現象と潜在的トピックの同時学習," 第27回人工知能学会全国大会 (JSAI2013), 富山, Jun 2013.
- 三澤 賢祐, 酒井 啓道, 吉川 友也, 水本 智也, 松本 裕治, "格構造に注目した日本語-日本語手話の並び替えと述語項構造に注目した語義曖昧性解消," 第27回人工知能学会全国大会 (JSAI2013), 富山, Jun 2013.
- 吉川 友也, 藤野 拓也, 酒井啓道, 三谷 亮介, 江崎 大嗣, "Twitter からの人口推定," 東日本大震災ビッグデータワークショップ -Project 311-, 東京, Oct. 2012.
- 吉川 友也, 岩田 具治, 澤田 宏, "トピックを考慮したソーシャルネットワーク上の情報拡散モデル, " 第26回人工知能学会全国大会 (JSAI2012), 山口, Jun. 2012.
- 吉川 友也, 酒井 啓道, 高橋 弘志, 柏木 潔, 三谷 亮介, "RT のコメントを使ったデマ分類," NLP 若手の会第6回シンポジウム, 奈良, Sep. 2011.

Awards

- 全国大会優秀賞, 第28回人工知能学会全国大会 (JSAI2014), 2014.
- 学生奨励賞・企業賞, 第5回 Web とデータベースに関するフォーラム (WebDB Forum 2012), 2012.