NAIST-IS-DD0561205

Doctoral Dissertation

Domain Adaptation of Statistical Word Segmentation System

Yuta Tsuboi

March 17, 2009

Department of Information Processing
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

Yuta Tsuboi

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Kiyohiro Shikano	(Co-supervisor)
Associate Professor Kentaro Inui	(Co-supervisor)

Domain Adaptation of Statistical Word Segmentation System*

Yuta Tsuboi

Abstract

It is not trivial to detect word boundaries for non-segmented languages such as Japanese or Chinese. Although, statistical methods have been successfully used for word segmentation tasks, they tend to perform poorly as the domain changes because of differences in vocabulary and writing style.

In the first part of this thesis, we address word-boundary annotation which is done only on parts of sentences. By limiting our focus on the crucial parts of sentences, we can effectively create a training data for each new target domain by using partial annotations. We propose a training algorithm for *Conditional Random Fields* (CRFs) using partial annotations. It is known that CRFs are well-suited to word segmentation tasks. However, conventional CRF learning algorithms require fully annotated sentences. The objective function of the proposed method is a marginal likelihood function, so that the CRF model can handle the partial annotations.

The second part of this thesis describes an *importance weighting* approach for domain adaptation. We propose a novel method that allows us to directly estimate *importance*, which is the ratio of test and training densities, from samples. An advantage of the proposed method is that the computation time is nearly independent of the number of target input samples, which is highly beneficial in word segmentation tasks with large numbers of unlabeled samples.

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561205, March 17, 2009.

Through experiments, we demonstrate that our approaches improve the performance of statistical models on a domain adaptation task of Japanese word segmentation.

Keywords:

Word Segmentation, Domain Adaptation, Conditional Random Fields, Partial Annotation, Density Ratio Estimation, Covariate Shift Adaptation

Acknowledgements

My first thanks must go to Professor Yuji Matsumoto, my supervisor. He is a wonderful advisor, and every aspect of this thesis has benefitted from his insight. I was very fortunate to be supervised by him and to be the beneficiary of his great knowledge, friendly persona, and tireless passion. He has generously spent much time discussing research ideas with me anytime I visited him. In addition to him, this thesis was shaped by a great committee: Professor Kiyohiro Shikano and Professor Kentaro Inui. I would like to thank Prof. Shikano for the remarkable breadth and depth of his feedback. I am also grateful to Prof. Inui for his invaluable advice on my work and I was greatly inspired by his passion to research work.

I would like to acknowledge my co-authors on work that has contributed largely to this thesis. Special thanks to Kashima Hisashi who had a deep influence on my research work. He was like a second advisor to me and I was fortunate enough to have a chance to work with him. I would like to express my gratitude to Shinsuke Mori and Hiroki Oda. The idea described in Chapter 3 is jointly developed with them. The work described in Chapter 4 are based on the line of work by Professor Masashi Sugiyama. I owe a very important debt to him for generous and proactive support and encouragement. Shohei Hido and Steffen Bickel give me insightful comments and suggestions to develop the idea in Chapter 4.

During my doctoral study, I have received the financial support from IBM Japan, Ltd. I feel grateful to Hideo Watanabe, my manager, for giving me enough freedom to explore my own interests. I also acknowledge Koich Takeda for the recommendation of the financial support. I have received many help from colleagues in IBM Tokyo Research Laboratory. Most notably, I would like to thank the members of the Text Mining group for the various valuable discussions.

Discussions with Masashi Shimbo, Masayuki Asahara, Ryu Iida, and all the other member of Computational Linguistics Laboratory (CL-lab.) in NAIST helped clarify and deepen the ideas presented here. I would also like to thank former students of the CL-lab: Tatsuo Yamashita, Hiroshi Matsuda, Hiroya Takamura, Hiroyasu Yamada, Kaoru Yamamoto, Daichi Mochihashi, Taku Kudo, Satoru Takabayashi, and Tetsuji Nakagawa. Their research work encouraged

me to work in computational linguistics. My thesis research would not be possible without an encounter with them during master's programs. I also thank the members of the T-PRIMAL (Tokyo PRobabilistic Inference and MAchine Learning) group for many helpful discussions related to this thesis. I was fortunate to join this great and passionate academic community.

Finally, and most importantly, I would like to thank my parents and my wife Mika. My parents provided an enormous amount of support which have made all of my work. I was fortunate to have great academic parents. Mika helped me go through all the difficulties to complete this work. I can't thank her enough. I dedicate this thesis to her.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is not trivial to detect word boundaries for non-segmented languages such as Japanese or Chinese since the words are not separated by spaces. For example, the correct segmentation of the Japanese phrase "切り傷やすり傷" (incised wound or abrasion) is shown by the bottom row of boxes segmented by the solid lines in Figure 1.1. However, there are several overlapping segmentation candidates, which are shown by the other boxes, and possible segmentation shown by the dashed lines.

In those non-segmented languages, natural language processing (NLP) initially requires word segmentation at the beginning, and the resulting segmented sentences can be the input to subsequent linguistic analyzers, such as syntactic parsing, and named entity recognition. These subsequent processes are highly dependent on the results of the word segmentations, so errors in the word segmentation task cause errors throughout the process and greatly degrade the performance of NLP applications.

The decisions on the word segmentation require considering the context, so simple dictionary lookup approach is not sufficient. Therefore statistical methods have been successfully used for Japanese Word Segmentation (JWS) tasks. However, in practice, a statistical word segment analyzer tends to perform worse with text from different domains. A major cause of errors is the occurrence of unknown words. For example, if "すり傷" (abrasion) is an unknown word, the system may accept the word sequence of "切り傷やすり傷" as "切り傷" (incised wound), "やすり" (file), and "傷" (injury) by mistake. In addition, the boundary

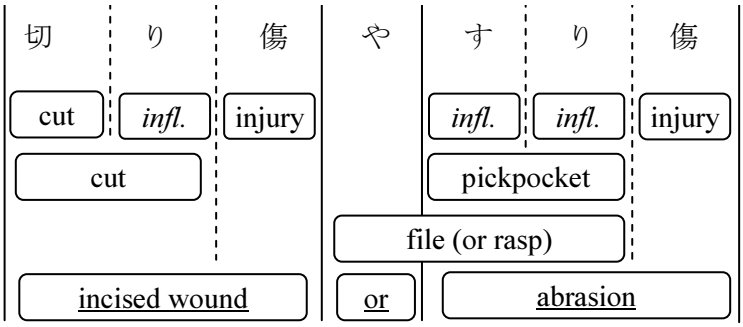


Figure 1.1. An example of word boundary ambiguities: *infl.* stands for an inflectional suffix of a verb.

agreement is even worse in Chinese [38].

One approach for domain adaptation is to use additional annotations for each target domain. However, it is time-consuming to annotate all of the elements in a sentence. It is much more efficient to focus on annotating certain parts of sentences that include domain-specific expressions. In Chapter 3, we will describe the effectiveness of partial annotations in the domain adaptation task for JWS. This motivated us to seek to incorporate such incomplete annotations into a state-of-the-art machine learning technique. One of the recent advances in statistical NLP is Conditional Random Fields (CRFs) [37] that evaluate the global consistency of complete structures for both parameter estimation and structure inference, instead of optimizing the local configurations independently. This feature is suited to many NLP tasks that include correlations between elements in the output structure. Previous work [36] showed CRFs outperform generative Markov models and discriminative history-based methods in JWS. However, conventional CRF algorithms require fully annotated sentences. To incorporate incomplete annotations into CRFs, we have extended the structured output problem. We also propose a parameter estimation method for CRFs using partial annotated corpora. The proposed method marginalizes out the unknown labels so as to optimize the likelihood of a set of possible label structures that are consistent with given incomplete annotations.

Since a large amount of unsegmented text tends to be available for target domains, another approach for domain adaptation is the modification of statistical

models for going from a source domain to a target domain using these unannotated samples of the target domain. In Chapter 4, we describe such a modification technique, *covariate shift adaptation*. A situation where the input distribution $p(\boldsymbol{x})$ is different in the training and test phases but the conditional distribution $p(y|\boldsymbol{x})$ of output values remains unchanged is called *covariate shift* [56]. If we assume covariate shift exists between source and target domains, we can play word segmentation learning to minimize the segmentation error in the target domain using segmented samples from the source domain. The key idea is weighting the source domain samples by importance, which is the ratio of test and training densities. We propose a novel method that allows us to directly estimate the importance from samples without going through the hard task of density estimation. An advantage of the proposed method is that the computation time is nearly independent of the number of test input samples, which is highly beneficial in recent applications with large numbers of unlabeled samples. The proposed method is computationally more efficient than existing approaches but remains comparable accuracy.

Through experiments, we demonstrate that our approaches improve the performance of statistical models on a domain adaptation task of JWS.

Chapter 2

Related Work

1. Learning by Example

Many state-of-the-art NLP systems are based on some form of statistical learning [27], and Manning and Schütze [40] have provided a great introduction to its use in NLP. Here, we briefly review supervised learning.

Let $\boldsymbol{x} \in \boldsymbol{X} \subset \Re^d$ be an input variable and $y \in Y$ be an output variable. In many NLP problems, supervised learning attempts to learn how to map an input text $\boldsymbol{x}$ to an output y from examples. We call these examples the *training examples*. In standard supervised learning settings, we assume that an example $(\boldsymbol{x}, y)$ is drawn from independent identical distribution (i.i.d.) $p(\boldsymbol{x}, y)$. A common approach for supervised learning is finding a model which minimizes the error of the training examples, and we use this process as the training procedures in this thesis.

2. Statistical Word Segmentation

The generative model of word sequences had been a common approach for statistical word segmentation systems. Let $\boldsymbol{x} = (x_1, \cdots, x_T)$ be a character sequence of length T and $\boldsymbol{y} = (y_1, \cdots, y_W)$ be a output word sequence of length W in this section. In the generative model, the generation process of a sample $(\boldsymbol{x}, \boldsymbol{y})$, i.e. a joint probability $p(\boldsymbol{x}, \boldsymbol{y})$, is estimated from the training examples. Since $p(\boldsymbol{x}, \boldsymbol{y})$

is too complex to estimate directly from a limited number of examples, we usually assume the *Markov* property so that the joint probability can be factorized as:

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y}),$$
$$p(\boldsymbol{x}|\boldsymbol{y}) = \prod_{t=1}^{W} p(x_{i_t,\cdots,j_t}|y_t), \tag{2.1}$$
$$p(\boldsymbol{y}) = p(y_1)\prod_{t=2}^{W} p(y_t|y_{t-1}), \tag{2.2}$$

where $x_{i_t,\cdots,j_t}$ denotes a character subsequence making up the word y_t. Equation 2.1 is called the *emission probability* and Equation 2.2 is called the *transition probability*. In the maximum likelihood procedure, these probabilities are estimated from the relative frequencies in the training data. Once these probabilities are estimated, we can predict the most probable word sequence for given $\boldsymbol{x}$ using

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x}) \propto \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{x}, \boldsymbol{y}).$$

Although there is exponential number of possible word segmentations, we can efficiently find the best word segmentation $\hat{\boldsymbol{y}}$ by using a dynamic programming procedure [42].

When this generative model involves unknown words that do not appear in the training data, the most commonly used approach is a hierarchical model [43]. This hierarchical model includes the probabilities of an unknown word, *UW*, defined as

$$p(y_t = \mathrm{UW}|y_{t-1})$$
$$p(x_{i_t,\cdots,j_t}|y_t = \mathrm{UW}). \tag{2.3}$$

In the same way as the transition probability of $\boldsymbol{y}$, Equation 2.3 is factorized into the product of $p(x_{i_t}|x_{i-1_t}, y_t = \mathrm{UW})$:

$$p(x_{i_t,\cdots,j_t}|y_t = \mathrm{UW}) = p(x_{i_t}|y_t = \mathrm{UW}) \prod_{s=i_t+1}^{j_t} p(x_s|x_{s-1}, y_t = \mathrm{UW})$$

All of these probabilities for the unknown words are estimated from the relative frequencies in the training data where the infrequent words are replaced with UW.

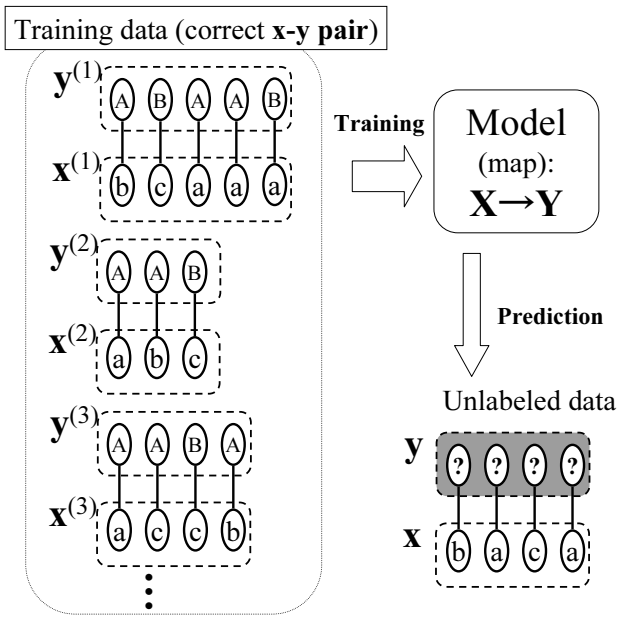


Figure 2.1. Supervised structured output learning.

3. Structured Output Learning

One of the recent advances in statistical NLP involves using discriminative learning methods applicable to produce structured outputs, and many core NLP tasks can now be interpreted as structured output tasks. In structured output learning, we assume the output $\boldsymbol{y} \in \boldsymbol{Y}$ has the structure of elements y. For example, part-of-speech (POS) tagging is the task of outputting the appropriate sequence structure of the POS tags for a given input sentence. Syntactic parsing is also considered as a task to output phrase structure trees. Again, we assume that an example $(\boldsymbol{x}, \boldsymbol{y})$ is drawn from i.i.d. $p(\boldsymbol{x}, \boldsymbol{y})$. Figure 2.1 depicts a supervised procedure for structured output learning in which $(\boldsymbol{x}, \boldsymbol{y})$ is represented by a graphical model where $X = \{a, b, c\}$ and $Y = \{A, B\}$.

The state-of-the-art approaches for structured output evaluate the global consistency of the entire structures for both parameter estimation and structure inference, instead of optimizing the local configurations independently. Starting with Conditional Random Fields (CRFs) [37], those discriminative structured output approaches have been applied to the NLP tasks that include correlations

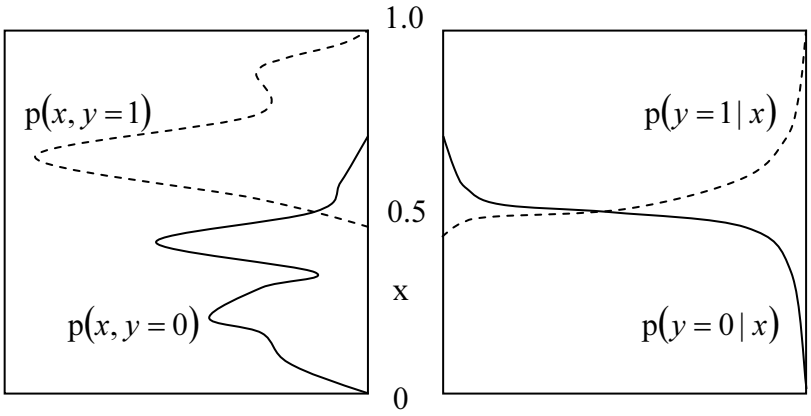


Figure 2.2. Joint $p(x, y)$ vs conditional distribution $p(y|x)$. The solid lines denotes the distribution of $y = 0$ and the dashed lines denotes that of $y = 1$.

between elements in the output structures [20, 53, 66, 51, 67].

The advantage of CRFs in the word segmentation task is that 1) they can involve various kind of features compared with the N-gram models, and 2) they can incorporate label interactions in the training phase.

First, a generative N-gram model [43] for word segmentation specifies a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ of input character sequences and word segmentations. However, the joint distribution can be complex because it includes the distribution of input character sequences $p(\boldsymbol{x})$. Therefore, it is not practical to involve the overlapping features, such as the characters themselves and the character types, to empirically estimate the generative N-gram models from the limited number of given samples. The left graph of Figure 2.2 depicts a joint distribution of one dimensional data where $x = [0, 1]$ and $y = \{0, 1\}$.

In contrast, discriminative models, including CRFs, specify a conditional distribution of word segmentations given input character sequences, so that it does not require the modeling of input features. The right graph of Figure 2.2 shows a conditional distribution which is simpler than the joint distribution. Therefore, we can design features freely without the assumption of feature independence. For example, although the character boundaries between different character types also tend to be word boundaries in Japanese, there are many exceptional cases. The combination use of a character and its character types can handle both these normal and exceptional cases.

Second, before CRFs were proposed, a discriminative classifier predicted for each character boundary whether or not it was a word boundary. However, the decisions at a character boundary t are correlated to the decisions at $t-1$ and $t+1$, as shown in Figure 1.1. CRFs can naturally represent the correlations of word boundary decisions by using label pairs as features, such as y_{t-1}, y_t. [1]

In Chapter 3, we extend CRFs to deal with partial labels (annotations).

4. Domain Adaptation Techniques

Although Section 3 noted that the training example errors will be minimized in learning procedures, we are not interested in the training set error, but the goal of supervised learning is to finding a model which generalizes well for unseen test data. In practice, we cannot assume identical distributions for the training examples and test samples. Domain adaptation of the statistical models is the learning problem where the data distribution in the source domain is different from that of the target domain. Since domain adaptation is required in not only in the word segmentation problem but also in many NLP problems, recently it has started to receive a great deal of attention. At the same time, some kind of domain adaptation problems have been addressed as different names within other fields, such as covariate shift [55], sample selection bias [31], transfer learning [22], and multi-task learning [9], and semi-supervised learning [65].

The problem settings for domain adaptation are different between each work. In the NLP literature, there is a large amount of labeled data available in certain source domain and a relatively large amount of unlabeled data available in various target domain. In Chapter 4, we introduce an importance estimation method that exploits the huge volumes of target domain data.

In addition, sometimes we also have a small amount of labeled data in a target domain. To the best of our knowledge, most of the domain adaptation research

[1] Before CRFs were proposed, there are a discriminative model which incorporates label pair features, called Maximum Entropy Markov Model (MEMM), had been proposed [69]. A MEMM predicts the value of y_t when the previous labels $y_1, y_2, \cdots, y_{t-2}, y_{t-1}$ are given. Therefore, MEMM has some drawbacks since it has a biased preference to a label pairs, as known as a label bias problem [37], and it shows the different results between forward prediction models and backward prediction models.

focuses on the training procedure in that sort of situation. A few studies in the domain adaptation research have been conducted on the reduction of the labeling effort in the target domain [16]. In relation to this kind of work, in Chapter 3 we propose a training procedure for the structured output learning that incorporates the efficient creation of labeled target domain data.

Although the complete description is beyond the scope of this thesis, we can organize the existing work into several categories. The first line of work is based on instance weighting. Jiang and Zhai proposed a general framework of instance weighting for NLP and empirically showed that instance weighting is beneficial for the domain adaptation of NLP [32]. They assign instance-dependent weights to the loss function while minimizing the expected loss over the distribution of the target data. However, they selected these weights with heuristics and did not provide any theoretical background for weight estimation. In Chapter 4, we provide a partial solution for the estimation procedure of weight estimation. The second approach in the existing work involves the changes of feature representation that minimize the expected losses in both source and target domains. Blitzer et al. proposed a structural correspondence learning (SCL) algorithm that finds a projected (low-ranked) feature space using the unlabeled data in the source and target domain [12]. Ben-David et al. empirically showed that this projected representation decreases the distance between the distributions of the source and target domains [6], which are the important quantities for computing the target domain error. Daumé III proposed a feature augmentation technique that simply adds features for the source and target domains into the original features[23]. This technique assumes that some labeled data for the target domain is available. Even if the classification rules are different in the source and target domains, the weights of these augmented features will be learned correctly for each domain. The third and last approach is maximum a posterior (MAP) estimation approach in which we encode source domain knowledge into a prior distribution of the model parameters [17].

Chapter 3

Training Conditional Random Fields using Partial Annotations

1. Partial Annotations

Annotated linguistic corpora are essential for building statistical NLP systems. Most of the corpora that are well-known in NLP communities are completely-annotated in general. In domain adaptation situations, it is time-consuming to annotate all of the elements in a sentence. Rather, it is efficient to annotate certain parts of sentences which include domain-specific expressions. Lists of new terms in the target domain are often available in the forms of technical term dictionaries, product name lists, or other sources. To utilize these domain word lists, Mori proposed a KWIC (KeyWord In Context) style annotation user interface (UI) with which a user can delimit a word in a context with a single user action [41]. In Figure 3.1, an annotator marks the occurrences of "すり傷", a word in the domain word list, if they are used as a real word in their context. The "すり傷" in the first row is a part of another word "こすり傷" (scratch), and the annotator marks the last two rows as correctly segmented examples. This UI simplifies annotation operations for segmentation to yes/no decisions, and this simplification can also be effective for the reduction of the annotation effort for other NLP tasks. For example, the annotation operations for unlabeled dependency parsing can be simplified into a series of yes/no decisions as to whether or not given two words have syntactic dependency. Compared

	感染、角膜のこ	すり傷	、角膜潰瘍、
○	皮膚に切り傷や	すり傷	を負った場合
○	泥まみれの深い	すり傷	や、皮下深く

Figure 3.1. An example of KWIC style annotation: marked lines are identified as a correct segmentation.

with sentence-wise annotation, the partial annotation is not only effective in terms of control operations, but also reduces annotation errors because it does not require annotating the word boundaries that an annotator is unsure of. This feature is crucial for annotations made by domain experts who are not linguists. [1] We believe partial annotation is effective in creating corpora for many other structured annotations in the context of the domain adaptations.

In addition to manual annotations, it should be possible to generate a partially segmented text from a raw target domain text using heuristic rules. A simple rule is segmentation using special symbol characters (e.g. commas, and HTML tags). Some symbols such as commas [2] can be removed from a sentence without violating its grammaticality. Careful selection of such symbols enables automatic partial annotation of segmentation boundaries by detecting such symbols and removing them from sentences. Examples of automatically annotated texts are shown in Figure 3.2. The first row of Figure 3.2 depicts the generation of annotated text by the rule deleting commas, and the second row of Figure 3.2 depicts the generation of annotated text by the rule deleting an HTML link tag. In this examples, at least, we know there exists word boundaries between two characters “け消”, “良胃”, “てわ”, and “きさ” in the generated texts. Note that this rule annotates only either the beginning or ending of words in many cases, and never annotates non-word boundary positions. Thus, we can automatically annotate parts of sentences without human hand.

In summary, compared with the complete annotations, it is easier to obtain the partial annotations for the target domain.

[1] The boundary policies of some words are different even among linguists. In addition, the boundary agreement is even lower in Chinese [38].

[2] Commas are optional and have a position flexibility in Japanese sentences.

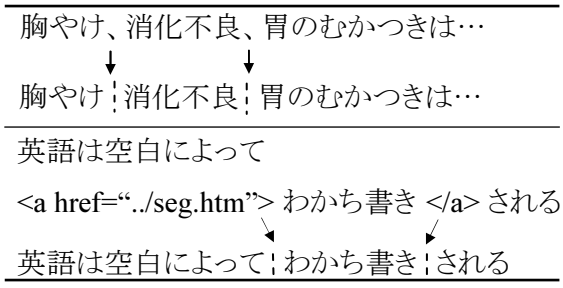


Figure 3.2. Examples of rule annotations by symbols: The dashed lines denote annotated positions.

2. Problem Formalization

In this section, we give a formal definition of the domain adaptation task of JWS using partial annotations.

In this thesis we assume that the input is the sequence of character boundaries and the output is the sequence of the corresponding labels, which specify whether the current position is a word boundary.[3]

Let $\boldsymbol{x}=(x_1, x_2, \cdots, x_T)$ be a sequence of observed variables $x_t \in X$ which represent the context of the t-th character boundary, and $\boldsymbol{y}=(y_1, y_2, \cdots, y_T)$ be the sequence of label variables $y_t \in Y$. For JWS, we define $Y = \{\bigcirc, \times\}$ where $\bigcirc$ represents word boundary label and $\times$ represents non-word boundary. Then, the supervised structured output problem for JWS can be defined as learning a map $\boldsymbol{X} \rightarrow \boldsymbol{Y}$.

Let $\boldsymbol{L}=(L_1, L_2, \cdots, L_T)$ be a sequence of label subsets for an observed sequence $\boldsymbol{x}$, where $L_t \in 2^Y - \{\emptyset\}$ represents the label candidates for the t-th position. The partial annotation at position s is where L_s is a singleton and

[3]Peng et al. defined the word segmentation problem as labeling each character as whether or not the previous character boundary of the current character is a word boundary [45]. However, we employ our problem formulation since it is redundant to assign the first character of a sentence as the word boundary in their formulation. In addition, Kudo et al. defined JWS task as finding the correct label path from the morpheme lattice made by dictionary lookup, and proposed to use CRFs for this task [36]. However, since the dictionary entries defines the candidate label paths, candidate paths do not include unknown words. Therefore we need to build some heuristic rules to generate unknown word candidate entries in the morpheme lattice. On the other hand, our formulation do not require such heuristics to handle unknown words.

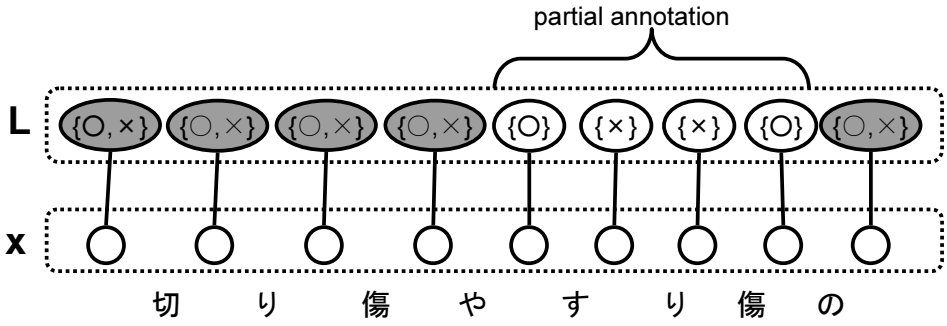


Figure 3.3. An example of partial annotations.

the rest $L_{t\neq s}$ is $\boldsymbol{Y}$. For example, if a sentence with 6 character boundaries (7 characters) is partially annotated using the KWIC UI described in Section 1, a word annotation where its boundary begins with $t = 2$ and ends with $t = 5$ will be represented as:

$$\boldsymbol{L} = (\{\bigcirc, \times\}, \underbrace{\{\bigcirc\}, \{\times\}, \{\times\}, \{\bigcirc\}}_{\text{partial annotation}}, \{\bigcirc, \times\}),$$

where $\bigcirc$ and $\times$ denote the word boundary label and the non-word boundary label, respectively. Figure 3.3 shows another example of partial annotations. In this example, "すり傷" is partially annotated in the phrase "切り傷やすり傷".

Note that, if all the elements of a given sequence are annotated, it is the special case such that the size of all elements is one, i.e. $|L_t| = 1$ for all $t = 1, \cdots, T$, so that this representation of annotations includes non-partial annotations as a special case. Comparing with the conventional structured output learning in Figure 2.1, a structured output learning procedure with partial annotations is depicted in Figure 3.4. The shaded elements in Figure 3.4 represent positions which are not annotated, i.e. $|L_t| > 1$.

Finally, we define the domain adaptation of word segmentation using these partial annotations. Let $D_{\mathrm{s}} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{L}^{(n)})\}_{n=1}^{N}$ be annotated sentences of size N for the source domain, and $D_{\mathrm{t}} = \{(\boldsymbol{x}^{(m)}, \boldsymbol{L}^{(m)})\}_{m=N+1}^{N+M}$ be partially annotated sentences of size M for the target domain. In this chapter, the goal of the domain adaptation is the improvement of the word segmentation performance using both D_{s} and D_{t} rather than using only D_{s}. [4]

[4]For the domain adaptation task, it is conceivable that we only use the target domain

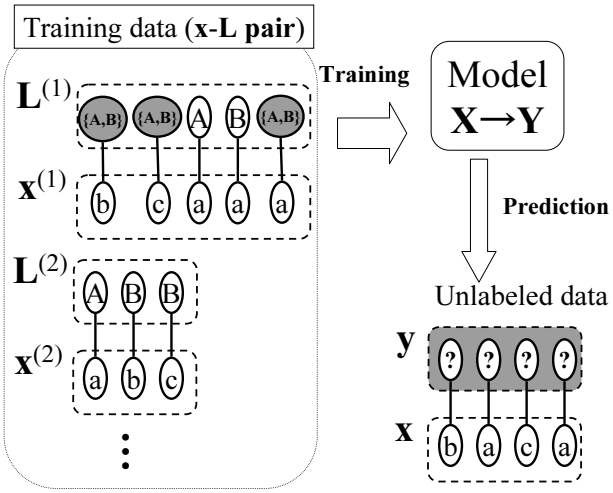


Figure 3.4. Structured output learning with partial annotations.

3. Conditional Random Fields

In this section, we review Conditional Random Fields (CRFs) [37].

Let $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{X} \times \boldsymbol{Y} \rightarrow \Re^d$ denote a map from a pair of an observed sequence $\boldsymbol{x}$ and a label sequence $\boldsymbol{y}$ to an arbitrary feature vector of d dimensions, and $\boldsymbol{\theta} \in \Re^d$ denotes the vector of the model parameters. CRFs model the conditional probability of a label sequence $\boldsymbol{y}$ given an observed sequence $\boldsymbol{x}$ as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle}}{Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y}}}, \tag{3.1}$$

where $\langle , \rangle$ denotes the inner product of the vectors, and the denominator is the normalization term that guarantees the model to be a probability:

$$Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{S}} = \sum_{\boldsymbol{y} \in \boldsymbol{S}} e^{\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle}.$$

$Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{S}}$ is also known as a *partition function*. CRFs are generalization of *logistic regression* (cf. Section 2.2 of Chapter 4) for structured output.

data D_{t}. However, we assume the combination of D_{s} and D_{t} is worth the target domain performance when the number of the target domain annotations is limited. In addition, the word segmentation performance using both the source and target domain data was empirically better than that using the target domain data in the preliminary experiments of the Section 6.

Using training examples D, the optimal parameter $\hat{\boldsymbol{\theta}}$ is obtained by the maximum likelihood estimator:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \left(\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y})\rangle - \ln Z_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{Y}}\right).\end{aligned}$$

This CRF learning is stated as a unconstrained maximization problem of a nonlinear function. Since the objective function of CRFs is a concave function [5], we can calculate $\hat{\boldsymbol{\theta}}$ by the gradient methods [53] using its partial derivative:

$$\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y}) - \sum_{\boldsymbol{y}\in\boldsymbol{Y}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y}). \tag{3.2}$$

Then once $\hat{\boldsymbol{\theta}}$ has been estimated from training examples, the label sequence can be predicted by:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y}\in\boldsymbol{Y}} p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}|\boldsymbol{x}).$$

Although Equation (3.1) includes the sum of all the possible configurations $\boldsymbol{Y}$ of label sequences, we can efficiently calculate it by a dynamic programming technique if we assume the Markov properties [37]. Even if we cannot assume the Markov properties for a task, we can use some approximation methods, such as *Loopy Belief Propagation*, for this kind of tasks [63].

4. Marginalized Likelihood Training of CRFs

In this section, we propose a parameter estimation procedure for the CRFs incorporating partial annotations.

Since the original CRF learning algorithm requires a completely labeled sequence $\boldsymbol{y}$, the incompletely annotated data $(\boldsymbol{x}, \boldsymbol{L})$ is not directly applicable to it.

[5] If a real value function f satisfies

$$f(a\boldsymbol{u} + (1-a)\boldsymbol{v}) \geq af(\boldsymbol{u}) + (1-a)f(\boldsymbol{v})$$

for arbitrary points, $\boldsymbol{u}$ and $\boldsymbol{v}$, where $0 \leq a \leq 1$, we state f as a concave function [44].

Let $\boldsymbol{Y_L}$ denote all of the possible label sequence consistent with $\boldsymbol{L}$. We propose to use the conditional probability of the subset $\boldsymbol{Y_L}$ given $\boldsymbol{x}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y_L}|\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}), \tag{3.3}$$

which is the marginal probability eliminating the candidate $\boldsymbol{y}$s by summation. Then the maximum likelihood estimator for this model can be obtained by maximizing the log likelihood function:

$$\begin{aligned} \mathrm{LL}(\boldsymbol{\theta}; D) &= \sum_{(\boldsymbol{x},\boldsymbol{L}) \in D} \ln p_{\boldsymbol{\theta}}(\boldsymbol{Y_L}|\boldsymbol{x}) \\ &= \sum_{(\boldsymbol{x},\boldsymbol{L}) \in D} \sum_{\boldsymbol{y} \in \boldsymbol{Y_L}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \sum_{(\boldsymbol{x},\boldsymbol{L}) \in D} \left(\ln Z_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{Y_L}} - \ln Z_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{Y}} \right). \end{aligned} \tag{3.4}$$

This modeling naturally embraces label ambiguities in the incomplete annotation.

Unfortunately, Equation (3.4) is not a concave function[6] so that there are local maxima in the objective function. Although this non-concavity prevents efficient global maximization of Equation (3.4), it still allows us to incorporate incomplete annotations using gradient ascent iterations [53].[7] Gradient ascent methods require the partial derivative of Equation (3.4):

$$\frac{\partial \, \mathrm{LL}(\boldsymbol{\theta}; D)}{\partial \boldsymbol{\theta}} = \sum_{(\boldsymbol{x},\boldsymbol{L}) \in D} \left(\sum_{\boldsymbol{y} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L}, \boldsymbol{x}) \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \sum_{\boldsymbol{y} \in \boldsymbol{Y}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \right), \tag{3.5}$$

where

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L}, \boldsymbol{x}) = \frac{e^{\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y}) \rangle}}{Z_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{Y_L}}} \tag{3.6}$$

is a conditional probability that is normalized over $\boldsymbol{Y_L}$. Since the value of Equation (3.5) is zero at the optimum, the maximization of the objective function can be stated as finding $\boldsymbol{\theta}$ to be match the feature expectation of $\boldsymbol{\Phi}(\boldsymbol{x}^{(n)}, \boldsymbol{y})$ over

[6] $\mathrm{LL}(\boldsymbol{\theta}; D)$ is the form of the difference of two concave function so that it is not concave function as a whole [14].

[7] The local maxima depend on the initial value of $\boldsymbol{\theta}$. We used the parameter of a CRF trained in the source domain as the initial value for the proposed algorithm in Section 6.

Table 3.1. Two types of example distributions: $\boldsymbol{L} = (\{\bigcirc, \times\}, \{\times\})$

$\boldsymbol{x}$	$\boldsymbol{y}$	$p_{\boldsymbol{\theta}}(\boldsymbol{y}\|\boldsymbol{x})$	$p_{\boldsymbol{\theta}}(\boldsymbol{y}\|\boldsymbol{Y_L}, \boldsymbol{x})$
(a, b)	$(\bigcirc, \bigcirc)$	0.2	0
	$(\times, \bigcirc)$	0.4	0
	$(\bigcirc, \times)$	0.3	0.75
	$(\times, \times)$	0.1	0.25

$p_{\boldsymbol{\theta},\boldsymbol{L}}(\boldsymbol{y}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, denoted as $\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L},\boldsymbol{x})}[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})]$ and $\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})]$. The derivative (3.2) of the conventional CRF learning procedure suggests that the learning algorithm iteratively minimizes the distance between $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ and $\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})]$. Comparing with the conventional CRF learning, the expectation $\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L},\boldsymbol{x})}[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})]$ is considered as the feature vector of a partially annotated training example in the proposed method.

Here, we explain how the proposed algorithm updates the model parameter with a simplified example. Let $(\boldsymbol{x} = (a, b), \boldsymbol{L} = (\{\bigcirc, \times\}, \{\times\}))$ be a partially annotated example where $X = \{a, b\}$. For a given $\boldsymbol{\theta}$, let's say $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ for each $\boldsymbol{x}$ and $\boldsymbol{y}$ pair is calculated as the values in the third column of Table 3.1, and $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L}, \boldsymbol{x})$ is calculated as the values in the forth column of Table 3.1. Suppose we use the sum of the occurrences of X and Y pairs as features, i.e., $\phi_{x,y}(\boldsymbol{x}, \boldsymbol{y}) = \sum_t [x_t = x \wedge y_t = y]$, the feature expectations under each distribution can be the value described in Table 3.2 where $\theta_{x,y}$ is the parameter corresponding to $\phi_{x,y}(\boldsymbol{x}, \boldsymbol{y})$. In this example, we can see that the information of the partial annotation at $t = 2$ propagates to the expected value of features which appeared in the adjacent element at $t = 1$ and the corresponding parameter of the features will be updated.

Equations (3.4) and (3.5) include the summations of all of the label sequences in $\boldsymbol{Y}$ or $\boldsymbol{Y_L}$. It is not practical to enumerate and evaluate all of the label configurations explicitly, since the number of all of the possible label sequences is exponential on the number of positions t with $|L_t| > 1$. However, under the Markov assumption, a modification of the *Forward-Backward algorithm* guarantees polynomial time computation for the Equations (3.4) and (3.5). We explain this algorithm in Appendix A.

Table 3.2. The example values of feature expectations and derivatives

x	y	$\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y})}[\phi_{x,y}(\boldsymbol{x},\boldsymbol{y})]$	$\mathrm{E}_{\boldsymbol{y}\sim p_{\boldsymbol{\theta}}(\boldsymbol{y}\mid\boldsymbol{Y_L},\boldsymbol{x})}[\phi_{x,y}(\boldsymbol{x},\boldsymbol{y})]$	$\frac{\partial \mathrm{LL}(\boldsymbol{\theta};D)}{\partial \theta_{x,y}}$
a	◯	0.5	0.75	0.25
a	×	0.5	0.25	−0.25
b	◯	0.6	0	−0.6
b	×	0.4	1	0.6

In the implementation of CRF learning, it is common to introduce a prior distribution $p(\boldsymbol{\theta})$ over the parameter $\boldsymbol{\theta}$ to avoid over-fitting [53]. In the experiments in Section 6, we used a *Gaussian* prior with the mean 0 and the variance σ^2 so that $-\frac{||\boldsymbol{\theta}||^2}{2\sigma^2}$ is added to Equation (3.4). In addition, it is natural to introduce a parameter which specifies the balancing weight between the likelihood of D_{S} and D_{T} for the objective function [32]. In this section, we denote ν as this weight parameter for the log likelihood of D_{T}.

Now, the objective function including the Gaussian prior and the weight parameter is:

$$\mathrm{LL}(\boldsymbol{\theta}; D_{\mathrm{S}}) + \nu\, \mathrm{LL}(\boldsymbol{\theta}; D_{\mathrm{T}}) - \frac{||\boldsymbol{\theta}||^2}{2\sigma^2},$$

and its first order derivative is defined as:

$$\frac{\partial\, \mathrm{LL}(\boldsymbol{\theta}; D_{\mathrm{S}})}{\partial \boldsymbol{\theta}} + \nu \frac{\partial\, \mathrm{LL}(\boldsymbol{\theta}; D_{\mathrm{T}})}{\partial \boldsymbol{\theta}} - \frac{\boldsymbol{\theta}}{\sigma^2}.$$

Note that σ and ν are the hyper parameters in the learning algorithm, and have to be selected in validation phases.

5. Discussion

In this section, we discuss other methods which can adapt statistical word segmentation system to a target domain using the partial annotations $\boldsymbol{L}$, and compare these methods with the proposed method.

One approach to train CRFs using $\boldsymbol{L}$ is that the predicted label for unannotated positions $\{t \mid |L_t| > 1\}$ are used to generate completely annotated examples. To elaborate this approach, let $\tilde{\boldsymbol{\theta}}$ be a parameter vector which is obtained

by training using the source domain data. For a given partially annotated sample $(\boldsymbol{x}, \boldsymbol{L})$, we can use $(\boldsymbol{x}, \hat{\boldsymbol{y}})$ as a training data where

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \boldsymbol{Y_L}} p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{y}|\boldsymbol{x})$$

is the most probable sequence of $p_{\tilde{\boldsymbol{\theta}}}$ which is consistent with $\boldsymbol{L}$. Note that $\hat{\boldsymbol{y}}$ can be computed in polynomial time by a variant of dynamic programming technique with given $\tilde{\boldsymbol{\theta}}$, $\boldsymbol{x}$, and $\boldsymbol{L}$ [21]. Since we select $\hat{\boldsymbol{y}}$ from the label sequences which are consistent with $\boldsymbol{L}$, $\hat{\boldsymbol{y}}$ reflects the partial annotations. We denote this method as *argmax*. The advantage of this approach is that, once $\hat{\boldsymbol{y}}$ is predicted for $(\boldsymbol{x}, \boldsymbol{L})$, a CRF model can be trained by the conventional training algorithms. However, the trained model of argmax as training data only reflects the most probable sequence, and ignores the rest of all the possible $\boldsymbol{y} \in \boldsymbol{Y_L}$ even if the most probable candidate $\hat{\boldsymbol{y}}$ and the other candidates have probabilities $p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{y}|\boldsymbol{x})$ with narrow margin. Since the selection of $\hat{\boldsymbol{y}}$ can be sensitive to the fluctuation of the parameter estimation in the source domain data, the variance of estimated parameters will be large when we train CRFs using argmax in the target domain. At the same time, the parameter estimation of the proposed method is relatively stable because the expectation of $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ under $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{Y_L}, \boldsymbol{x})$ is used as a training example. In other words, instead of a single label sequence $\hat{\boldsymbol{y}}$, all the possible $\boldsymbol{y} \in \boldsymbol{Y_L}$ are dealt in the training phase of the proposed method.

As another possible solution, we can design JWS as a prediction problem in which each y_t is independently predicted by a classifier. In this formulation, the training of these predictors using partial annotations is rather simple because we can just ignore unannotated positions and only use annotated positions $\{t \mid |L_t| = 1\}$ as training examples. In addition, we can employ discriminative methods for this prediction problem to incorporate various kinds of features. We denote this approach as a *point-wise classifier*. As we mentioned in Section 3, the disadvantage of point-wise classifiers is that the previous and next segmentation of a position t does not effect the decision of y_t so that the consistency of word segmentation in a whole sentence is not learned by this approach. The empirical result in Section 6, the advantage of CRFs over point-wise classifiers remains even if training data includes partial annotations.

Table 3.3. Data statistics.

	domain	usage	#sentences	#words
(A)	conversation	training	11,700	145,925
(B)	conversation	validation	1,300	16,348
(C)	medical manual	test & annotation	1,000	29,216

Table 3.4. Feature templates: Each subscript stands for the relative distance from a character boundary.

Types	Template
Characters	$c_{-1}, c_{+1},$
Character types	$c_{-2}c_{-1}, c_{-1}c_{+1}, c_{+1}c_{+2},$
Term in dictionary	$c_{-2}c_{-1}c_{+1}, c_{-1}c_{+1}c_{+2}$
Term in dictionary starts at	c_{+1}
Term in dictionary ends at	c_{-1}

6. Experiments

In this section, we show the results of domain adaptation experiments for the JWS task to assess the proposed method. We assume that only partial annotations are available for the target domain. In this experiment, the corpus for the source domain is composed of example sentences in a dictionary of daily conversation [35]. The text data for the target domain is composed of sentences in a medical reference manual [5] . The sentences of all of the source domain corpora (A), (B) and a part of the target domain text (C) were manually segmented into words (see Table 3.3).

The performance measure in the experiments is the standard F measure score,

$F = 2RP/(R + P)$ where

$$R = \frac{\#\text{ of correct words}}{\#\text{ of words in test data}} \times 100$$
$$P = \frac{\#\text{ of correct words}}{\#\text{ of words in system output}} \times 100.$$

In this experiment, the performance was evaluated using 2-fold cross-validation that averages the results over two partitions of the data (C) into the data for annotation and training (C1) versus the data for testing (C2).

We implemented first order Markov CRFs. As the features for the observed variables, we use the characters and character type n-gram (n=1, 2, 3) around the current character boundary. The character types are categorized into *Hiragana*, *Katakana*, *Kanji*, English alphabet, Arabic numerals, and symbols. The usage of character type features reflects the background knowledge which the character boundaries between different character types tend to be word boundaries in Japanese. We also used lexical features consulting a dictionary: one is to check if any of the above defined character n-grams appear in a dictionary [45], and the other is to check if there are any words in the dictionary that start or end at the current character boundary. We used the *unidic*[8] (281K distinct words) as the general purpose dictionary, and the *Japanese Standard Disease Code Master* (JSDCM)[9] (23K distinct words) as the medical domain dictionary. The templates for the features we used are summarized in Table 3.4. For example, the features for the character boundary between "り" and "傷" of a example string "やすり | 傷を" are described below. The features of character n-grams are { り |, | 傷, すり |, り | 傷, | 傷を, すり | 傷, り | 傷を }. The features of character type n-grams are { H|, |K,HH|,H|K,|KH,HH|K,H|KH } where "|" denotes an auxiliary symbol that specifies the focused character boundary and "H" and "K" denote Hiragana and Kanji, respectively. In addition, let suppose "やすり" and "傷" in a dictionary. Then, the dictionary feature of c_{+1} have a value because of "| 傷" occurrence and the dictionary feature starts at c_{-1} have a value because of "やすり |" occurrence.

To reduce the number of parameters, we selected only frequent features in the source domain data (A) or in about $50K$ of the unsegmented sentences of the

[8]Ver. 1.3.5; http://www.tokuteicorpus.jp/dist/

[9]Ver. 2.63; http://www2.medis.or.jp/stdcd/byomei/

Table 3.5. The word segmentation performance without domain adaptation.

domain	data set	F
source	B	96.92
target	C	92.30

target domain.[10] The total number of distinct features was about $300K$.

In the preliminary experiment, a CRF that was trained using only the source domain corpus (A), CRF^{S}, achieved F=96.84 in the source domain validation data (B). Note that the hyper-parameter σ of CRF^{S} was selected by data (B). However, it showed the need for the domain adaptation that this CRF^{S} suffered severe performance degradation (F=92.30) on the target domain data. Table 3.5 summarize the result of the preliminary experiment.

We used *conjugate gradient* method to find the local maximum value with the initial value being set to be the parameter vector of CRF^{S}. Since the amount of annotated data for the target domain was limited, $\nu = 1$ was used [11] and σ was the same value of CRF^{S}.

For the comparison with the proposed method, we implemented 1) the CRFs were trained using the most probable label sequences consistent with $\boldsymbol{L}$ (*argmax*) and 2) a *point-wise classifier* which independently learns/classifies each character boundary. For argmax, the most probable label sequences were predicted by the CRF^{S}. As the point-wise classifier, we implemented a logistic regression (maximum entropy classifier) which models a conditional probability $p(y_t|x_t)$. For the logistic regression, we uses the same features and optimizer as CRFs. The performance of a point-wise classifier which is trained using data (A), denoted as point-wise $\text{classifier}^{\text{S}}$, was $F = 91.29$ where the hyper-parameter of point-wise $\text{classifier}^{\text{S}}$ was tuned using data (B) in the same manner as the hyper-parameter selection of CRFs. This result suggests the advantage of the word segmentation system of CRFs over that of point-wise classifier before domain

[10] The data (B) and (C), which were used for validation and test, were excluded from this feature selection process.

[11] Jiang and Zhai [32] reported $\nu > \frac{N}{M}$ was suitable for improving the performance of target domains in some domain adaptation experiments of NLP. However, we still need annotated data for the target domain to select the appropriate value for ν.

adaptation.

This experiment was designed for the case in which a user selects the occurrences of words in the word list using the KWIC interface described in Section 1. We employed JSDCM as a word list in which 224 distinct terms appeared on average over 2 test sets (C1). The number of word annotations varied from 100 to 1000 in this experiment. We prioritized the occurrences of each word in the list using a selective sampling technique. We used label entropy [2],

$$H(\boldsymbol{y}_t^s) = -\sum_{\boldsymbol{y}_t^s \in \boldsymbol{Y}_t^s} p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{y}_t^s|\boldsymbol{x}) \ln p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{y}_t^s|\boldsymbol{x}),$$

as importance metric of each word occurrence, where $\tilde{\boldsymbol{\theta}}$ is the model parameter of CRF$^{\mathrm{S}}$, and $\boldsymbol{y}_t^s = (y_t, y_{t+1}, \cdots, y_s) \in \boldsymbol{Y}_t^s$ is a subsequence starting at t and ending at s in $\boldsymbol{y}$.[12] Intuitively, this metric represents the prediction confidence of CRF$^{\mathrm{S}}$, and the high entropy of $\boldsymbol{y}_t^s$ means that $\boldsymbol{y}_t^s$ occurs in the difficult context for CRF$^{\mathrm{S}}$ to determine the label subsequence.[13] As training data, we mixed the complete annotations (A) and these partial annotations on data (C1) because the performance was better than using only the partial annotations.

6.1 Performances varying the number of partial annotations

First, Figure 3.5 shows the performance comparisons varying the number of word annotations. Note that the F score at 0 of the number of word annotations describes the performance of CRF$^{\mathrm{S}}$ and point-wise classifier$^{\mathrm{S}}$. The performance of all the methods adding partial annotations was improved from that of the

[12]We selected word occurrences in a batch mode since each training of the CRFs takes too much time for interactive use.

[13]Since the average entropy of the occurrences of words in the domain-specific word list was higher than that of the occurrences of words not in the word list, the use of domain-specific word list for annotations can select the important position in the point of entropy.

On the other hand, All the possible subsequence in target domain sentences can be candidates for annotations and we may select the most informative one from them. However, although the longer subsequence requires the larger cost for annotations, the entropy tends to be higher for the longer subsequence. Therefore it is still open question to determine the balance between informative score and annotation cost.

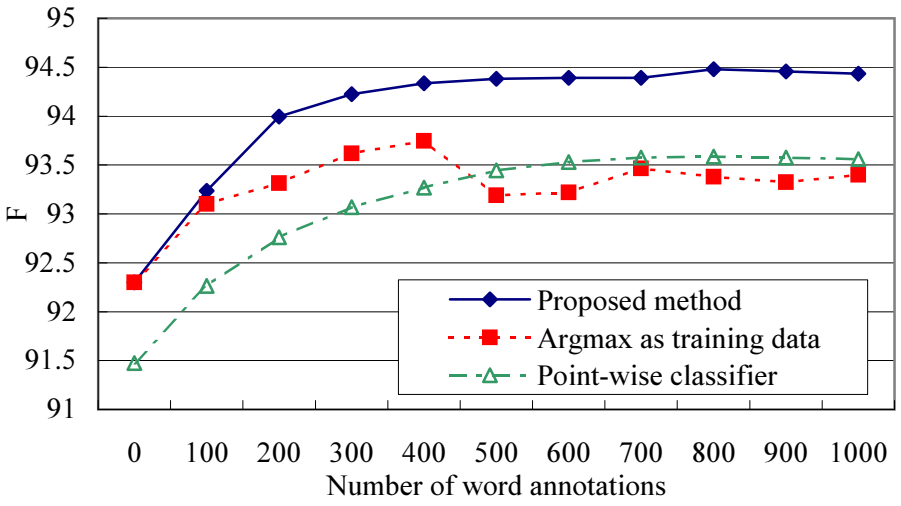


Figure 3.5. Average performances varying the number of word annotations over 2 trials.

models of the source domain. Since 1,000 word annotations is only 7% of total word occurrences in data (C1), the cost of these partial annotations is lower than that of complete annotations. However, the performance of argmax was not stably improved as the number of word annotations increased. This result is consistent with the discussion of argmax in Section 5. On the other hand, the proposed method significantly outperformed *point-wise classifier* (and *argmax*) based on the *Wilcoxon signed rank test* at the significance level of 5%. This result suggests that the proposed method maintains CRFs' advantage over the *point-wise classifier* and properly incorporates partial annotations.

6.2 Performances of prioritized annotations

Second, we conducted the experiment comparing 1) the prioritized annotations by entropy and 2) annotations in the order of data. Note that we used the same value for hyper-parameter and initial parameter value as CRF^{S}.

Figure 3.6 shows the performance comparisons varying the number of word annotations. For the smaller number of word annotations (100 to 500), we observed prioritized annotations improve the performance of JWS. The differences

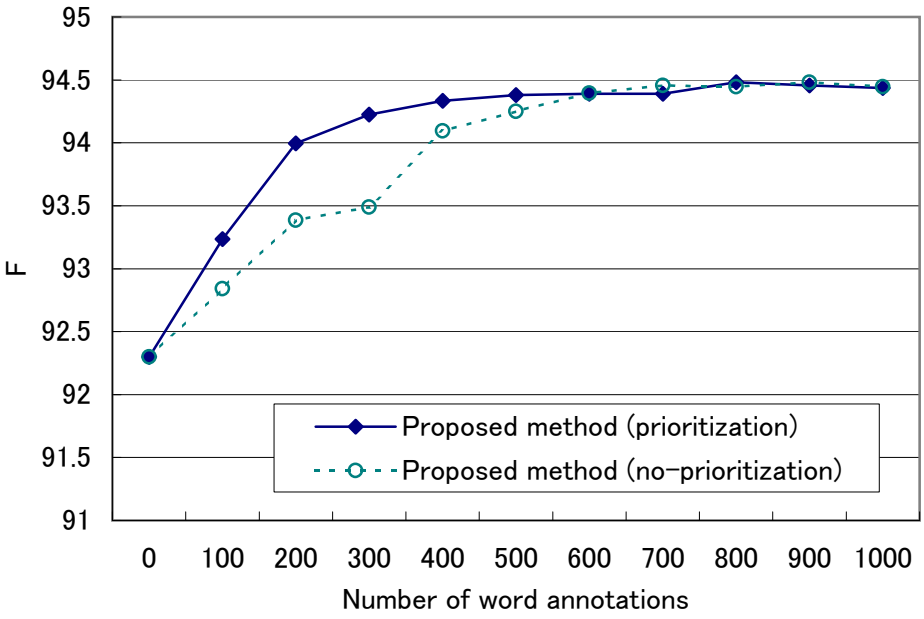


Figure 3.6. Average performances of annotation with and without prioritization over 2 trials

between two result is significant based on the *Wilcoxon signed rank test* at the significance level of 5%.

The proposed method enables to learn CRFs using partial annotations selected by selective sampling method, so that the combination of them achieved the improvement of JWS using a few word annotations.

6.3 Performances varying initial parameter value

We also investigated the performances of JWS when we use the different value of initial parameter value for the proposed method.

Figure 3.7 shows the JWS performances when we initialized the parameter as zero ("init=0") and initialized the parameter as the value of CRF^{S} ("init=CRF-S"). Since the differences between two result is not significant based on the *Wilcoxon signed rank test* at the significance level of 5%, the proposed method is not sensitive for the choice of initial parameter.

In summary, the combination of both the proposed method and the selective

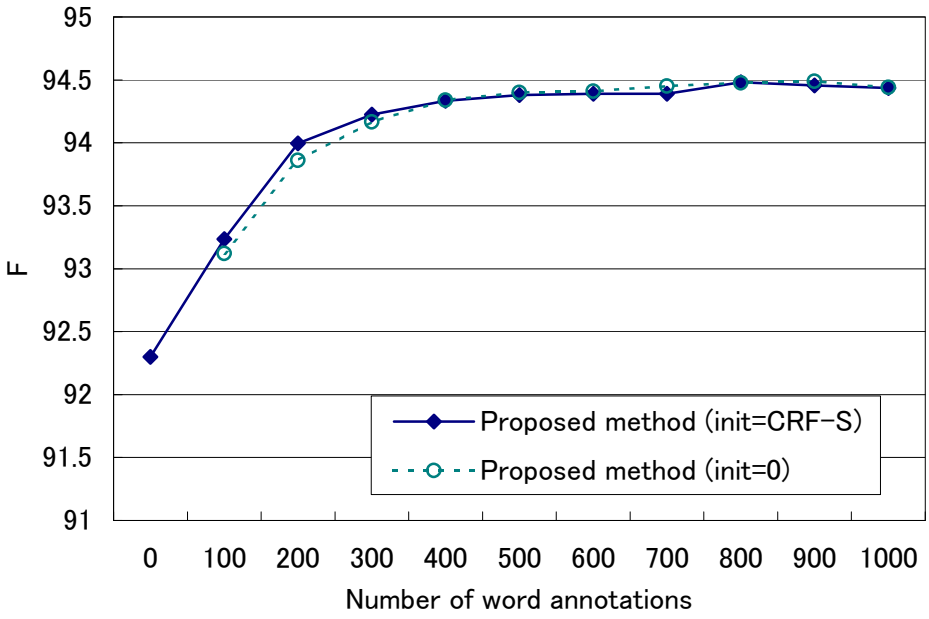


Figure 3.7. Average performances with parameter initialized as zero and CRF^{S} over 2 trials

sampling method showed that a small number of word annotations effectively improved the word segmentation performance.

7. Related Work

One of the research issue of word segmentation is unknown word handling. The main topics of unknown word handling are 1) statistical models of unknown words and 2) lexical acquisition [43]. Generative N-gram modeling requires the estimation of not only word N-gram distribution but also character (type) distribution for unknown word handling. On the other hand, discriminative models are interoperable with various features for unknown words and do not require extra unknown word models.

However, for both generative and discriminative model, lexical acquisition is important issue, especially in the domain adaptation situation. We assumed a domain specific word list is available in the experiment of Section 6. However, such

a word list is sometimes not enough to cover all the unknown words in the target domain data. In this line of work, Peng et al. [45] proposed an unknown word detection method in which the probability of CRFs is used to find unknown word candidates. However, they also reported that the usage of these word candidates can cause the performance degradation of word segmentation. Instead of the blind use of these candidates, if the occurrences of these candidates are checked manually, the relatively steady progress of statistical models will be achieved as seen in Section 6.

In a parsing context, Pereira et al. [46] proposed a grammar acquisition method for partially bracketed corpus. Their work can be considered a generative model for the tree structure output problem using partial annotations. Our discriminative model can be extended to such parsing tasks.

Our model is interpreted as one of the CRFs with hidden variables (HCRFs) [48], in which the probability of a target variable $\boldsymbol{t}$ given $\boldsymbol{x}$ is defined as $p_{\boldsymbol{\theta}}(\boldsymbol{t}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} p_{\boldsymbol{\theta}}(\boldsymbol{t}, \boldsymbol{h}|\boldsymbol{x})$ with hidden variables $\boldsymbol{h} \in \boldsymbol{H}$. Let the target variable and the set of hidden variables be specified by $\boldsymbol{t} = \boldsymbol{Y_L}$ and $\boldsymbol{H} = \boldsymbol{Y_L}$. Then, we can derive the Equation (3.3) from the HCRF model:

$$
\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{Y_L}|\boldsymbol{x}) &= \sum_{\boldsymbol{h} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta}}(\boldsymbol{Y_L}, \boldsymbol{h}|\boldsymbol{x}) \\
&= \sum_{\boldsymbol{h} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{Y_L}, \boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{Y_L}|\boldsymbol{x}) \\
&= \sum_{\boldsymbol{h} \in \boldsymbol{Y_L}} \frac{e^{\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle}}{Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y_L}}} \frac{Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y_L}}}{Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y}}} \\
&= \sum_{\boldsymbol{h} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{x}),
\end{aligned}
$$

referring to the definitions in Equations (3.1) and (3.6). There are previous work which handles hidden variables in discriminative parsers [18, 47]. In their methods, the objective functions are also formulated as same as Equation (3.4).

For interactive annotation, Culotta et al. [21] proposed corrective feedback that effectively reduces user operations utilizing partial annotations. Although they assume that the users correct entire label structures so that the CRFs are trained as usual, our proposed method extends their system when the users cannot annotate all of the labels in a sentence.

One of the related research fields is *semi-supervised learning*, which deals with labeled and unlabeled training instances. The main difference is that a training instance is composed of both labeled and unlabeled parts in our problem setting. Semi-supervised learning for CRF (SSL-CRF) has been proposed [33, 39], and this could be extended to learn an SSL-CRF using partial annotations. According to the semi-supervised CRF framework, the modified objective function for incomplete annotations would be

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \boldsymbol{Y}_{\boldsymbol{L}^{(n)}}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(n)}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(n)}).$$

Its objective function is also not concave and the derivative of the function is more complex than for our proposed method. Our preliminary experimental results show that this extended SSL-CRF for incomplete annotations would be slower than our proposed method, though the performance is comparable.

There have been previous works on active learning for structured output tasks [3, 52, 1, 50]. However, the focus is on sample selections and all of them assume that each local element of structures is independently learnable. On the other hand, the proposed method achieves the combination between selective sampling of substructures and structured output learning with the weaker assumption (the Markov assumption).

8. Summary

In this chapter, we address word-boundary annotation which is done only on part of sentences. By limiting our focus on crucial part of sentences, we can effectively create a training data for each new target domain by conducting such partial annotations. We propose a training algorithm for Conditional Random Fields (CRFs) using partial annotations. It is known that CRFs are well-suited to word segmentation tasks and many other sequence labeling problems in NLP. However, conventional CRF learning algorithms require fully annotated sentences. The objective function of the proposed method is a marginal likelihood function, so that the CRF model incorporates such partial annotations. Through experiments, we show our method effectively utilize partial annotations on a domain adaptation task of Japanese word segmentation.

Chapter 4

Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation

1. Covariate Shift Adaptation

An assumption that is commonly imposed—either explicitly or implicitly—in virtually all supervised learning methods is that the training and test samples follow the *same* probability distribution. However, this fundamental assumption is often violated in practice, causing standard machine learning methods not to work as expected. In this section, we address supervised learning problems in the absence of this fundamental assumption.

If the training and test distributions share nothing in common, we may not be able to learn anything about the test distribution from the training samples. For a meaningful discussion, the training and test distributions should be related to each other in some sense. A situation where the input distribution $p(\boldsymbol{x})$ is different in the training and test phases but the conditional distribution of output values, $p(y|\boldsymbol{x})$, remains unchanged is called *covariate shift* [56]. In many real-world applications such as robot control [64, 54, 30], bioinformatics [4, 13], spam filtering [8], natural language processing [32], brain-computer interfacing [73, 60], or econometrics [29], covariate shift is likely. Covariate shift is also naturally induced in selective sampling or active learning scenarios [24, 19, 72, 34, 59]. For

this reason, learning under covariate shift is receiving a lot of attention these days in the machine learning community (such as in the NIPS2006 workshop [15] and the ECML2006 workshop [7]).

Under covariate shift, standard learning methods such as maximum likelihood estimation are no longer *consistent*, i.e., they do not produce the optimal solution even when the number of training samples tends to be infinity. Thus, there exists an estimation bias induced by covariate shift. It has been shown that the bias can be asymptotically canceled by weighting the log likelihood terms according to the *importance* [25, 56, 74]:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})},$$

where $p_{\mathrm{te}}(\boldsymbol{x})$ and $p_{\mathrm{tr}}(\boldsymbol{x})$ are the test and training input densities. Since the importance is usually unknown in reality, the central issue of practical covariate shift adaptation is how to accurately estimate the importance[1].

A naive approach to importance estimation is to first estimate the training and test densities separately from the training and test input samples, and then estimate the importance by taking the ratio of the estimated densities. However, density estimation is known to be a hard problem particularly in high dimensional cases [26]. Therefore, this naive approach is usually ineffective—directly estimating the importance *without* estimating the densities is more promising. Therefore, several methods that allow us to directly obtain importance estimates without going through density estimation have been proposed recently, such as *kernel mean matching* (KMM) [31], the *logistic regression* based method (Lo-

[1]Covariate shift matters in parameter learning only when the model used for function learning is *misspecified* (i.e., the model is so simple that the true learning target function cannot be expressed) [56]. When the model is correctly (or overly) specified, the ordinary maximum likelihood estimation is still consistent. On this basis, there is a criticism that importance weighting is not needed, but just the use of a sufficiently complex model can settle the problem. However, overly complex models result in large estimation variances, and so in practice we need to choose a complex enough but not overly complex model. To choose such an appropriate model, we usually use a model selection technique such as cross-validation (CV). However, the ordinary CV score is biased due to covariate shift and we still need to importance-weight the CV score (or any other model selection criteria) for unbiasedness [56, 74, 61, 60]. For this reason, estimating the importance is indispensable when covariate shift occurs.

gReg) [10], and the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [62].

KMM is based on a special property of *universal reproducing kernel Hilbert spaces* (Gaussian reproducing kernel Hilbert spaces are typical examples) [58], and KMM allows us to directly obtain the importance estimates at the training input points. Since the KMM optimization problem is formulated as a convex quadratic programming problem, it always leads to the unique global solution. KMM has been shown to work well, as long as the kernel parameters such as the Gaussian width are chosen appropriately. However, to the best of our knowledge, there is no reliable method to determine the Gaussian width and the regularization parameter in the KMM algorithm[2]. Therefore, the lack of model selection procedures is a critical limitation of KMM in practical applications.

LogReg builds a probabilistic classifier that separates training input samples from test input samples, and the importance can be directly estimated by LogReg. The maximum likelihood estimation of the LogReg can be formulated as a convex optimization problem, so the unique global optimal solution can be obtained. In addition, since LogReg only solves a standard supervised classification problem, the tuning parameters such as the kernel width and the regularization parameter can be optimized by the standard cross-validation (CV) procedure. This is a very useful property in practice.

KLIEP tries to match an importance-based estimation of the test input distribution to the true test input distribution in terms of the Kullback-Leibler divergence. KLIEP solves this matching problem in a non-parametric fashion. The training and test input distributions are not parameterized, but only the importance is parameterized. The KLIEP optimization problem is convex and

[2]Intuitively, it seems possible to optimize the kernel width and the regularization parameter simply by using CV for the performance of subsequent learning algorithms. However, this is highly unreliable since the ordinary CV score is biased under covariate shift. For unbiased estimation of the prediction performance of subsequent learning algorithms, the CV procedure itself needs to be importance-weighted [74, 60]. Since the importance weight has to have been fixed when model selection is carried out using the importance weighted CV, it cannot be used for model selection of importance estimation algorithms. Note that once the importance weight has been fixed, the importance-weighted CV can be used for model selection of subsequent learning algorithms.

therefore a unique global optimal solution can be obtained. Furthermore, the global solution tends to be sparse, so it is computationally efficient in the test phase. Since KLIEP is based on the minimization of the Kullback-Leibler divergence, the model selection of KLIEP, such as the choice of the kernel width and the regularization parameter, can be carried out naturally through the *likelihood CV* procedure [26], so no open tuning parameter remains.

As reviewed above, LogReg and KLIEP seem to have advantages over KMM, since they are equipped with built-in model selection procedures. On the other hand, from the viewpoint of scalability, all three of the methods have limitations—in recent applications such as spam filtering [8] and information retrieval [28], the number of test (unlabeled) samples is enormous, especially on the Web. In these text processing applications, the distribution of training and test inputs can be changed between domains because of differences in vocabulary and writing style. The purpose of this chapter is to develop a computationally efficient covariate shift adaptation method that can deal with a large number of unlabeled data points.

Our new method is primarily based on KLIEP. The key difference is that the original KLIEP uses a linearly parameterized function for modeling the importance, while we adopt a *log-linear* model. By definition, the log-linear model only takes non-negative values. This allows us to reformulate the KLIEP optimization problem as an *unconstrained* convex problem. Then we develop a new scalable estimation procedure whose computation time is nearly independent of the number of test samples. More precisely, we need to scan a large number of test samples only once to compute a summary statistic in the beginning (this precomputation can be carried out in linear time and constant storage space). The main optimization procedure does not use the test samples themselves, but only uses the summary statistic. Therefore, the computation time of the main optimization procedure is independent of the number of test samples.

The experiments show that the proposed method is computationally much more efficient than the existing approaches. Therefore the range of application of covariate shift adaptation can be greatly enlarged towards large-scale problems. As for estimation accuracy, we experimentally show that the performance of the proposed method is comparable to the best existing methods for small and mid-

dle sized problems (since the existing methods cannot be applied to large-scale problems due to the computational costs). Thus the proposed method can be a useful alternative to the existing covariate shift adaptation methods.

2. Problem Formalization

In this section, we formulate the supervised learning problem under covariate shift and briefly review existing techniques for covariate shift adaptation.

2.1 Supervised learning under covariate shift

Let $\boldsymbol{x} \in \boldsymbol{X} \subset \Re^d$ be an input variable and $y \in Y$ be an output variable. Y is a real space in regression cases or a set of categories in classification cases. In standard supervised learning frameworks, it is assumed that $\boldsymbol{x}$ is independently drawn from an input distribution and y is independently drawn from a conditional distribution, both in training and test phases. In contrast, here we consider a situation called *covariate shift* [56], i.e., the input distribution differs in the training and test phases, but the conditional distribution remains unchanged.

Suppose we have independent and identically distributed (i.i.d.) training input samples $D_{\mathrm{tr}} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ from a distribution with strictly positive density $p_{\mathrm{tr}}(\boldsymbol{x})$, and test input samples $D_{\mathrm{te}} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\mathrm{te}}}$ from a distribution with density $p_{\mathrm{te}}(\boldsymbol{x})$. In addition to the input samples, suppose we have training output samples $\{y^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ drawn from the conditional distribution with conditional density $p(y|\boldsymbol{x} = \boldsymbol{x}^{(i)})$, respectively. Typically, the number N_{tr} of training samples is rather small due to the high labeling cost, while the number N_{te} of test input samples is very large since they are often easily available. We denote training sample pairs of input and output as:

$$Z_{\mathrm{tr}} = \{z^{(i)} \mid z^{(i)} = (\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N_{\mathrm{tr}}}.$$

We use the following linear model:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle \,, \tag{4.1}$$

where $\boldsymbol{\theta}$ is the parameter vector, $\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{X} \to \Re^h$ is a basis function of $\boldsymbol{x}$, and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denotes the Euclidean inner product between vector $\boldsymbol{u}$ and $\boldsymbol{v}$: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle =$

$\sum_{l=1}^{h} u_l v_l$. Note that this model can contain a bias parameter by just including a constant basis function in $\boldsymbol{\phi}(\boldsymbol{x})$. Throughout the section, we suppose that this linear model is not generally specified correctly, i.e., the true input-output function is not necessarily included in the above linear model. Since we do not know the true function class in practice, dealing with misspecified models is quite realistic.

The goal of supervised learning is to learn the parameter $\boldsymbol{\theta}$ so that the output values for the test inputs can be accurately predicted. Thus our error metric (which is usually called the *generalization error*) is given by

$$\iint \mathrm{Loss}(\boldsymbol{x}, y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) p_{\mathrm{te}}(\boldsymbol{x}) p(y|\boldsymbol{x}) d\boldsymbol{x} dy, \tag{4.2}$$

where $\mathrm{Loss}(\boldsymbol{x}, y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) : \boldsymbol{X} \times Y \times Y \to \Re$ is a loss function, such as the squared loss in a regression case or the zero-one loss in a classification case.

In supervised learning under covariate shift, the following quantity called the *test domain importance* plays an important role:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}. \tag{4.3}$$

The importance can be used for adjusting the difference between the training and test input distributions: for any function $A(\boldsymbol{x})$,

$$\int A(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int A(\boldsymbol{x}) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{4.4}$$

2.2 Parameter learning under covariate shift

Here we review two typical parameter learning methods under covariate shift: one is *importance weighted least squares* (IWLS) for regression and the other is *importance weighted logistic regression* (IWLR) for classification.

IWLS A standard learning method in regression scenarios would be ordinary least squares (LS):

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x}, y) \in Z_{\mathrm{tr}}} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y \right)^2 \right].$$

LS is known to be consistent under a usual setting. However, it is no longer consistent for misspecified models under covariate shift. Instead, IWLS is consistent [56]:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\right)^2 + \lambda \|\boldsymbol{\theta}\|^2 \right], \tag{4.5}$$

where the importance $w(\boldsymbol{x})$ is used as weights. Here we also added a penalty term $\lambda\|\boldsymbol{\theta}\|^2$ for regularization, where λ is a regularization parameter.

For the linear model (4.1), the above optimization problem is convex and the unique global solution $\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}}$ can be computed in a closed-form as

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} = (\boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{y},$$

where $\boldsymbol{I}$ is the identity matrix,

$$\Phi_{i,l} = \phi_l(\boldsymbol{x}^{(i)}), \quad \boldsymbol{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(N_{\mathrm{tr}})})^\top, \text{ and}$$
$$\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(N_{\mathrm{tr}})}).$$

IWLR For simplicity, we focus on the two-class case, i.e., $Y = \{-1, 1\}$; we note that it is straightforward to extend all of the discussions in this section to multi-class cases.

Let us model the posterior probability of class y given $\boldsymbol{x}$ using a parametric model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ as

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{1 + \exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}. \tag{4.6}$$

Then a test input sample $\boldsymbol{x}$ is classified by choosing the most probable class:

$$\hat{y} = \operatorname*{argmax}_{y} p_{\boldsymbol{\theta}}(y|\boldsymbol{x}). \tag{4.7}$$

A standard learning method for the above probabilistic classification scenarios would be ordinary logistic regression (LR):

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\mathrm{LR}} &\equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} -\ln\, p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \right] \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} \left(\ln\left(1 + \exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \right].\end{aligned}$$

Similar to the case of LS, LR is consistent under a usual setting, but is no longer consistent for misspecified models under covariate shift. Instead, IWLR is consistent:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLR}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\ln\left(1+\exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) + \lambda\|\boldsymbol{\theta}\|^2 \right]. \quad (4.8)$$

Here we also added a penalty term $\lambda\|\boldsymbol{\theta}\|^2$ for regularization, where λ is a regularization parameter.

This optimization problem is known to be convex and a unique optimal solution can be computed using standard non-linear optimization techniques such as a gradient descent method or some variants of the Newton method. The gradient of the above objective function is given by

$$\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(y\, p_\theta(y|\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) - y\,\boldsymbol{\phi}(\boldsymbol{x})\right) + 2\lambda\boldsymbol{\theta}.$$

2.3 Model selection under covariate shift

In the above learning methods, the choice of model parameters such as the basis functions $\boldsymbol{\phi}$ and the regularization parameter λ heavily affects the prediction performance. This problem is called model selection and is one of the key concerns in machine learning.

A popular model selection method in the machine learning community would be cross-validation (CV). The performance of CV is guaranteed in the sense that it gives an unbiased estimate of the generalization error. However, this useful theoretical property is no longer true under covariate shift [74]. To cope with this problem, a variant of CV called *importance weighted CV* (IWCV) has been proposed for model selection under covariate shift [60]. It has been proved that IWCV gives an unbiased estimate of the generalization error even under covariate shift.

Here, let us briefly describe the IWCV procedure. We first divide the training samples $\{z^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ into R disjoint subsets $\{Z^r\}_{r=1}^{R}$. Then we learn a function $f_{\boldsymbol{\theta}}^r(\boldsymbol{x})$ from $\{Z^j\}_{j\neq r}$ by IWLS/IWLR and compute its mean test error for the remaining

samples Z^r:

$$\frac{1}{|Z^r|}\sum_{(\boldsymbol{x},y)\in Z^r} w(\boldsymbol{x})\left(f^r_{\boldsymbol{\theta}}(\boldsymbol{x})-y\right)^2, \quad \text{(regression)}$$

$$\frac{1}{|Z^r|}\sum_{(\boldsymbol{x},y)\in Z^r} w(\boldsymbol{x})I(\hat{y}=y), \quad \text{(classification)}$$

where $I(\cdot)$ denotes the indicator function. We repeat this procedure for $r = 1, 2, \ldots, R$ and choose the model such that the average of the above mean test error is minimized.

2.4 Importance estimation

As we have seen in the previous section, the importance $w(\boldsymbol{x})$ plays a central role in covariate shift adaptation. However, the importance is unknown in practice so we need to estimate it from samples.

Direct importance estimation methods that do not involve density estimation steps have been developed recently [31, 10, 62]. Here we review one of those direct methods called the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [62]. Other methods will be reviewed in Section 5.

2.5 KLIEP

Let us model $w(\boldsymbol{x})$ with the following linear model:

$$\hat{w}(\boldsymbol{x}) = \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle, \tag{4.9}$$

where $\boldsymbol{\alpha} \in \Re^b$ is a model parameter vector and $\boldsymbol{\psi}(\boldsymbol{x}) \in \Re^b$ is a basis function. Since the importance should be non-negative by definition, we suppose that both $\boldsymbol{\alpha}$ and $\boldsymbol{\psi}(\boldsymbol{x})$ are non-negative.

Using the importance estimation $\hat{w}(\boldsymbol{x})$, we can estimate the test input density $p_{\mathrm{te}}(\boldsymbol{x})$ by

$$\hat{p}_{\mathrm{te}}(\boldsymbol{x}) = p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x}). \tag{4.10}$$

Now we learn the parameter $\boldsymbol{\alpha}$ so that the Kullback-Leibler divergence from

$p_{\mathrm{te}}(\boldsymbol{x})$ to $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ is minimized:

$$\begin{aligned}\mathrm{KL}[p_{\mathrm{te}}(\boldsymbol{x})||\hat{p}_{\mathrm{te}}(\boldsymbol{x})] &= \int_D p_{\mathrm{te}}(\boldsymbol{x}) \ln \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \int_D p_{\mathrm{te}}(\boldsymbol{x}) \ln \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} d\boldsymbol{x} - \int_D p_{\mathrm{te}}(\boldsymbol{x}) \ln\, \hat{w}(\boldsymbol{x}) d\boldsymbol{x}. \end{aligned} \tag{4.11}$$

Since the first term in Equation (4.11) is independent of $\boldsymbol{\alpha}$, we ignore it and focus on the second term, which we denote by J_{KLIEP}:

$$J_{\mathrm{KLIEP}} = \int_D p_{\mathrm{te}}(\boldsymbol{x}) \ln\, \hat{w}(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \ln\, \hat{w}(\boldsymbol{x}), \tag{4.12}$$

where an empirical approximation based on the test input samples is used. This is the objective function to be maximized. The value of $\hat{w}(\boldsymbol{x})$ should be properly normalized since it is a probability density function:

$$1 = \int_D \hat{p}_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int_D p_{\mathrm{tr}}(\boldsymbol{x}) \hat{w}(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}(\boldsymbol{x}), \tag{4.13}$$

where the empirical approximation based on the training samples is used.

Then the resulting optimization problem is expressed as

$$\underset{\boldsymbol{\alpha}}{\mathrm{maximize}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \ln \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle \ \text{subject to} \ \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle \ = \ N_{\mathrm{tr}} \ \text{and} \ \boldsymbol{\alpha} \ \geq \ \mathbf{0},$$

which is convex. Thus the global solution can be obtained by iteratively performing gradient ascent and feasibility satisfaction.

2.6 Model selection by likelihood CV

The performance of KLIEP depends on the choice of the basis functions $\psi(\boldsymbol{x})$ (and possibly an additional regularization parameter). Since KLIEP is based on the maximization of the score J_{KLIEP}, it would be natural to select the model such that J_{KLIEP} is maximized. The expectation over $p_{\mathrm{te}}(\boldsymbol{x})$ involved in J_{KLIEP} can be numerically approximated by *likelihood CV* (LCV) [26] as follows: First, divide the test samples D_{te} into R disjoint subsets $\{D_{\mathrm{te}}^r\}_{r=1}^R$. Then, obtain an importance estimate $\hat{w}^r(\boldsymbol{x})$ from $\{D_{\mathrm{te}}^t\}_{t \neq r}^R$ and approximate the score J_{KLIEP} using D_{te}^r as

$$\hat{J}_{\mathrm{KLIEP}}^r = \frac{1}{|D_{\mathrm{te}}^r|} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}^r} \hat{w}^r(\boldsymbol{x}). \tag{4.14}$$

This procedure is repeated for $r = 1, 2, \ldots, R$ for each model and choose the model such that the average of $\hat{J}^r_{\mathrm{KLIEP}}$ for all r is maximized.

One of the potential general limitations of CV is that it is not reliable in small sample cases, since data splitting by CV further reduces the sample size. A key advantage of the LCV procedure is that, not the training samples, but the test input samples are cross-validated. This contributes greatly to improving the model selection accuracy, since the number of training samples is typically limited while there are lots of test input samples available.

As basis functions, it is suggested to use Gaussian kernels centered at a subset of the test input points D_{te} [62]:

$$K_s(\boldsymbol{x}, \boldsymbol{x}_l) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{x}_l\|^2}{2s^2}\right\}, \tag{4.15}$$

where $\boldsymbol{x}_l \in D_{\mathrm{te}}$ is a template test sample and s is the kernel width. This is a heuristic to allocate many kernels at high test input density regions since many kernels may be needed in the region where the output of the target function is large. In the original paper, the number of Gaussian centers was fixed at $N_{\mathrm{te}}/10$ for computational efficiency and the kernel width s was chosen by LCV.

3. KLIEP for Log-linear Models

As shown above, KLIEP has its own model selection procedure and has been shown to work well in importance estimation [62]. However, it has a weakness in computation time. In each step of gradient ascent, the summation over all test input samples needs to be computed, which is prohibitively slow in large-scale problems. The main contribution in this section is to extend KLIEP so that it can deal with large sets of test input data.

3.1 LL-KLIEP

In the original KLIEP, a linearly parameterized model (4.9) is used for modeling the importance function. Here, we propose using a (normalized) log-linear model for modeling the importance $w(\boldsymbol{x})$ as

$$\hat{w}(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)}, \tag{4.16}$$

where the denominator guarantees the normalization constraint (4.13) [3]. By definition, the log-linear model takes only non-negative values. Therefore, we no longer need the non-negative constraint for the parameter (and the basis functions).

Then the optimization problem becomes *unconstrained*:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}}\, J_{\text{LL-KLIEP}}(\boldsymbol{\alpha}),$$

where

$$\begin{aligned} J_{\text{LL-KLIEP}}(\boldsymbol{\alpha}) &= \frac{1}{N_{\text{te}}} \sum_{\boldsymbol{x} \in D_{\text{te}}} \ln\, \hat{w}(\boldsymbol{x}) \\ &= \frac{1}{N_{\text{te}}} \sum_{\boldsymbol{x} \in D_{\text{te}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \ln \frac{1}{N_{\text{tr}}} \sum_{\boldsymbol{x} \in D_{\text{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle). \end{aligned} \tag{4.17}$$

Below, we refer to this method as *LL-KLIEP* (log-linear KLIEP). In practice, we may add a penalty term for regularization:

$$\jmath(\boldsymbol{\alpha}) = J_{\text{LL-KLIEP}}(\boldsymbol{\alpha}) - \frac{||\boldsymbol{\alpha}||^2}{2\varsigma^2}, \tag{4.18}$$

where ς^2 is a regularization parameter.

An advantage of LL-KLIEP over the original KLIEP is its computational efficiency. The gradient of $\jmath(\boldsymbol{\alpha})$ can be computed as

$$\begin{aligned} \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} =& \frac{1}{N_{\text{te}}} \sum_{\boldsymbol{x} \in D_{\text{te}}} \boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\text{tr}}} \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\sum_{\boldsymbol{x}' \in D_{\text{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)} \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\varsigma^2} \\ =& F - \frac{1}{N_{\text{tr}}} \sum_{\boldsymbol{x} \in D_{\text{tr}}} \hat{w}(x) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\varsigma^2}, \end{aligned} \tag{4.19}$$

[3] The log-linear model can have numerical problems since it contains an exponential function. To cope with this problem, we do not directly compute the value of $\hat{w}(\boldsymbol{x})$, but we compute it in the exponential of the logarithmic domain, i.e.,

$$\exp(\ln\, \hat{w}(\boldsymbol{x})) = \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \ln \frac{1}{N_{\text{tr}}} \sum_{\boldsymbol{x} \in D_{\text{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)).$$

To further stabilize the computation, we compute the logarithmic sum of the exponential functions as

$$\ln(\exp(a) + \exp(b)) = \ln(1 + \exp(b - a)),$$

where we pick the smaller exponent as b.

where

$$F = \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \boldsymbol{\psi}(\boldsymbol{x}).$$

This means that once we pre-compute the value of F, we do not need to use the test samples when we compute the gradient. This contributes greatly to reducing the computation time when the number of test samples is large. In addition, we do not need to store all of the test samples in memory since we only need the value of F. The required storage capacity is only $\Omega(cN_{\mathrm{tr}})$, where c is the average number of non-zero basis entries.

As the model selection of KLIEP, LCV can be used to find the optimal hyper-parameters of LL-KLIEP. Since $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ is evaluated using both training and test samples, both test and training samples can be divided into R disjoint subset $\{D^r_{\mathrm{te}}\}_{r=1}^R$ and $\{D^r_{\mathrm{tr}}\}_{r=1}^R$ in the LCV procedure. After the estimation of $\hat{w}^r(\boldsymbol{x})$, $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ can be approximated as

$$\hat{J}^r_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) = \frac{1}{|D^r_{\mathrm{te}}|} \sum_{\boldsymbol{x} \in D^r_{\mathrm{te}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \ln \frac{1}{|D^r_{\mathrm{tr}}|} \sum_{\boldsymbol{x} \in D^r_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle).$$

Although the proposed optimization procedure may be more efficient than original KLIEP, there still exists a potential weakness: we still need to use all the test samples when computing the values of $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ or $\jmath(\boldsymbol{\alpha})$. The value of $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ is needed when we choose a model by LCV, and the value of $\jmath(\boldsymbol{\alpha})$ is often utilized in line search or in the stopping criterion.

3.2 LL-KLIEP(LS)

Here, we introduce another optimization technique for LL-KLIEP that enables us to overcome the above weakness. Our basic idea is to encourage the derivative of the convex objective function to be zero. We use a squared norm to measure the 'magnitude' of the derivative (4.19):

$$\jmath_{\mathrm{LS}}(\boldsymbol{\alpha}) = \frac{1}{2} \left\| \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \right\|^2. \tag{4.20}$$

The partial derivative of Equation (4.20) with respect to $\boldsymbol{\alpha}$ is expressed as

$$\frac{\partial \jmath_{\mathrm{LS}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{\partial^2 \jmath(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}. \tag{4.21}$$

Note that the first component of Equation (4.21) is the Hessian matrix of $\jmath(\boldsymbol{\alpha})$:

$$\frac{\partial^2 \jmath(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} = \left(\sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\overline{\boldsymbol{\psi}}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{x}) \right) \boldsymbol{\psi}(\boldsymbol{x})^T - \frac{I}{\varsigma^2} \right),$$

where

$$\overline{\boldsymbol{\psi}}(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \boldsymbol{\psi}(\boldsymbol{x}),$$

$\boldsymbol{\psi}(\boldsymbol{x})^T$ is the transpose of $\boldsymbol{\psi}(\boldsymbol{x})$, and I is the identity matrix. When we explicitly compute the Hessian matrix of $\jmath(\boldsymbol{\alpha})$, the computational complexity of the derivative is $O(b^2 N_{\mathrm{tr}})$, which is independent of N_{te}. Also, the required storage space is independent of N_{te}: $\Omega(b^2 + cN_{\mathrm{tr}})$. We refer to this approach as *LL-KLIEP(LS1-a)* below.

The computation time and storage space of LL-KLIEP(LS1-a) are quadratic functions of the number of parameters b, which could be a bottleneck in high dimensional problems. To cope with this problem, we propose two approaches.

One approach for high dimensional problems is directly computing the product between the Hessian matrix and the gradient vector of $\jmath(\boldsymbol{\alpha})$ without storing the Hessian matrix:

$$\frac{\partial \jmath_{\mathrm{LS}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \left(\sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\overline{\boldsymbol{\psi}}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{x}) \right) \langle \boldsymbol{\psi}(\boldsymbol{x}), G \rangle - \frac{G}{\varsigma^2} \right), \tag{4.22}$$

where $G = \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$. Since the inner product $\langle \boldsymbol{\psi}(\boldsymbol{x}), G \rangle$ requires $O(b)$ time, we can compute Equation (4.22) in total with $O(bN_{\mathrm{tr}})$ computation time and $\Omega(cN_{\mathrm{tr}})$ space. We refer this approach as *LL-KLIEP(LS1-b)*, which is still independent of N_{te} and suitable for high dimensional problems compared with LL-KLIEP(LS1-a).

In the other approach for high dimensional problems, we make use of the *representer theorem* [71]. Our idea is to represent the parameter $\boldsymbol{\alpha}$ as a linear combination of the input samples:

$$\boldsymbol{\alpha} = \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}) \beta_{\boldsymbol{x}},$$

where $\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}$ is a data-wise parameter. Then Equation (4.20) can be rewritten

as

$$\jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}) = \frac{1}{2}\left\| F - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x})\omega(\boldsymbol{x}) - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} \frac{\boldsymbol{\psi}(\boldsymbol{x})\beta_{\boldsymbol{x}}}{\varsigma^2} \right\|^2, \tag{4.23}$$

where

$$\begin{aligned} \omega(\boldsymbol{x}) &= \frac{\exp(\sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}} K(\boldsymbol{x},\boldsymbol{x}')\beta_{\boldsymbol{x}'})}{\sum_{\boldsymbol{x}''\in D_{\mathrm{tr}}} \exp(\sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}} K(\boldsymbol{x}'',\boldsymbol{x}')\beta_{\boldsymbol{x}'})}, \\ K(\boldsymbol{x},\boldsymbol{x}') &= \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x}') \rangle . \end{aligned} \tag{4.24}$$

The partial derivative of Equation (4.23) with respect to $\beta_{\boldsymbol{x}}$ is:

$$\begin{aligned} &\frac{\partial \jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x}\in D_{\mathrm{tr}}})}{\partial \beta_{\boldsymbol{x}}} = \\ &\left\langle F - \sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}') \left(\omega(\boldsymbol{x}') - \frac{\beta_{\boldsymbol{x}'}}{\varsigma^2} \right), \sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}} \omega(\boldsymbol{x}')\boldsymbol{\psi}(\boldsymbol{x}') \langle \boldsymbol{\varphi}(\boldsymbol{x}'), \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \frac{\boldsymbol{\psi}(\boldsymbol{x})}{\varsigma^2} \right\rangle, \end{aligned} \tag{4.25}$$

where $\boldsymbol{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}} \omega(\boldsymbol{x}')\boldsymbol{\psi}(\boldsymbol{x}') - \boldsymbol{\psi}(\boldsymbol{x})$. By the change of variables, it is not required to calculate the partial derivative with respect to $\boldsymbol{\alpha}$ so that we can avoid the computation of the Hessian matrix of $\jmath(\boldsymbol{\alpha})$. We refer to this approach as *LL-KLIEP(LS2)*.

The computation of LL-KLIEP(LS2) requires $O(bN_{\mathrm{tr}}^2)$ time and $\Omega(N_{\mathrm{tr}}^2 + cN_{\mathrm{tr}})$ space. The computation time is linear with respect to the number of parameters b and the storage space is independent of b. This is also an improvement over the direct computation of the partial derivative in Equation (4.21).

For LL-KLIEP(LS), LCV can also be computed very efficiently. In each validation set using D_{te}^r and D_{tr}^r, we can compute the validation error as

$$\hat{J}^r_{\text{LL-KLIEP(LS)}} = \left\| F^r - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}^r} \hat{w}^r(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}) \right\|^2,$$

where

$$F^r = \frac{1}{|D_{\mathrm{te}}^r|} \sum_{\boldsymbol{x}\in D_{\mathrm{te}}^r} \boldsymbol{\psi}(\boldsymbol{x}).$$

Table 4.1. Computational complexity and space requirements. N_{tr} is the number of training samples, N_{te} is the number of test samples, b is the number of parameters, and c is the average number of non-zero basis entries. "Precomp." denotes the computational complexity of once-off precomputation.

	Computational complexity			Space requirement	
	Precomp.	Objective	Derivative	Objective	Derivative
KLIEP	0	$bN_{\mathrm{tr}}+bN_{\mathrm{te}}$	$bN_{\mathrm{tr}}+bN_{\mathrm{te}}$	$cN_{\mathrm{tr}}+cN_{\mathrm{te}}$	$cN_{\mathrm{tr}}+cN_{\mathrm{te}}$
LL-KLIEP	bN_{te}	$bN_{\mathrm{tr}}+bN_{\mathrm{te}}$	bN_{tr}	$cN_{\mathrm{tr}}+cN_{\mathrm{te}}$	cN_{tr}
LL-KLIEP(LS1-a)	bN_{te}	bN_{tr}	b^2N_{tr}	cN_{tr}	b^2+cN_{tr}
LL-KLIEP(LS1-b)	bN_{te}	bN_{tr}	bN_{tr}	cN_{tr}	cN_{tr}
LL-KLIEP(LS2)	bN_{te}	bN_{tr}^2	bN_{tr}^2	cN_{tr}	$N_{\mathrm{tr}}^2+cN_{\mathrm{tr}}$

Note that, once the mean basis vectors F^r are calculated for all R disjoint subsets of D_{te}, $\hat{J}^r_{\text{LL-KLIEP(LS)}}$ can be evaluated independently of the size of the test data D^r_{te}.

The computational complexity and storage space of each method are summarized in Table 4.1. In terms of the complexity analysis, LL-KLIEP(LS1-b) is the best solution for the large amount of test inputs. We verified the analysis by the computational experiments in Section 6.

4. Illustrative Examples

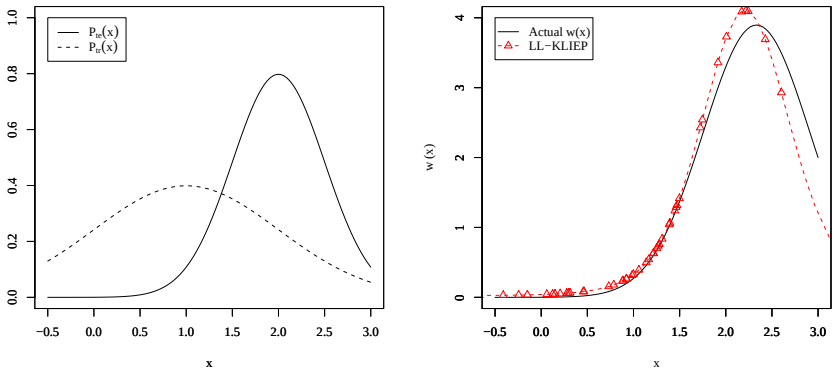
In this section, we illustrate the behavior of the proposed LL-KLIEP and show how it can be applied in covariate shift adaptation.

4.1 Regression under covariate shift

Let us consider an illustrative regression problem of learning

$$f(x) = \mathrm{sinc}(x).$$

Let the training and test input densities be $p_{\mathrm{tr}}(x) = \mathcal{N}(x; 1, 1^2)$ and $p_{\mathrm{te}}(x) = \mathcal{N}(x; 2, 0.5^2)$, where $\mathcal{N}(x; \mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2. We create the training output value $\{y^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ as $y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}$,



(a) Training and test densities (b) Estimated importance

Figure 4.1. Importance estimation.

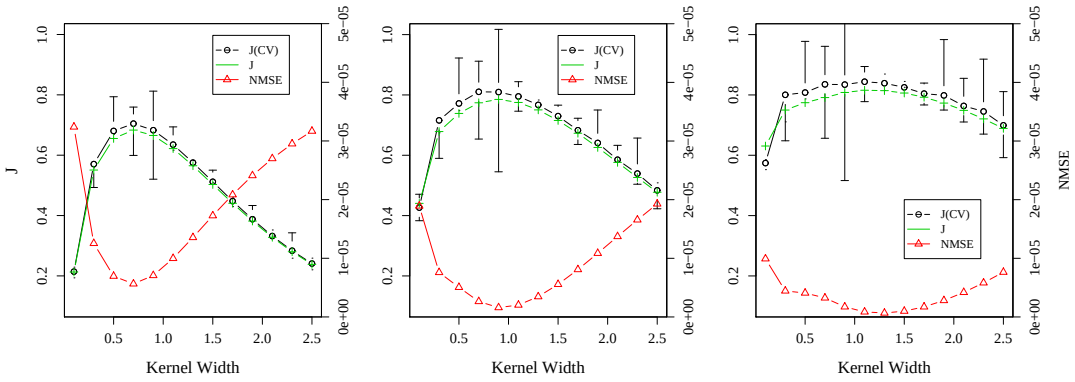
where the noise $\{\epsilon^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ has density $\mathcal{N}(\epsilon; 0, 0.25^2)$. Let the number of training samples be $N_{\mathrm{tr}} = 200$ and the number of test samples be $N_{\mathrm{te}} = 1000$. These settings imply that we are considering an extrapolation problem (see Figure 4.1(a)).

We used 100 Gaussian basis functions centered at randomly chosen test input samples. Figure 4.1(b) shows the actual importance $w(x)$ and an estimated importance $\hat{w}(x)$ by using LL-KLIEP, where the hyper-parameters such as the Gaussian width and the regularization parameter are selected by LCV. We also tested LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2), but we omit their graphs since their solutions are almost identical to the solution of LL-KLIEP.

Figure 4.2 depicts the values of the true $J_{\mathrm{LL-KLIEP}}$ (see Equation (4.17)) and its estimate by 5-fold LCV. The means, the 25 percentiles, and the 75 percentiles over each validation are plotted as functions of the kernel width s for the different $\varsigma = 0.5, 1, 2$. We also plot the normalized mean squared error of the estimated importance:

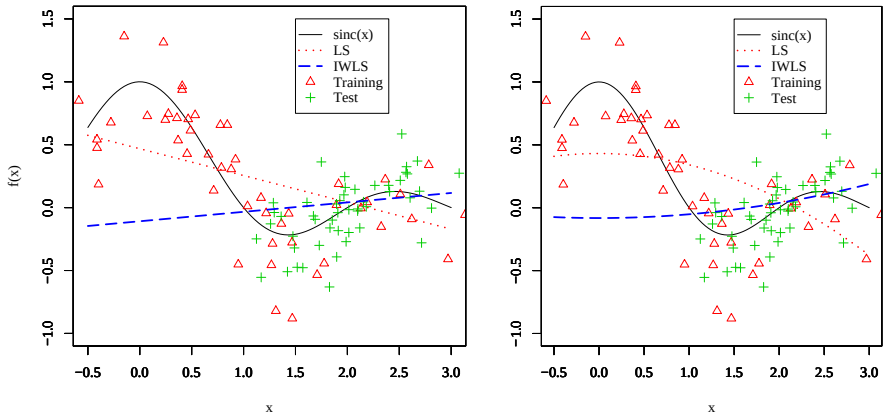
$$\mathrm{NMSE} = \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \left(\frac{\hat{w}(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \hat{w}(\boldsymbol{x}')} - \frac{w(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} w(\boldsymbol{x}')} \right)^2 . \tag{4.26}$$

The graph shows that LCV gives a very good estimate of $J_{\mathrm{LL-KLIEP}}$ and also NMSE. Figure 4.2 also shows that ς value affects the importance estimation in terms of NMSE.



(a) 5CV ($\varsigma = 0.5$) (b) 5CV ($\varsigma = 1$) (c) 5CV ($\varsigma = 2$)

Figure 4.2. Model selection curve.



(a) Polynomials of order 1 (b) Polynomials of order 2

Figure 4.3. Regression under covariate shift (True and learned functions).

Table 4.2. Specifications of illustrative classification data.

		Training $p_{\mathrm{tr}}(\boldsymbol{x}, y)$		Test $p_{\mathrm{te}}(\boldsymbol{x}, y)$	
		$y = 0$	$y = 1$	$y = 0$	$y = 1$
Figure 4.4(a)	μ	(-1,-1)	(3,-1)	(0,3.5)	(4,2.5)
	Σ	$\begin{pmatrix} 0.25 & 0 \\ 0 & 4 \end{pmatrix}$		$\begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	
Figure 4.4(b)	μ	(-1,0)	(4,2)	(0,2)	(3,1)
	Σ	$\begin{pmatrix} 0.75 & 0 \\ 0 & 1.5 \end{pmatrix}$		$\begin{pmatrix} 0.75 & 0.5 \\ 0.01 & 0.1 \end{pmatrix}$	

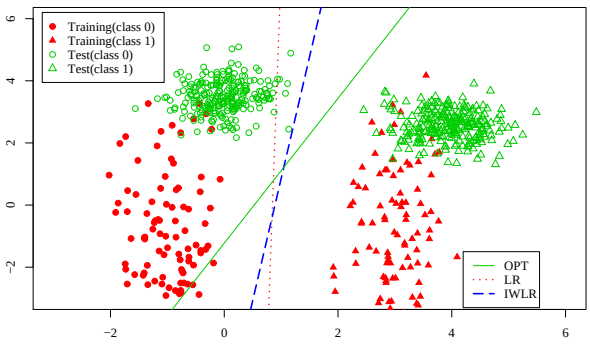
Figure 4.3 shows the true learning target function and functions learned by ordinary LS and IWLS with a linear basis function (Figure 4.3(a)), i.e., $\boldsymbol{\phi}(x) = (1, x)^\top$, and a quadratic basis function (Figure 4.3(b)), i.e., $\boldsymbol{\phi}(x) = (1, x, x^2)^\top$ (Section 2.2). The regularization parameter λ was selected by CV for LS and IWCV for IWLS (Sections 2.3). The results show that the learned function using IWLS goes reasonably well through the test samples, while that of ordinary LS overfits the training samples. Note that the output of the test samples are not used to obtain the learned functions.

4.2 Classification under covariate shift

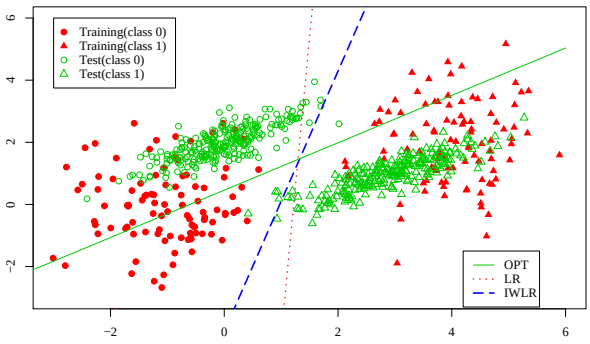
Next, let us consider two illustrative binary classification problems, where two-dimensional samples were generated from Gaussian distributions (see Table 4.2 and Figure 4.4). These data sets correspond to a 'linear shift' and a 'non-linear shift' (rotation).

Let the number of the training samples be $N_{\mathrm{tr}} = 200$ and that of the test samples be $N_{\mathrm{te}} = 1000$ (only 500 test samples are plotted for clarity). We used LR/IWLR for the training classifiers (see Section 2.2), and employed CV/IWCV for the regularization parameter tuning (see Section 2.3). We used a linear basis function for LR/IWLR: $\boldsymbol{\phi}(\boldsymbol{x}) = (1, \boldsymbol{x}^\top)^\top$.

Figure 4.4 shows the decision boundaries obtained by LR+CV and IWLR+IWCV. For references, we also show 'OPT', which is the optimal decision boundary obtained using the test input-output samples. For the data set depicted in



(a) $p_{\mathrm{te}}(\boldsymbol{x})$ is linearly shifted from $p_{\mathrm{tr}}(\boldsymbol{x})$.



(b) $p_{\mathrm{te}}(\boldsymbol{x})$ is rotated from $p_{\mathrm{tr}}(\boldsymbol{x})$.

Figure 4.4. Classification examples under covariate shift.

Figure 4.4(a), the correct classification rate of LR+CV is 99.1% while that of IWLR+IWCV is 100%. For the data set depicted in Figure 4.4(b), the correct classification rate of LR+CV is 97.2% while that of IWLR+IWCV is 99.1%. Thus, for both cases, the prediction performance is improved by importance weighting.

5. Discussion

In this section, we compare the proposed LL-KLIEP with existing importance estimation approaches.

5.1 Kernel density estimator

The kernel density estimator (KDE) is a non-parametric technique to estimate a density $p(\boldsymbol{x})$ from samples $\{\boldsymbol{x}_l\}_{l=1}^N$. For the Gaussian kernel, KDE is expressed as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{(2\pi s^2)^{d/2} N} \sum_{l=1}^{N} K_s(\boldsymbol{x}, \boldsymbol{x}_l), \tag{4.27}$$

where $K_s(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian kernel (4.15). The performance of KDE depends on the choice of the kernel width s, which can be optimized by LCV [26]. Note that LCV corresponds to choosing s such that the Kullback-Leibler divergence from $p(\boldsymbol{x})$ to $\hat{p}(\boldsymbol{x})$ is minimized.

KDE can be used for importance estimation by first obtaining $\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$ and $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ separately from $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ and $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\mathrm{te}}}$ and then estimating the importance as $\hat{w}(\boldsymbol{x}) = \hat{p}_{\mathrm{te}}(\boldsymbol{x})/\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$. A potential limitation of this approach is that KDE suffers from the *curse of dimensionality* [26], since the number of samples needed to maintain the same approximation quality grows exponentially as the dimension of the input space increases. This is particularly critical when estimating $p_{\mathrm{tr}}(\boldsymbol{x})$ since the number of training input samples is typically limited. In addition, model selection by LCV is unreliable in such cases, since data splitting in the CV procedure further reduces the sample size. Therefore, in high-dimensional cases LL-KLIEP may be more reliable than the KDE-based approach.

5.2 Kernel mean matching

The kernel mean matching (KMM) method avoids density estimation and directly gives an estimate of the importance at the training input points [31].

The basic idea of KMM is to find $w(\boldsymbol{x})$ such that the mean discrepancy between nonlinearly transformed samples drawn from $p_{\mathrm{te}}(\boldsymbol{x})$ and $p_{\mathrm{tr}}(\boldsymbol{x})$ is minimized in a *universal reproducing kernel Hilbert space* [58]. The Gaussian kernel (4.15) is an example of kernels that induce universal reproducing kernel Hilbert spaces and it has been shown that the solution of the following optimization problem agrees with the true importance:

$$\min_{w(\boldsymbol{x})} \left\| \int K_s(\boldsymbol{x},\cdot) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} - \int K_s(\boldsymbol{x},\cdot) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} \right\|_{\mathcal{F}}^2$$
$$\text{subject to } \int w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} = 1 \text{ and } w(\boldsymbol{x}) \geq 0,$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the norm in the Gaussian reproducing kernel Hilbert space and $K_s(\boldsymbol{x},\boldsymbol{x}')$ is the Gaussian kernel (4.15).

An empirical version of the above problem is reduced to the following quadratic program:

$$\min_{\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}} \left[\frac{1}{2} \sum_{\boldsymbol{x},\boldsymbol{x}'\in D_{\mathrm{tr}}} w(\boldsymbol{x}) w(\boldsymbol{x}') K_s(\boldsymbol{x},\boldsymbol{x}') - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x}) \kappa(\boldsymbol{x}) \right]$$
$$\text{subject to } \left| \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x}) - N_{\mathrm{tr}} \right| \leq N_{\mathrm{tr}} \epsilon, \text{ and}$$
$$0 \leq w(\boldsymbol{x}) \leq B \text{ for all } \boldsymbol{x} \in D_{\mathrm{tr}},$$

where

$$\kappa(\boldsymbol{x}) = \frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}} \sum_{\boldsymbol{x}'\in D_{\mathrm{te}}} K_s(\boldsymbol{x},\boldsymbol{x}').$$

B (≥ 0), ϵ (≥ 0), and s (≥ 0) are tuning parameters. The solution $\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$ is an estimate of the importance at the training input points.

Since KMM does not require density estimates, it is expected to work well even in high dimensional cases. However, the performance is dependent on the tuning parameters B, ϵ, and s and they cannot be optimized easily, e.g., by CV, since estimates of the importance are available only at the training input points.

Thus, an out-of-sample extension is needed to apply KMM in the CV framework, but this currently seems to be an open research issue.

Here, we show that LL-KLIEP(LS2) (see Equation (4.23)) has a tight connection to KMM. Up to irrelevant constants, Equation (4.23) without a regularizer can be expressed as

$$\frac{1}{2}\sum_{\boldsymbol{x},\boldsymbol{x}'\in D_{\mathrm{tr}}} w(\boldsymbol{x})w(\boldsymbol{x}')K_s(\boldsymbol{x},\boldsymbol{x}') - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x})\kappa(\boldsymbol{x}),$$

which is exactly the same form as the objective function of KMM. Thus, KMM and LL-KLIEP(LS2) share a common objective function, although they are derived from very different frameworks.

However, KMM and LL-KLIEP(LS2) still have a significant difference—KMM directly optimizes the importance values $\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$, while LL-KLIEP(LS2) optimizes the parameter $\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$ in the importance model (4.24). Thus, LL-KLIEP(LS2) learns the entire importance function and therefore it allows us to *interpolate* the value of the importance function at any input point. This interpolation property is a significant advantage over KMM since it allows us to use LCV for model selection. Therefore, LL-KLIEP(LS2) may be regarded as an extension of KMM.

5.3 Logistic regression discriminating training and test input data

Another method to directly estimate the importance weights is to use a probabilistic classifier. Let us assign a selector variable $\delta = -1$ to the training inputs and $\delta = 1$ to the test inputs. This means that the training and test input densities are written as

$$p_{\mathrm{tr}}(\boldsymbol{x}) = p(\boldsymbol{x}|\delta = -1), \quad p_{\mathrm{te}}(\boldsymbol{x}) = p(\boldsymbol{x}|\delta = 1).$$

A simple calculation shows that the importance can be expressed in terms of δ as [10]:

$$w(\boldsymbol{x}) = \frac{p(\delta = -1)}{p(\delta = 1)}\frac{p(\delta = 1|\boldsymbol{x})}{p(\delta = -1|\boldsymbol{x})}. \tag{4.28}$$

The probability ratio $p(\delta = -1)/p(\delta = 1)$ may be simply estimated using the ratio of the numbers of training and test input samples. The conditional probability $p(\delta|\boldsymbol{x})$ may be learned by discriminating between the test input samples and the training input samples using LR, where δ plays the role of a class variable (cf. Equation (4.6)). Let us train the LR model by regularized maximum likelihood estimation. The objective function to be maximized is given by

$$\mathrm{LR}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \ln(1 + \exp(\delta_{\boldsymbol{x}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)) - \frac{||\boldsymbol{\alpha}||^2}{2\varsigma^2}, \tag{4.29}$$

where the first term is the main likelihood term, the second term is a normalizer, and the third term is a regularizer. Since this is a convex optimization problem, the global solution can be obtained by standard non-linear optimization methods. The gradient of the objective function is given as

$$\frac{\partial \mathrm{LR}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} \boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} p_{\boldsymbol{\alpha}}(\delta_{\boldsymbol{x}}|\boldsymbol{x}) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{||\boldsymbol{\alpha}||^2}{2\varsigma^2}. \tag{4.30}$$

Then the importance estimate is given by

$$\hat{w}(\boldsymbol{x}) = \frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle). \tag{4.31}$$

We refer to this approach as *LogReg*.

Equation (4.31) shows that the function model of the importance in LogReg is actually the same as that of LL-KLIEP except for a scaling factor (cf. Equation (4.16)). However, the optimization criteria of LL-KLIEP and LogReg are different—in LL-KLIEP, the summation is taken only over the training or test input samples but not both, while the summation in LogReg is over both the training and test input samples. This difference is significant since LogReg does not allow us to use the computational trick we proposed in Section 3.2. Thus LL-KLIEP has the advantage in computation time and storage space consumption over LogReg.

Bickel et al. [10] proposed simultaneous optimization of both importance estimator and classifier. Although their method can perform better than our two stage method which solves importance estimation and classifier's parameter

Table 4.3. Relation between the proposed and related methods.

	Model selection	Direct importance model	Optimization
KDE	Available	Not Available	Analytic
KMM	Not Available	Non-parametric	Constraint quadratic program
LogReg	Available	Log-linear	Unconstraint non-linear
KLIEP	Available	Linear	Constraint non-linear
LL-KLIEP	Available	Log-linear	Unconstraint non-linear

estimation separately, it has a weakness in model selection. Since the hyper-parameter of their method is supposed to be tuned for test samples, CV is not applicable if no labeled test sample is available. On the other hand, our method can select hyper-parameters for both importance estimation by LCV and classification by IWCV.

The characteristics of the proposed and related methods are summarized in Table 4.3.

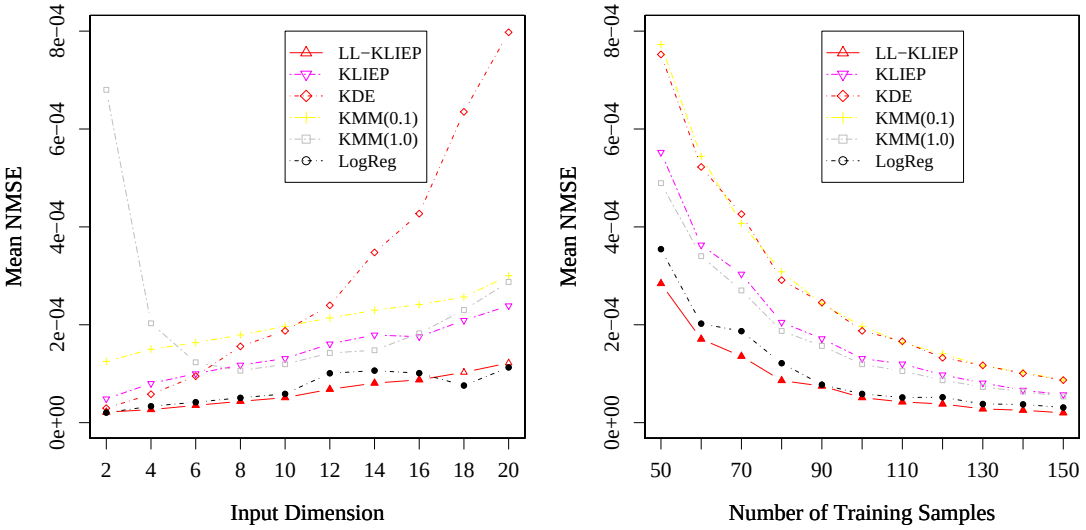
6. Toy Experiments

In this section, we experimentally compare the performance of LL-KLIEP with existing methods.

Let $p_{\mathrm{tr}}(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and $p_{\mathrm{te}}(\boldsymbol{x}) = \mathcal{N}((1, 0, \ldots, 0)^\top, 0.75^2 \boldsymbol{I}_d)$. The task is to estimate the importance at the training input points:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} \text{ for } \boldsymbol{x} \in D_{\mathrm{tr}}.$$

We compared KLIEP, KDE, KMM, LogReg, LL-KLIEP, LL-KLIEP(LS1-a),LL-KLIEP(LS1-b), and LL-KLIEP(LS2). For LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2), we used 5-fold LCV to choose the regularization parameter ς and the kernel width s. For KLIEP, we use 5-fold LCV to choose the kernel width s. For KDE, we used 5-fold LCV to choose the kernel widths for the training and test densities. For KMM, we used $B = 1000$ and $\epsilon = (\sqrt{N_{\mathrm{tr}}} - 1)/\sqrt{N_{\mathrm{tr}}}$ following the suggestion in the original KMM paper [31].



(a) Varying input dimension

(b) Varying training sample size

Figure 4.5. Mean NMSE over 100 trials. The filled plot markers indicate the best method and comparable ones based on the *Wilcoxon* signed rank test at the significance level 1% in terms of the NMSE. 'KMM(s)' denotes KMM with kernel width s.

We tested two different values of the kernel width ($s = 0.1$ and $s = 1.0$) for KMM since there is no reliable method to determine the kernel width. For LogReg, we used 5-fold CV to choose the regularization parameter ς and the kernel width s.

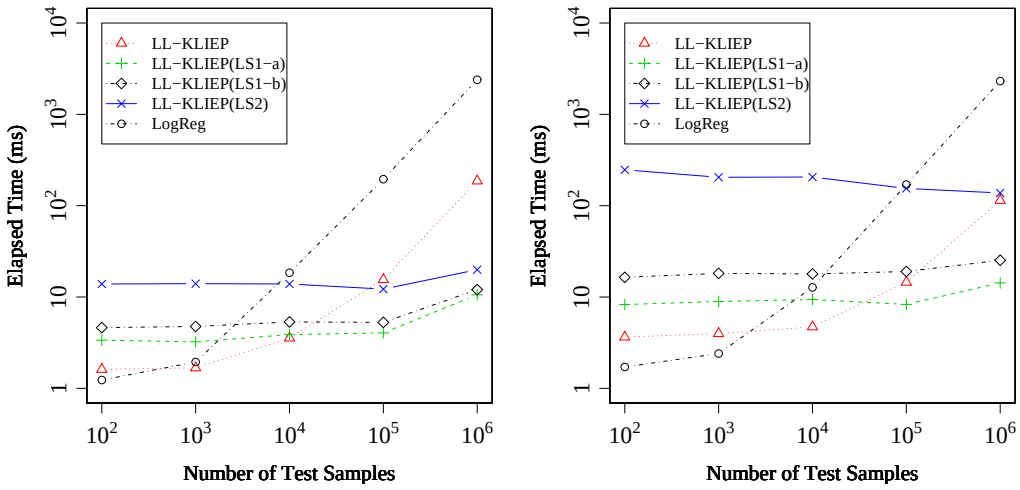
We fixed the number of test input samples at $N_{\mathrm{te}} = 1000$ and considered the following setting for the number of training input samples N_{tr} and the input dimension d:

1. $N_{\mathrm{tr}} = 100$ and $d = 2, 4, \ldots, 20$.

2. $d = 10$ and $N_{\mathrm{tr}} = 50, \ldots, 150$.

We ran the simulation 100 times for each d and N_{tr}, and evaluated the estimation accuracy of $\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$ by the mean NMSE (see Equation (4.26)).

The mean NMSE over 100 trials is plotted in Figure 4.5. The filled plot markers indicate the best method and comparable ones based on the *Wilcoxon* signed rank test at the significance level 1% in terms of the NMSE. We omitted



(a) $d = 10, N_{\mathrm{tr}} = 100$ (b) $d = 10, N_{\mathrm{tr}} = 1000$

Figure 4.6. Average computation time over 100 trials. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the elapsed time (millisecond), respectively.

the graphs of LL-KLIEP(LS1-a), LL-KLIEP(LS1-b) and LL-KLIEP(LS2) since they are almost identical to the result of LL-KLIEP. Figure 4.5(a) shows that the error of KDE sharply increases as the input dimension grows, while LL-KLIEP, KLIEP, and LogReg tend to give much smaller errors than KDE. Figure 4.5(b) shows that the errors of all methods tend to decrease as the number of training samples grows. Again, LL-KLIEP and LogReg are shown to work well. These would be the fruit of directly estimating the importance without going through density estimation. The results of LL-KLIEP and LogReg are slightly better than KLIEP, perhaps because the original KLIEP does not contain a regularizer; we believe that the performance of KLIEP could be improved by adding a regularizer as used in LL-KLIEP and LogReg. KMM also works reasonably well, as long as the kernel width s is chosen appropriately. However, the performance of KMM is highly dependent on s and determining its appropriate value may be difficult. Overall, the accuracy of LL-KLIEP is comparable to the best existing approaches.

Next, we compared the computational cost of LL-KLIEP, LL-KLIEP(LS1-

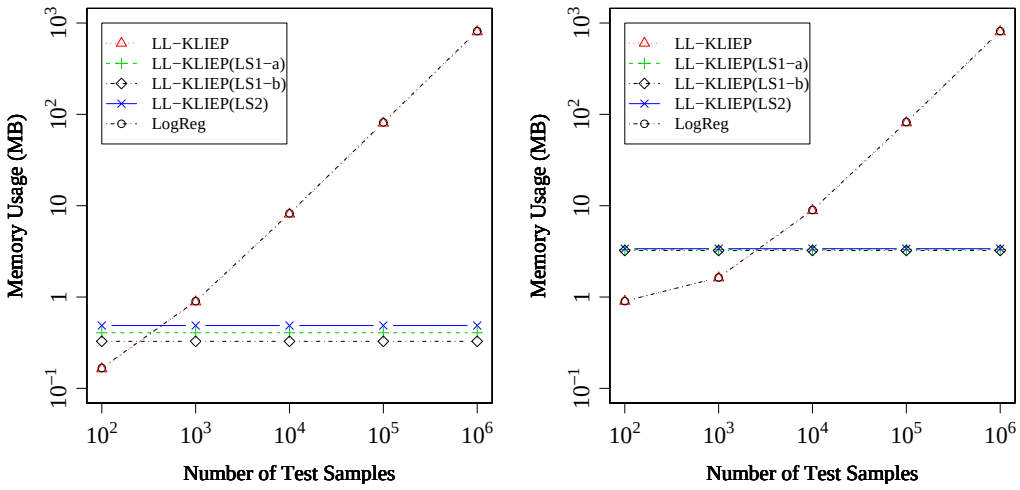
a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg, which have good accuracy in the previous experiments. We investigated the entire computation time of all of them including cross-validation and the precomputation times for the test samples. Note that the Gaussian width s and the regularization parameter ς are chosen over the 5×5 equidistant grid in this experiment for all the methods. We fixed the input dimension at $d = 10$ and changed the number of training input points $N_{\mathrm{tr}} = 10^2, 10^3$ and the number of test samples $N_{\mathrm{te}} = 10^2, 10^3, \ldots, 10^6$. We repeated the experiments 100 times for each N_{tr} and N_{te} on the PC server with an Intel® Xeon® 2.66GHz. All of them are implemented on R (http://www.r-project.org) and *conjugate gradient* method was used to optimize their objective functions.

Figure 4.6 shows the average elapsed times for LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg. The results show that the computational cost of LL-KLIEP and LogReg increases as the amount of test data N_{te} grows, but the computational cost of LL-KLIEP(LS) is nearly independent of the number of test samples N_{te}. This is in good agreement with our theoretical analysis in Section 3.2. Thus the cost of dealing with a large amount of test data in each optimization step is much higher than that at one time precomputation.

We also compared the memory usage of LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg. We used the same implementation and computational environment as the previous experiments. Figure 4.7 shows the memory usage of each method. The results show that the space requirement of LL-KLIEP and LogReg increases as the amount of test data N_{te} grows, but that of LL-KLIEP(LS) is independent of the number of test data N_{te}.

In addition, we compared the computational cost of LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2) in detail. We examined the combination of the following setting for the number of test samples N_{te}, the number of training inputs N_{tr}, and the input dimension d:

- $N_{\mathrm{te}} = 10^2, 10^3, \ldots, 10^6$
- $N_{\mathrm{tr}} = 10^2, 10^3$
- $d = 10^2, 10^3, 10^4$.



(a) $d = 10, N_{\mathrm{tr}} = 100$ (b) $d = 10, N_{\mathrm{tr}} = 1000$

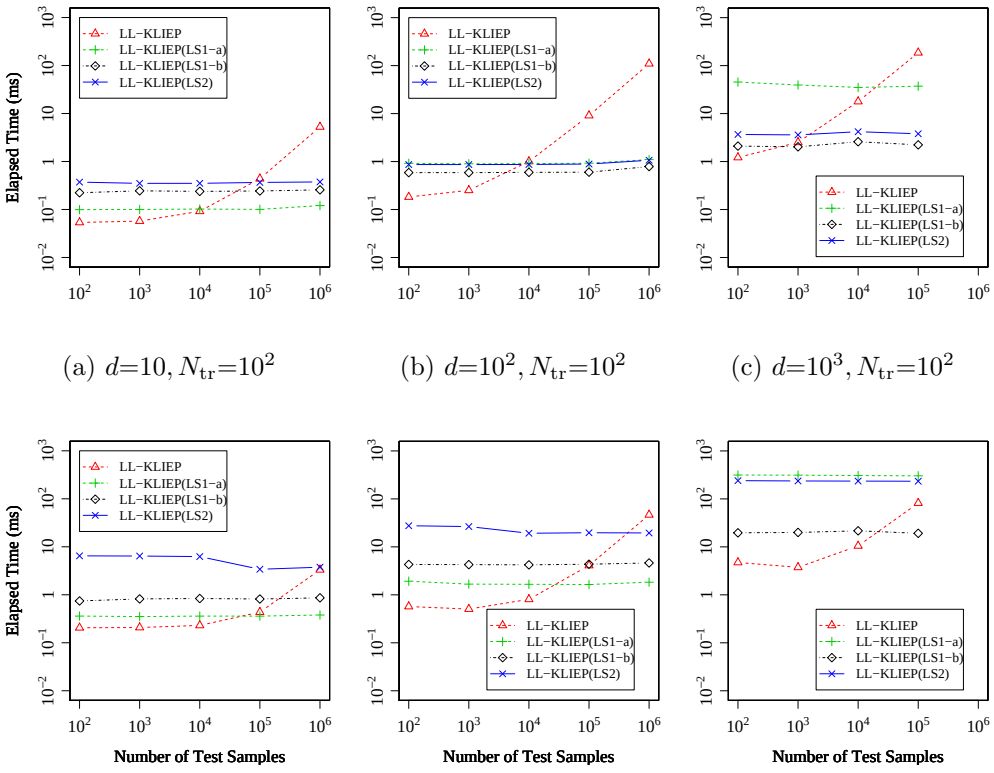
Figure 4.7. Memory usage. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the memory usage (MB), respectively.

In this experiment, we used a linear basis function so that the number of bases is equivalent to the input dimension. Since the computational time of cross-validation is conceptually a scalar multiple of that of each optimization step, we compared the computational time including the precomputation times for the test inputs after the model parameters are fixed. We repeated the experiments 100 times for each N_{te}, N_{tr}, and d using the same implementation and computational environment as the previous experiments.

Figure 4.8 shows the average elapsed times for LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2). When $d = 10^3$, the result of $N_{\mathrm{te}} = 10^6$ was excluded because of the large memory requirements. As we expected,

1. LL-KLIEP is faster than LL-KLIEP(LS) when the number of test samples is small,

2. LL-KLIEP(LS1-a) is faster than LL-KLIEP(LS2) for lower dimensional data,



(a) d=10, N_{tr}=10^2 (b) d=10^2, N_{tr}=10^2 (c) d=10^3, N_{tr}=10^2

(d) d=10, N_{tr}=10^3 (e) d=10^2, N_{tr}=10^3 (f) d=10^3, N_{tr}=10^3

Figure 4.8. Average computation time over 100 trials. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the elapsed time (millisecond), respectively.

3. both LL-KLIEP(LS1-b) and LL-KLIEP(LS2) are advantageous for high dimensional problems, and

4. the computational cost of LL-KLIEP(LS1-b) increases for the larger amount of training inputs.

Since LL-KLIEP(LS1-b) outperformed LL-KLIEP(LS2) in all the settings, we conclude LL-KLIEP(LS1-b) is more suitable for high dimensional problems. However, LL-KLIEP(LS1-a) is faster than LL-KLIEP(LS1-b) for lower dimensional data against the complexity analysis. One reason for this result might be that the computation of LL-KLIEP(LS1-b) is relatively complex in the iteration of the training inputs compared with LL-KLIEP(LS1-a). Therefore, LL-KLIEP(LS1-a) runs faster than LL-KLIEP(LS1-b) if the number of dimensions is small enough not to ignore this overhead.

7. Experiments

Now, we show the empirical result that LL-KLIEP is applied to JWS task. It is reasonable to consider the domain adaptation task of word segmentation systems as a covariate shift adaptation problem since word segmentation policy ($p(\boldsymbol{y}|\boldsymbol{x})$) is rarely changed between domains in the same language, but the distribution of characters ($p(\boldsymbol{x})$) tends to be changed between domains.

In this experiments, we used the same corpus as in Section 6 of Chapter 3. We used the segmented data in the source domain as training data and the unsegmented data in the target domain as test data. For the unsegmented data for the target domain, we used extra $53,834$ sentences of the medical reference manual, denoted as data (D).

We implemented a CRF training algorithm (IWCRF) in which the original log likelihood is weighted by importance.

$$\begin{aligned}\mathrm{IWCRF}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} w(\boldsymbol{x}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}|\boldsymbol{x}^{(n)}) \\ &= \sum_{n=1}^{N} w(\boldsymbol{x}) \left(\left\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}) \right\rangle - \ln Z_{\boldsymbol{\theta}, \boldsymbol{x}^{(n)}, \boldsymbol{Y}} \right).\end{aligned}$$

Note that, since the loss function of CRFs is defined as the sum of the log loss for each sentence, $w(\boldsymbol{x})$ also should be evaluated for each sentence. We implemented the first order Markov CRFs which uses the same feature and optimizer as Section 6 of Chapter 3).

To estimate the importance of each sentence by LL-KLIEP, we used the source domain data (A) and (B) as the training examples D_{tr} and the target domain data (D) as the test samples D_{te}. We defined the basis function for the importance model $\hat{w}(\boldsymbol{x})$ as the average value of observation features in a sentence:

$$\boldsymbol{\psi}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t. \tag{4.32}$$

In this experiment, we used the observation features defined in Table 3.4. To determine the hyper-parameter of LL-KLIEP, we used 5-fold CV.

We tuned the hyper parameter σ of IWCRF with *importance weighted F measure score*, IWF, in which the number of correct words is weighted by the importance of the sentence that the word belong to:

$$\mathrm{IWF}(D) = \frac{2 \times \mathrm{IWR}(D) \times \mathrm{IWP}(D)}{\mathrm{IWR}(D) + \mathrm{IWP}(D)}$$

for the validation set D where

$$\mathrm{IWR}(D) = \frac{\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} w(\boldsymbol{x}) \sum_{v_t \in \boldsymbol{y}} [\hat{v}_t = v_t]}{\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \sum_{v_t \in \boldsymbol{y}} w(\boldsymbol{x})} \times 100$$

$$\mathrm{IWP}(D) = \frac{\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} w(\boldsymbol{x}) \sum_{v_t \in \boldsymbol{y}} [\hat{v}_t = v_t]}{\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \sum_{\hat{v}_t \in \hat{\boldsymbol{y}}} w(\boldsymbol{x})} \times 100, \text{ and}$$

v_t denotes a t-th word in a sentence $(\boldsymbol{x}, \boldsymbol{y})$ and $\hat{v}_t$ denotes a t-th word of a system prediction $\hat{\boldsymbol{y}} = \mathrm{argmax}_{\boldsymbol{y}}\, p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. We used 1/10 of training data D_{tr} as the validation set.

Then, we compared the target domain performance between CRF, IWCRF, and CRF which was trained additionally using $1,000$ partial word annotations of the target domain CRF as mentioned in Section 6, denoted as "CRF + $1,000$". For importance estimation, we compared LL-KLIEP and LogReg for which we employed 5-fold CV to find the optimal hyper-parameter ς.

As we mentioned in Section 1, the occurrences of domain specific words increase the errors in adaptation phase. Therefore, we also compared the recall

Table 4.4. Word segmentation performance in the target domain. "CRF + 1,000" stands for the performance of a CRF additionally using 1,000 manual word segmentations of the target domain.

	F	R	P	OOV R
CRF	92.30	90.58	94.08	84.11
IWCRF (LL-KLIEP)	**94.46**	**94.32**	94.59	89.66
IWCRF (LogReg)	93.68	94.30	93.07	88.94
CRF + 1,000	94.43	93.49	**95.39**	**90.4**

value of words which is not appeared in the source domain data (A) but appeared in the target domain test data (C), denoted as OOV R (Out of Vocabulary Recall):

$$OOVR = \frac{\#\text{ of correct } \textbf{unknown} \text{ words}}{\#\text{ of } \textbf{unknown} \text{ words in test data}} \times 100$$

Table 4.4 shows the result of the performance of each method. Surprisingly, the F score of IWCRF outperformed not only that of CRF, but also that of "CRF + 1,000", so the benefit of the importance weighting is worth the manual annotation of 1,000 words. In analysis in depth, "CRF + 1,000" showed the better performance in the Precision (P) and OOV R. This implies the partial annotation of the word list improves the accurate segmentation of domain specific words. On the other hand, the covariate shift adaptation technique improves the coverage in the target domain.

8. Summary

In this chapter, we addressed the problem of estimating the importance for covariate shift adaptation. We proposed a scalable direct importance estimation method called *LL-KLIEP*. The computation time of LL-KLIEP is nearly independent of the amount of test data, which is a significant advantage over existing approaches when we deal with a large number of test samples. Our experiments highlighted this advantage, and we experimentally confirmed that the accuracy of the proposed method is comparable to the best existing methods. We concluded

that the proposed method is a promising method for large-scale covariate shift adaptation. We also confirmed that the proposed domain adaptation technique improves the performance of JWS in the domain adaptation task.

Chapter 5

Conclusion and Future Work

This thesis has considered how to *adapt* the statistical word segmentation system to new domain text. The domain adaptation of word segmentation system addresses the situation in which we possess a large amount of segmented data for a source domain but little or no segmented data for a target domain where we wish to apply the model. When we apply natural language processing to the real world data, the domain adaptation of word segmentation systems is a first step to successfully process a target text.

The first part of this thesis address the integration between effective segmented corpus building system and the state-of-the-art structured output learning system. We proposed a method to train CRFs using partially segmented sentences which are easier to build than fully segmented sentences for new target domains. The second part of this thesis address the situation where we have a large amount of unsegmented data for the target domain. In this work, each segmented sentences in the source domain is weighted by an importance so that the expected error of a learned model is minimized in the target domain. We proposed an importance estimation algorithm which efficiently processes unsegmented data in the target domain.

We highlight a few directions of future research in the rest of this chapter.

1. Partial Annotations for Other NLP Tasks

We believe that partial annotations in Chapter 3 are also useful to other tasks in NLP, such as syntactic parsing, information extraction, and so on. However, there are some NLP tasks, such as the word alignment task [67], in which it is not possible to efficiently calculate the sum score of all of the possible label configurations (the partition function). Recently, Verbeek et al. [70] independently proposed a parameter estimation method for CRFs using partially labeled images. Although the objective function in their formulation is equivalent to Equation (3.4), they used *Loopy Belief Propagation* to approximate the partition function for their application (scene segmentation). Their results imply these approximation methods can be used for such applications that cannot use dynamic programming techniques.

2. Ambiguous Annotations

The problem formulation of partial annotations in Section 2 of Chapter 3 is directly extended to ambiguous annotations which are a set of candidate labels annotated for a part of a structured instance. For many NLP tasks, it is sometimes difficult to decide which label is appropriate in a particular context. For example, the following sentence from the Penn treebank (PTB) corpus includes an ambiguous annotation for the part-of-speech (POS) tag of "pending":

$$\text{That/DT suit/NN is/VBZ pending/VBG|JJ ./. ,}$$

where words are paired with their part-of-speech tag by a forward slash ("/").[1] Uncertainty concerning the proper POS tag of "pending" is represented by the disjunctive POS tag ("VBG and JJ") as indicated by a vertical bar. According to the formulation in Section 2, this example sentence can be represented as:

$$\boldsymbol{L} = (\{\text{DT}\}, \{\text{NN}\}, \{\text{VBZ}\}, \underbrace{\{\text{VBG}, \text{JJ}\}}_{\text{ambiguous annotation}}, \{.\}).$$

[1]These POS tags used here are DT:determiner, NN:common noun, VBZ:present tense 3rd person singular verb, VBG:gerund or present participle verb, and JJ:adjective.

We observed promising results for a POS tagging task using ambiguous annotations that are contained in the PTB corpus [68].

We believe that ambiguous annotations are more common in the tasks that deal with semantics, such as information extraction tasks. Worth mentioned that each ambiguous part can have dependency on the label of the other ambiguous part in these tasks. For example, an annotated phrase for named entity (NE) extraction can consist of the set of NE label candidates over more than two words such as

White/ORG|LOC House/ORG|LOC

where ORG represents organization names and LOC represents location names. In this case, both of the two words ("White" and "House") should be labeled by the same NE label, but this kind of label dependency can not be described by the representation in Section 2. Although it may not be difficult to generalize the representation for these label dependencies, it is not trivial to efficiently compute the partition function for the long term dependency of ambiguous parts. The development of CRF learning algorithm using these ambiguous annotations is an interesting topic for future work.

3. Support Vector Learning Using Partial Annotations

Instead of CRFs, it is also an interesting direction to train other structured output model using partition annotations. Although the conditional probability model is trained by the regularized maximum likelihood estimation in Chapter 3, one may maximize the log likelihood ratio:

$$r_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \ln \frac{\boldsymbol{p}_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}{\max_{\tilde{\boldsymbol{y}} \neq \boldsymbol{y}} \boldsymbol{p}_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}$$

between the correct label and the most incorrect labeling. Let be $\max(1 - r_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}), 0)$ is a loss function which penalized by the instances whose ratio is smaller than one. This loss function is a generalization of the hinge loss to multi-class Support Vector Machine (SVM) [49]. According to the treatment of missing

values in this framework [57], The log likelihood ratio of a partially annotated sentence can be defined as:

$$\begin{aligned} r_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) &= \ln \frac{\sum_{\boldsymbol{y} \in \boldsymbol{Y_L}} \boldsymbol{p_\theta}(\boldsymbol{y}|\boldsymbol{x})}{\max_{\tilde{\boldsymbol{y}} \notin \boldsymbol{Y_L}} \boldsymbol{p_\theta}(\boldsymbol{y}|\boldsymbol{x})} \\ &= \ln Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y_L}} - \max_{\boldsymbol{y} \notin \boldsymbol{Y_L}} \langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle . \end{aligned}$$

Then, we can derive the constrained minimization problem for training SVM using partial annotations.

$$\begin{aligned} &\min \sum_{n=1}^{N+M} \xi^{(n)} + \frac{||\boldsymbol{\theta}||^2}{2\sigma^2} \\ &\text{subject to } \ln Z_{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{Y_L}^{(n)}} - \max_{\boldsymbol{y} \notin \boldsymbol{Y_L}^{(n)}} \left\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}^{(n)}, \boldsymbol{y}) \right\rangle \geq 1 - \xi^{(n)} \text{ and } \xi^{(n)} \geq 0. \end{aligned}$$

Although the constraints of the problem is not convex, we can still find a local optimum for this problem by Concave-Convex Procedure (CCCP)[57].

4. Joint Density Ratio Estimation

In Chapter 4, we treat the domain difference as covariate shift in which we assume the input distribution differs in the training and test phases, but the conditional distribution remains unchanged, i.e. $p_{\mathrm{tr}}(\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{x})$ and $p_{\mathrm{tr}}(\boldsymbol{y}|\boldsymbol{x}) = p_{\mathrm{te}}(\boldsymbol{y}|\boldsymbol{x})$. However this assumption may too strong, and the conditional distribution can also be changed, i.e. $p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y}) \neq p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y})$. This situation can reasonably appear in the domain adaptation of word sense disambiguation (WSD), sentiment analysis (SA), and so on. Since it is natural that the meaning of words or the positive/negative evaluation of items can differ between the domains, we need to assume that classification rules of WSD or SA can be changed, i.e. $p_{\mathrm{tr}}(\boldsymbol{y}|\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{y}|\boldsymbol{x})$. In this case, we may want to weight training instances by *joint density ratio*,

$$w(\boldsymbol{x}, \boldsymbol{y}) = \frac{p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y})}{p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})},$$

to minimize the expected loss in the target domain:

$$\begin{aligned}&\iint p_{\mathrm{te}}(\boldsymbol{x},\boldsymbol{y})\mathrm{Loss}(\boldsymbol{x},\boldsymbol{y},f_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}d\boldsymbol{y}\\&=\iint p_{\mathrm{tr}}(\boldsymbol{x},\boldsymbol{y})w(\boldsymbol{x},\boldsymbol{y})\mathrm{Loss}(\boldsymbol{x},\boldsymbol{y},f_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}d\boldsymbol{y}.\end{aligned}$$

So the estimation method of the joint density ratio can be an interesting direction of the domain adaptation research. Bickel et al. factorize the joint density ratio model $\hat{w}(\boldsymbol{x},\boldsymbol{y})$ into two parts,

$$\hat{w}(\boldsymbol{x},\boldsymbol{y})=\hat{u}(\boldsymbol{x},\boldsymbol{y})\hat{v}(\boldsymbol{x}),$$

and estimate $\hat{u}(\boldsymbol{x},\boldsymbol{y})$ and $\hat{v}(\boldsymbol{x})$ separately [11]. In a similar way as their approach, it can be possible to estimate $\hat{u}(\boldsymbol{x},\boldsymbol{y})$ using labeled examples of both source and target domain, and estimate $\hat{v}(\boldsymbol{x})$ by LL-KLIEP. However, since a small amount of labeled data in a target domain is often available for the domain adaptation of NLP tasks, we need to take into account the stability of the joint density ratio estimation.

Bibliography

[1] Brigham Anderson and Andrew Moore. Active learning for hidden Markov models: Objective functions and algorithms. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 9–16, 2005.

[2] Brigham Anderson, Sajid Siddiqi, and Andrew Moore. Sequence selection for active learning. Technical Report CMU-IR-TR-06-16, Carnegie Mellon University, 2006.

[3] Shlomo Argamon-Engelson and Ido Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[4] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* MIT Press, Cambridge, 1998.

[5] Mark H. Beers. *The Merck Manual of Medical Information (in Japanese).* Nikkei Business Publications, Inc, Home edition, 2004.

[6] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.

[7] S. Bickel. ECML 2006 discovery challenge workshop, 2006.

[8] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems.* MIT Press, Cambridge, MA, 2007.

[9] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, , and Tobias Scheffer. Multi-task learning for hiv therapy screening. In *In Proceedings of the International Conference on Machine Learning*, 2008.

[10] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81 – 88. ACM Press, 2007.

[11] Steffen Bickel, Christoph Sawade, and Tobias Scheffer. Transfer learning by distribution matching for targeted advertising. In *Advances in Neural Information Processing Systems*, 2009.

[12] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006.

[13] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, chapter 3.2.1. Cambridge University Press, 2004.

[15] J. Q. Candela, N. Lawrence, A. Schwaighofer, and M. Sugiyama. NIPS 2006 workshop on learning when test and training inputs have different distributions, 2006.

[16] Yee Seng Chan and Hwee Tou Ng. Domain adaptation with active learning for word sense disambiguation. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 49–56, 2007.

[17] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 285–292, 2004.

[18] Stephen Clark and James R. Curran. Partial training for a lexicalized-grammar parser. In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, pages 144–151, 2006.

[19] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[20] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

[21] Aron Culotta, Trausti Kristjansson, Andrew McCallum, and Paul Viola. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence Journal*, 170:1101–1122, 2006.

[22] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *In Proceedings of the International Conference on Machine Learning*, pages 193–200, 2007.

[23] Hal III Daumé. Frustratingly easy domain adaptation. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 256–263, 2007.

[24] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[25] G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, Berlin, 1996.

[26] Wolfgang Härdle, Marlene Müller, Stefan Sperlich, and Axel Werwatz. *Nonparametric and Semiparametric Models*. Springer Series in Statistics. Springer, Berlin, 2004.

[27] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[28] David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the trec-8 web track. In *Proceedings of the Eighth Text REtrieval Conference*, pages 131–150.

[29] J. J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–162, 1979.

[30] Hachiya Hirotaka, Takayuki Akiyama, Masashi Sugiyama, and Jan Peters. Adaptive importance sampling with automatic model selection in value function approximation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, USA, 2008.

[31] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, Cambridge, MA, 2007. MIT Press.

[32] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 264–271, 2007.

[33] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, pages 209–216, 2006.

[34] T. Kanamori and H. Shimodaira. Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, 116(1):149–162, 2003.

[35] Donald Keene, Hiroyoshi Hatori, Haruko Yamada, and Shouko Irabu, editors. *Japanese-English Sentence Equivalents (in Japanese).* Asahi Press, Electronic book edition, 1992.

[36] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of Empirical Methods in Natural Language Processing*, 2004.

[37] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[38] Xiaoquan Luo. A maximum entropy chinese character-based parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 192–199, 2003.

[39] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of Human Language Technologies Conference - the Annual Conference of the North American Chapter of of the Association for Computational Linguistics (NAACL-HLT)*, pages 109–112, 2007.

[40] Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, May 1999.

[41] Shinsuke Mori. Language model adaptation with a word list and a raw corpus. In *Proceedings of the 9th International Conference on Spoken Language Processing*, 2006.

[42] Masaaki Nagata. A stochastic Japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. In *Proceedings of the 15th conference on Computational linguistics*, pages 201–207, 1994.

[43] Masaaki Nagata. Automatic extraction of new words from Japanese texts using generalized forward-backward search. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 48–59, Somerset, New Jersey, 1996. Association for Computational Linguistics.

[44] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, 2nd edition edition, June 2006.

[45] Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the International Conference on Computational Linguistics*, 2004.

[46] Fernando C. N. Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of Annual Meeting Association of Computational Linguistics*, pages 128–135, 1992.

[47] Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. In *Advances in Neural Information Processing Systems*, pages 1153–1160, Cambridge, MA, 2008. MIT Press.

[48] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems*, 2004.

[49] Gunnar Rätsch and Alexander J. Smola. Adapting codes and embeddings for polychotomies. In *Advances in Neural Information Processing Systems*, 2002.

[50] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *Proceedings of the European Conference on Machine Learning*, pages 413–424. Springer, September 2006.

[51] Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, 2005.

[52] Tobias Scheffer and Stefan Wrobel. Active learning of partially hidden markov models. In *Proceedings of the ECML/PKDD Workshop on Instance Selection*, 2001.

[53] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, Edmonton, Canada, 2003.

[54] C. R. Shelton. *Importance Sampling for Reinforcement Learning with Multiple Objectives*. PhD thesis, Massachusetts Institute of Technology, 2001.

[55] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[56] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[57] Alex J. Smola, S.V.N. Vishwanathan, and Thomas Hofmann. Kernel methods for missing variables. In *Proc. 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[58] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[59] M. Sugiyama. Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, 7:141–166, Jan. 2006.

[60] M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, May 2007.

[61] M. Sugiyama and K.-R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4):249–279, 2005.

[62] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 2008. MIT Press.

[63] Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the International Conference on Machine Learning*, 2004.

[64] R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[65] Jun Suzuki and Hideki Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *In Proceedings of the*

46th Annual Meeting of the Association for Computational Linguistics, pages 665–673, 2008.

[66] Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

[67] Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2005.

[68] Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. Training conditional random fields using incomplete annotations. In *Proceedings of 22nd International Conference on Computational Linguistics*, pages 897–904, 2008.

[69] Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 91–99, 2001.

[70] Jakob Verbeek and Bill Triggs. Scene segmentation with CRFs learned from partially labeled images. In *Advances in Neural Information Processing Systems*, pages 1553–1560, Cambridge, MA, 2008. MIT Press.

[71] G. Wahba. *Spline Model for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1990.

[72] D. P. Wiens. Robust weights and designs for biased regression models: Least squares and generalized M-estimation. *Journal of Statistical Planning and Inference*, 83(2):395–412, 2000.

[73] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

[74] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 21st International Conference on Machine Learning*, New York, NY, 2004. ACM Press.

List of Publications

Journal Papers

1. Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. "Training Conditional Random Fields Using Partial Annotations for Domain Adaptation of Japanese Word Segmentation" IPSJ Journal, 2009 (in Japanese) (to appear)

2. Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. "Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation" Journal of Information Processing, Vol. 17, 2009

International Conferences (Refereed)

1. Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. "Training Conditional Random Fields Using Incomplete Annotations" Proceedings of 22nd International Conference on Computational Linguistics (COLING), 2008

2. Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. "Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation", Proceedings of SIAM International Conference on Data Mining (SDM), 2008

Workshops

1. Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. "Training Conditional Random Fields Using Partial and Ambiguous Structured Labels" IPSJ SIG-NL (NL-182), 2007 (in Japanese)

Other Publications (Not Included in This Thesis)

Journal Papers

1. Hisashi Kashima, Shoko Suzuki, Shohei Hido, Yuta Tsuboi, Toshihiro Takahashi, Tsuyoshi Ide, Rikiya Takahashi and Akira Tajima "A Semisupervised Approach Using Spatio-temporal Information for Indoor Location Estimation" In Qiang Yang, Sinno Jialin Pan and Vincent Wenchen Zheng, Estimating Location Using Wi-Fi, IEEE Intelligent Systems, Vol. 23, No. 1, pp. 8-13, 2008.

2. Shoko Suzuki, Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Toshihiro Takahashi, Tsuyoshi Ide, Rikiya Takahashi and Akira Tajima "A Dimensionality Reduction Approach" In Qiang Yang, Sinno Jialin Pan and Vincent Wenchen Zheng, Estimating Location Using Wi-Fi, IEEE Intelligent Systems, Vol. 23, No. 1, pp. 8-13, 2008.

International Conferences (Refereed)

1. Yuta Tsuboi and Hisashi Kashima. "A New Objective Function for Sequence Labeling" Proceedings of International Conference on Pattern Recognition (ICPR), 2008

2. Hisashi Kashima and Yuta Tsuboi. "Kernel-based Discriminative Learning Algorithms for Labeling Sequences, Trees and Graphs" Proceedings of 21st International Conference on Machine Learning (ICML), 2004

Workshops

1. Yuta Tsuboi and Hisashi Kashima. "Design of Discriminative Models for Labeling Structured Data" Workshop on Information-Based Induction Sciences(IBIS), 2005 (in Japanese)

2. Hisashi Kashima and Yuta Tsuboi. "Kernel-based Discriminative Learning Algorithms for Labeling Structured Data" IEICE SIG-AI, 2004 (in Japanese)

3. Yuta Tsuboi. "Mining Frequent Substrings" IPSJ SIG-NL, 2003 (in Japanese)

4. Yuta Tsuboi and Yuji Matsumoto. "Authorship Identification for Heterogeneous Documents" IPSJ SIG-NL, 2002 (in Japanese)

5. Kudo Taku, Yamamoto Kaoru, Tsuboi Yuta, and Matsumoto Yuji. "Text Mining using Linguistic Information" IPSJ SIG-NL, 2002 (in Japanese)

Appendix

A. Computation of Objective and Derivative functions

Here we explain the effective computation procedure for Equation (3.4) and (3.5) in Chapter 3 using dynamic programming techniques.

Under the first-order Markov assumption[2], two types of features are usually used: one is pairs of an observed variable and a label variable (denoted as $\boldsymbol{f}(x_t, y_t) : X \times Y$), the other is pairs of two label variables (denoted as $\boldsymbol{g}(y_{t-1}, y_t) : Y \times Y$) at time t. Then the feature vector can be decomposed as $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=1}^{T+1} \boldsymbol{\phi}(x_t, y_{t-1}, y_t)$ where $\boldsymbol{\phi}(x_t, y_{t-1}, y_t) = \boldsymbol{f}(x_t, y_t) + \boldsymbol{g}(y_{t-1}, y_t)$. In addition, let S and E be special label variables to encode the beginning and ending of a sequence, respectively. We define $\boldsymbol{\phi}(x_t, y_{t-1}, y_t)$ to be $\boldsymbol{\phi}(x_t, S, y_t)$ at the head $t = 1$ and $\boldsymbol{g}(y_{t-1}, E)$ at the tail where $t = T + 1$. The technique of the effective calculation of the normalization value is the precomputation of the $\alpha_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}[t, j]$, and$\beta_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}[t, j]$ matrices with given $\boldsymbol{\theta}, \boldsymbol{x}$, and $\boldsymbol{L}$. The matrices α and β are defined as follows, and should be calculated in the order of $t = 1, \cdots, T$,

[2]Note that, although the rest of the explanation based on the first-order Markov models for purposes of illustration, the following arguments are easily extended to the higher order Markov CRFs and semi-Markov CRFs.

and $t = T+1, \cdots, 1$, respectively

$$\alpha_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}[t,j] = \begin{cases} 0 & \text{if } j \notin L_t \\ \langle \boldsymbol{\theta}, \boldsymbol{\phi}(x_t, S, j) \rangle) & \text{else if } t = 1 \\ \ln \sum_{i \in L_{t-1}} e^{\alpha[t-1,i] + \langle \boldsymbol{\theta}, \boldsymbol{\phi}(x_t, i, j) \rangle} & \text{else} \end{cases}$$

$$\beta_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}[t,j] = \begin{cases} 0 & \text{if } j \notin L_t \\ \langle \boldsymbol{\theta}, \boldsymbol{g}(j, E) \rangle & \text{else if } t = T+1 \\ \ln \sum_{k \in L_{t+1}} e^{\langle \boldsymbol{\theta}, \boldsymbol{\phi}(x_t, j, k) \rangle + \beta[t+1,k]} & \text{else} \end{cases}$$

Note that $\boldsymbol{L} = (Y, \cdots, Y)$ is used to calculate all the entries in $\boldsymbol{Y}$. In the rest of this section, we omit the subscripts $\boldsymbol{\theta}, \boldsymbol{x}$, and $\boldsymbol{L}$ of α, β, Z unless misunderstandings could occur. The time complexity of the $\alpha[t,j]$ or $\beta[t,j]$ computation is $O(T|Y|^2)$.

Finally, Equations (3.4) and (3.5) are efficiently calculated using α, β. The logarithm of Z in Equation (3.4) is calculated as:

$$\ln Z_{\boldsymbol{\theta},\boldsymbol{Y_L}} = \ln \sum_{j \in L_T} e^{\alpha_{\boldsymbol{\theta},\boldsymbol{L}}[T,j] + \langle \boldsymbol{\theta}, \boldsymbol{g}(j,E) \rangle}.$$

Similarly, the first and second terms of Equation (3.5) can be computed as:

$$\sum_{\boldsymbol{y} \in \boldsymbol{Y_L}} p_{\boldsymbol{\theta},\boldsymbol{L}}(\boldsymbol{y}|\boldsymbol{x}) \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in L_T} \varepsilon_{\boldsymbol{L}}(T, i, E) \boldsymbol{g}(i, E)$$
$$+ \sum_{t=1}^{T} \sum_{j \in L_t} \left(\gamma_{\boldsymbol{L}}(t,j) \boldsymbol{f}(x_t, j) + \sum_{i \in L_{t-1}} \varepsilon_{\boldsymbol{L}}(t,i,j) \boldsymbol{g}(i,j) \right)$$

where $\boldsymbol{\theta}, \boldsymbol{x}$ are omitted in this equation, and $\gamma_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}$ and $\varepsilon_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}$ are the marginal probabilities:

$$\begin{aligned} \gamma_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}(t,j) &= p_{\boldsymbol{\theta},\boldsymbol{L}}(y_t = j|\boldsymbol{x}) \\ &= e^{\alpha[t,j] + \beta[t,j] - \ln Z_{\boldsymbol{Y_L}}} \text{ , and} \\ \varepsilon_{\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{L}}(t,i,j) &= p_{\boldsymbol{\theta},\boldsymbol{L}}(y_{t-1} = i, y_t = j|\boldsymbol{x}) \\ &= e^{\alpha[t-1,i] + \langle \boldsymbol{\theta}, \boldsymbol{\phi}(x_t,i,j) \rangle + \beta[t,j] - \ln Z_{\boldsymbol{Y_L}}}. \end{aligned}$$

Note that $\boldsymbol{Y_L}$ is replaced with $\boldsymbol{Y}$ and $\boldsymbol{L} = (Y, \cdots, Y)$ to compute the second term.