Institut für Informatik
Arbeitsgruppe Maschinelles Lernen

Learning under Differing Training and Test Distributions

Dissertation

zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)
in der Wissenschaftsdisziplin Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
Steffen Bickel

Potsdam, den 22.07.2009

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2009/3333/
URN urn:nbn:de:kobv:517-opus-33331
[http://nbn-resolving.org/urn:nbn:de:kobv:517-opus-33331]

Abstract

One of the main problems in machine learning is to train a predictive model from training data and to make predictions on test data. Most predictive models are constructed under the assumption that the training data is governed by the exact same distribution which the model will later be exposed to. In practice, control over the data collection process is often imperfect. A typical scenario is when labels are collected by questionnaires and one does not have access to the test population. For example, parts of the test population are underrepresented in the survey, out of reach, or do not return the questionnaire. In many applications training data from the test distribution are scarce because they are difficult to obtain or very expensive. Data from auxiliary sources drawn from similar distributions are often cheaply available.

This thesis centers around learning under differing training and test distributions and covers several problem settings with different assumptions on the relationship between training and test distributions—including multi-task learning and learning under covariate shift and sample selection bias. Several new models are derived that directly characterize the divergence between training and test distributions, without the intermediate step of estimating training and test distributions separately. The integral part of these models are rescaling weights that match the rescaled or resampled training distribution to the test distribution. Integrated models are studied where only one optimization problem needs to be solved for learning under differing distributions. With a two-step approximation to the integrated models almost any supervised learning algorithm can be adopted to biased training data.

In case studies on spam filtering, HIV therapy screening, targeted advertising, and other applications the performance of the new models is compared to state-of-the-art reference methods.

Zusammenfassung

Eines der wichtigsten Probleme im Maschinellen Lernen ist das Trainieren von Vorhersagemodellen aus Trainingsdaten und das Ableiten von Vorhersagen für Testdaten. Vorhersagemodelle basieren üblicherweise auf der Annahme, dass Trainingsdaten aus der gleichen Verteilung gezogen werden wie Testdaten. In der Praxis ist diese Annahme oft nicht erfüllt, zum Beispiel, wenn Trainingsdaten durch Fragebögen gesammelt werden. Hier steht meist nur eine verzerrte Zielpopulation zur Verfügung, denn Teile der Population können unterrepräsentiert sein, nicht erreichbar sein, oder ignorieren die Aufforderung zum Ausfüllen des Fragebogens. In vielen Anwendungen stehen nur sehr wenige Trainingsdaten aus der Testverteilung zur Verfügung, weil solche Daten teuer oder aufwändig zu sammeln sind. Daten aus alternativen Quellen, die aus ähnlichen Verteilungen gezogen werden, sind oft viel einfacher und günstiger zu beschaffen.

Die vorliegende Arbeit beschäftigt sich mit dem Lernen von Vorhersagemodellen aus Trainingsdaten, deren Verteilung sich von der Testverteilung unterscheidet. Es werden verschiedene Problemstellungen behandelt, die von unterschiedlichen Annahmen über die Beziehung zwischen Trainings- und Testverteilung ausgehen. Darunter fallen auch Multi-Task-Lernen und Lernen unter Covariate Shift und Sample Selection Bias. Es werden mehrere neue Modelle hergeleitet, die direkt den Unterschied zwischen Trainings- und Testverteilung charakterisieren, ohne dass eine einzelne Schätzung der Verteilungen nötig ist. Zentrale Bestandteile der Modelle sind Gewichtungsfaktoren, mit denen die Trainingsverteilung durch Umgewichtung auf die Testverteilung abgebildet wird. Es werden kombinierte Modelle zum Lernen mit verschiedenen Trainings- und Testverteilungen untersucht, für deren Schätzung nur ein einziges Optimierungsproblem gelöst werden muss. Die kombinierten Modelle können mit zwei Optimierungsschritten approximiert werden und dadurch kann fast jedes gängige Vorhersagemodell so erweitert werden, dass verzerrte Trainingsverteilungen korrigiert werden.

In Fallstudien zu Email-Spam-Filterung, HIV-Therapieempfehlung, Zielgruppenmarketing und anderen Anwendungen werden die neuen Modelle mit Referenzmethoden verglichen.

Acknowledgements

First of all, I would like to thank my supervisor Tobias Scheffer for his invaluable support and guidance. He was a great source of inspiration and motivation. I gratefully acknowledge funding from the German Science Foundation DFG. I thank Michael Brückner and Jasmina Bogojeska for the fruitful collaboration. I owe my basic biomedical knowledge to Jasmina. Sincere thanks to Thomas Lengauer who gave me the opportunity to evaluate my algorithms on data from HIV treatment histories collected for the EuResist project. I learned a lot from our industry collaborations with Strato AG and nugg.ad AG. This collaboration motivated our work on spam filtering and targeted advertising. It was inspiring to supervise the diploma thesis of Barbara Pogorzelska and the student project of Christoph Sawade. Under my supervision Barbara elaborated my ideas for the nested Gaussian process model.

I am grateful to Michael Brückner, Laura Dietz, Barbara Pogorzelska, Peter Haider, and Barbara Mai for proofreading (parts of) this thesis. I also thank Matthias Hein for helpful discussions. I greatly appreciate the support of colleagues and student assistants at Humboldt-Universität zu Berlin and at Max-Planck-Institut in Saarbrücken: Ulf Brefeld, Isabel Drost, Michael Brückner, Laura Dietz, Peter Haider, Uwe Dick, Sascha Schulz, Rolf Schimpfky, Barbara Pogorzelska, Thoralf Klein, and Christoph Sawade. I enjoyed numerous discussions with them. Our Christmas workshops always were a lot of fun and gave me the opportunity to improve my snowboarding skills and to collect some bruises.

Last but not least, I owe a lot to the love and support of my parents.

Contents

1 Introduction

One of the core problems in machine learning is to infer predictions for output variables given input variables. In order to learn the relationship between inputs and outputs and thereby learn a predictor, pairs of input and output variables are required—referred to as training data. The inputs for which the output is unknown and predictions are sought are called test data.

Intuitively, the predictions for the test data can only be successful if the training data are representative for the true but unknown relationship between the inputs and outputs in the test data. Most machine learning methods reflect this intuition by assuming that training and test data are governed by the same probability distribution over input-output pairs. If this assumption is violated many common theoretical guarantees and bounds on the error of the predictions are no longer valid and standard models are not expected to work well in practice.

In the statistics and econometrics communities the earliest work on learning from differing training and test distributions date back to the second half of the last century. Good (1965) explores hierarchical Bayesian models to share training data across related learning tasks. In his Nobel Prize winning work, Heckman (1979) studies linear regression models for non-randomly selected training data. In recent years, the machine learning community became more and more aware that the assumption of identical training and test distributions is often violated in practice and started to advance this field.

In this thesis we continue prior research in machine learning and statistics and study and develop learning mechanisms that account for the divergence between the distributions of training and test data.

1.1 Differing Distributions: Motivating Examples

Before we begin with the more technical parts of this thesis, we want give the reader examples of application scenarios in which the assumption of identical training and test distributions is violated. We will empirically study some of the these applications in

the following chapters.

Training on Publicly Available Standard Corpora

Training data collected from publicly available standard corpora might not reflect application specific distributions. For example, part-of-speech taggers trained on standard newspaper corpora might not achieve good results on topics uncommon in newspapers, such as poetry or scientific topics. Machine translation systems are usually trained on parallel corpora. An abundant source of such parallel texts are documents published by the EU commission but training on these texts will bias the translation models towards government and EU specific topics. Similarly, email spam filters trained on publicly available data sets might not work well on user specific email distributions. We study this application in Section 3.8.2.

Similar Learning Tasks with Scarce Training Data

In many applications prediction models for several related tasks are to be learned and the training data for most tasks are scarce. For example, in collaborative filtering for movie recommendation, each user has its user-specific preference function. For a large fraction of users there are only a few preference labels available and labeled data from other similar users might be considered as substitute.

Learning to predict the effect of combinations of drugs from past treatments can be difficult when some drug combinations have been rarely used in the past. In this case one might be tempted to share training data across similar drug combinations (cf. Section 4.5).

Advertising campaigns are usually targeted to specific market segments that can be characterized by sociodemographic features like age, gender, or marital status. For advertising campaigns on the web it is desirable to predict sociodemographic features of web users based on their surfing behavior in order to directly deliver ads to the right users. For small web portals the number of training examples for a predictor is scarce and as a remedy one could share training data across similar web portals. But usually the distribution over web surfing behavior and sociodemographic features is not identical across portals. We study this setting in Section 5.3.

Collecting Data from Test Distribution is Expensive or in Some Other Way Difficult

Labeled data from the test distribution are often difficult to obtain, very expensive, or not accessible at all at training time but labeled data from similar data sources are cheaply available. In landmine detection from radar images there can be no labeled training data for a new geographic region because the collection of labeled data is very dangerous and time consuming. Training data from another geographic region might be available but different from the test conditions in the new geographic region because of different lighting, vegetation, camera, or soil conditions. We study landmine detection in Section 3.8.4.

In predictive modeling for drug screening, the collection of labeled data involves the administration of a drugs to human beings. There might be drugs included in such a prospective clinical study that have a high prior probability of not being medicative for the disease at hand and risk the death of the patient. In this case, ethical concerns may prevent the collection of a large number of training data. Collection of training data from similar drugs, from other species, or from in-vitro studies is usually much easier but exhibit the drawback that the training distribution is different from the test distribution.

Collection of Training Data is Exposed to Selection Process

In some cases the collection of training data involves some selection or rejection process that leads to diverging training and test distributions. For example, in credit scoring the labeled training data is based on the customers that were granted a loan in the past and are therefore exposed to a selection process. The test data are all customers that ask for a loan.

If the target population (the test data) of some prediction problem are all citizens of a country and the training data is collected by a survey, for example, an internet survey, parts of the population are underrepresented because some do not use the internet at all or use it less frequently than other equal sized parts of the population. The training distribution might even more diverge from the test distribution if questionnaires get ignored or refused by some potential participants. We study learning from web surveys for targeted advertising in Section 5.3.

In clinical studies the effect of drugs on patients with a specific disease is analyzed and modeled. The selection of patients for a study might not be random and

therefore not representative for all patients with the disease. For example, if there is a financial compensation for the participation in the study the results can be biased towards patients with lower income and lower overall health conditions or doctors might select patients that they like or dislike. In some studies only men are selected because women can get pregnant and drop out of the study.

In active learning procedures the attention of a labeler is directed towards examples whose label is believed to convey a maximum of information in order to minimize labeling costs. The resulting training data is governed by a selection process, biased towards difficult examples.

Test Distribution Changes Over Time

Under topic drift the data distribution changes over time. If the training data is collected before the test data, the training data may be out-dated and not representative for the distribution of the test data. A typical example is learning recommendations, e.g., for movies or books, when the preference of a user for certain genres, topics, actors, or authors changes over time. We study text classification over time-dependent training and test data in Section 3.8.3.

Test Distribution is Purposefully Altered

In some applications the test distribution is purposefully altered by adversaries in reaction to prediction models. A typical example is spam filtering where spammers try to construct spam emails in a way that they get through the spam filter. A filter directly trained from past spam and non-spam emails does not reflect the distribution of future altered input to the filter. Similarly, in network intrusion detection the training data capture past intrusions but future intruders will try to find new ways to get access to the network.

Taxonomy Classification

Taxonomy classification is the task of assigning objects to nodes of a forest-structured graph, for example, the assignment of webpages into a hierarchy of topics or the assignment of words into a hierarchy of word senses for word sense disambiguation. The edges in the graph encode topical similarities between nodes. For learning of a taxonomy classifier training data or some other information on the node models can be shared across similar nodes. This sharing is essential

for nodes with only a few labeled training examples but the different data distributions of nodes need to be accounted for. A theoretical study of taxonomy classification can be found in Section 4.2.2.

1.2 Contributions

The contributions of this thesis are spread across Chapters 3 to 5. In the following we point out the contributions for each chapter separately. In Chapter 3 on **Learning under Covariate Shift** the contributions are:

- We derive a discriminative expression for rescaling weights that match the rescaled training distribution to the test distribution. With this expression density ratios can be directly estimated without estimating the densities in the numerator and denominator of the ratio separately. The latter procedure is prevalent in the literature (Shimodaira, 2000; Sugiyama and Müller, 2005) but unnecessarily difficult because only the density ratio and not the separate densities are required. Estimating separate densities of potentially high dimensional distributions is prone to estimation errors.

- We formulate the search for the parameters of the discriminative covariate shift model and the parameters of the target model as one integrated optimization problem. This complements the predominant procedure of first estimating the covariate shift of the training sample, and then learning the predictive model on a weighted version of the training sample.

- We derive a primal and kernelized variant of the integrated optimization problem and provide Newton gradient descent updates.

- A convexity analysis reveals that the integrated optimization problem can be convex, depending on the model type; it is convex for the exponential loss.

- A two-stage approximation to the integrated problem leads to a conceptually simple procedure that can be used to modify almost any standard predictive model to account for covariate shift in the training data. The two-stage approximation has the additional advantage that the optimization problems are convex for any convex loss function.

- We derive a new interpretation for the existing kernel mean matching procedure (Huang et al., 2007) that shows its relationship to our model. Until today, it has

been unknown how to tune model parameters of a kernel mean matching model. Our findings lead to an out-of-sample extension of kernel mean matching that can be used for parameter tuning by cross-validation.

- Empirical results on spam filtering, text classification, and landmine detection show that the integrated discriminative model significantly outperforms the *iid* baseline and separate kernel density estimation. The performance of the two-stage approximation is comparable to the integrated model.

- To our knowledge this is the first empirical study on learning under covariate shift that uses real data with a natural divergence between training and test distribution. All previous studies artificially introduce covariate shift into standard data sets.

The contributions in Chapter 4 on **Multi-Task Learning** are:

- We derive a new model for multi-task learning that is based on rescaling weights which match the mixture distribution over all tasks to the distribution of a target task. The model is different from all existing multi-task models because no assumption on the relationship between tasks is required.

- We show that the rescaling weights can be formulated as a discriminative expression and can be estimated with logistic regression. Prior knowledge on task similarity can be encoded in a Gaussian prior on the model parameters. Once the rescaling weights are estimated, a target model over reweighted data is trained.

- We derive a new nested hierarchical Bayesian model for Gaussian processes. The model can be applied to multi-task settings with grouped tasks.

- We show that a popular multi-task model (Evgeniou and Pontil, 2004) becomes a hierarchical Bayesian model by replacing the hinge loss with the logistic or squared loss function.

- We point out that taxonomy classification is related to multi-task learning and we show that a standard model for taxonomy classification with conditional random fields is equivalent to a nested hierarchical Bayesian model. This finding paves the way for new models for taxonomy classification.

- We conduct a case study on HIV therapy screening and show that multi-task learning by distribution matching outperforms two *iid* baselines and two hierarchical Bayesian models in most of the cases.

In Chapter 5 on **Multi-Task Learning under Covariate Shift** the contributions are:

- We introduce a new problem setting that is a combination of multi-task learning and covariate shift. The problem setting naturally arises in a multi-task setting if the labeled training data for each task are collected by surveys.
- We derive a solution for the new problem setting based on rescaling weights that match the mixture distribution over all tasks to the distribution of the test data for a specific target task.
- We reformulate the rescaling weights as a product of two discriminative expressions that can be easily estimated with two logistic regression models. The final predictive model for a specific target task is trained over reweighted training data from all tasks.
- We conduct a case study on targeted advertising and observe that the distribution matching method outperforms reference methods in almost all cases.

1.3 Own Previously Published Work

Parts of this thesis have been previously published or are accepted for publication. This section provides a list of these publications together with attributions of contributions to authors. Chapter 3 of this thesis is partially based on [1] and [2]. Parts of Chapter 4 have been published in [3] and Chapter 5 comprises [4]. Publications [5],[6], and [7] are closely related but are not covered in this thesis.

[1] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

The project started with my derivation of the discriminative expression for rescaling weights. Michael Brückner, Tobias Scheffer, and I formulated the integrated model for learning under covariate shift. Michael derived and implemented the Newton updates for the integrated logistic regression. I conducted the experiments.

[2] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning under covariate shift with a single optimization problem. In J. Quinonero Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors, *Dataset Shift in Machine Learning*. MIT Press, Cambridge, 2008.

For this paper I generalized the integrated model of [1] to arbitrary loss functions, extended the implementation for the exponential loss function, and I conducted the new experiments. I and Michael Brückner proved the conditions for convexity of the generalized model.

[3] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for HIV therapy screening. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

This paper originated from our cooperation with the computational biology group of Thomas Lengauer. Thomas Lengauer and Jasmina Bogojeska contributed to the project with their sound biomedical expertise. I derived and implemented the models and I conducted the experiments. Jasmina prepared the data sets and derived and implemented the mutation table kernel.

[4] S. Bickel, C. Sawade, and T. Scheffer. Transfer learning by distribution matching for targeted advertising. In *Advances in Neural Information Processing Systems (NIPS)*, 2009, to appear.

The idea for this paper arose out of the cooperation with our industry partner nugg.ad AG. I derived and implemented the models and conducted the experiments. Christoph Sawade prepared the data sets and I together with him developed the evaluation methodology.

The following publications did not find their way into this thesis but are closely related.

[5] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

In this publication we study learning under sample selection bias in a multi-task setting. The setting is motivated by the problem of filtering spam messages for many users, given one common labeled training set collected from publicly available sources and unlabeled inboxes from the individual users.

[6] J. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. In *Proceedings of the SIAM International Conference on Data Mining*, 2008.

In this paper we devise a log-linear extension to the KLIEP model of Sugiyama et al. (2008a). The extension has computational advantages and permits learning under covariate shift with very large data sets.

[7] S. Bickel, editor. *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.

I organized the ECML-PKDD Discovery Challenge and the Discovery Challenge workshop in conjunction with the European Conference on Machine Learning 2006. The competition was about personalized spam filtering and generalization across related learning tasks. The problem setting is an instance of learning under differing training and test distributions.

1.4 Outline

The outline of this thesis is as follows. In Chapter 2 we start with basic principles of predictive modeling, loss minimization, and the *iid* assumption. The following three chapters each begin with the definition of a problem setting, proceed with solutions, and empirical studies. Chapter 3 covers learning under covariate shift and centers around discriminative models to capture the discrepancy between training and test distributions. A distribution matching approach to multi-task learning is derived in Chapter 4 in addition to new insights on hierarchical Bayesian modeling. The problem of multi-task learning in combination with covariate shift is discussed in Chapter 5 and a solution based on distribution matching is derived. Chapter 6 concludes the thesis.

2 Learning Predictive Models from Data

The minimization of loss functions over theoretical and empirical distributions plays a key role in this thesis. This chapter gives a short introduction to learning predictive functions from data by loss minimization (Sections 2.1 and 2.2). The *iid* assumption in predictive modeling and its violation in the covariate shift and multi-task settings is outlined in Sections 2.2 and 2.3.

2.1 Predictive Modeling by Loss Minimization

Predictive modeling is typically the task of finding a continuous function $f(\mathbf{x})$ of an input vector $\mathbf{x}$. For binary classification with output labels $y \in \{1, -1\}$ predictions y^* for an input $\mathbf{x}$ can be obtained by thresholding the function,

$$y^* = \text{sign}(f(\mathbf{x})).$$

In order to find a good function $f(\mathbf{x})$ a goodness criterion is needed. As such a criterion one can use a loss function $\ell(f(\mathbf{x}), y)$ that returns a small value if the output of $f(\mathbf{x})$ leads to the correct prediction y and a large value if the output is wrong. The task of learning amounts to minimizing the loss $\ell(f(\mathbf{x}), y)$ over all possible inputs $\mathbf{x}$ and outputs y,

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|\lambda)}[\ell(f(\mathbf{x}), y)] = \int\int \ell(f(\mathbf{x}), y) p(\mathbf{x}, y|\lambda) d\mathbf{x} dy, \tag{2.1}$$

that is the expectation over the probability density $p(\mathbf{x}, y|\lambda)$. λ parameterizes the joint density over inputs and outputs.

A natural loss function for classification is the zero-one loss,

$$\ell_{0/1}(f(\mathbf{x}), y)) = \begin{cases} 1 & \text{if } \text{sign}(f(\mathbf{x})) = y, \\ 0 & \text{otherwise.} \end{cases}$$

The expectation over the zero-one loss is the probability of assigning $\mathbf{x}$ to the wrong class. Unfortunately, minimizing the zero-one loss is difficult because it is not convex and not continuous in the parameters of f.

Convex and continuous loss functions exhibit more favorable properties in optimization. The most common choices for classification are the logistic, hinge, and exponential loss function. The logistic loss function has the form

$$\ell_{\log}(f(\mathbf{x}), y)) = \log(1 + \exp(-yf(\mathbf{x}))),$$

and leads to a logistic regression model. Support vector machines are based on the hinge loss,

$$\ell_{\text{hinge}}(f(\mathbf{x}), y)) = \max(0, 1 - yf(\mathbf{x})),$$

and the exponential loss is given by

$$\ell_{\exp}(f(\mathbf{x}), y)) = \exp(-yf(\mathbf{x})).$$

For regression with continuous output variables the linear function can be directly applied for predictions y^* given an input $\mathbf{x}$,

$$y^* = f(\mathbf{x}).$$

The most common loss function for regression is the squared loss function,

$$\ell_{\text{squared}}(f(\mathbf{x}), y)) = (y - f(\mathbf{x}))^2.$$

2.2 Empirical Regularized Loss and IID Assumption

In the previous section we pointed out that in order to learn a function $f(\mathbf{x})$ it is desirable to minimize the expected loss of Equation 2.1. In practice the theoretical distribution $p(\mathbf{x}, y|\lambda)$ in Equation 2.1 is unknown and is replaced by the empirical distribution over a sample L drawn from $p(\mathbf{x}, y|\lambda)$. This leads to the objective of minimizing the empirical regularized loss in Equation 2.2.

$$\mathbf{E}_{(\mathbf{x},y)\sim L}[\ell(f(\mathbf{x}), y)] + \frac{\Omega(f)}{2\sigma^2} \quad = \quad \frac{1}{|L|} \sum_{(\mathbf{x},y)\in L} \ell(f(\mathbf{x}), y) + \frac{\Omega(f)}{2\sigma^2} \tag{2.2}$$

The additional regularization term restricts the complexity of the function f. The influence of the regularizer is controlled by parameter σ^2. Usually, the regularizer improves

the generalization performance of the predictor and helps to avoid ill-posed optimization problems. In some cases the empirical loss without regularizer is convex but not strictly convex and if $\Omega(f)$ is strictly convex, Equation 2.2 is also strictly convex in the parameters of f. The most common choice for $\Omega(f)$ is the L2-norm of the parameters of f. For an in-depth introduction and a more theoretical discussion of loss functions, generalization error, and regularization the reader is referred to Schölkopf and Smola (2002) and Herbrich (2002).

Once the prediction model is learned by minimizing Equation 2.2, predictions for test data can be obtained as outlined in the previous section. A common assumption in predictive modeling is that the training and test data are *independently* and *identically distributed* (iid). This means that each single training and test instance is drawn from the identical distribution $p(\mathbf{x}, y|\lambda)$ and that training and test instances are conditionally independent given the parameters λ of the joint distribution $p(\mathbf{x}, y|\lambda)$.

If the training data is governed by $p(\mathbf{x}, y|\lambda)$ the minimum loss on the test data can only be achieved in general if it is also governed by the same distribution $p(\mathbf{x}, y|\lambda)$.

2.3 Violation of IID Assumption: Covariate Shift and Multi-Task Setting

The *iid* assumption mentioned in the previous section is violated when training and test distributions are different. Let us assume that the training data is drawn from $p(\mathbf{x}, y|\lambda)$ and the test data is drawn from a different distribution $p(\mathbf{x}, y|\theta)$. In this case the minimizer of the expected training loss does not in general minimize the expected test loss,

$$\operatorname*{argmin}_f \mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|\lambda)}[\ell(f(\mathbf{x}), y)] \neq \operatorname*{argmin}_f \mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|\theta)}[\ell(f(\mathbf{x}), y)]. \tag{2.3}$$

Many theoretical bounds and guarantees for the test error of predictive models are based on the *iid* assumption. If we learn a predictive model by minimizing the loss over the training data drawn from $p(\mathbf{x}, y|\lambda)$ this model is not guaranteed to yield a small error on the test data governed by $p(\mathbf{x}, y|\theta)$.

In order to distinguish different assumptions on the relationship between $p(\mathbf{x}, y|\lambda)$ and $p(\mathbf{x}, y|\theta)$ we factorize training and test density into marginal and conditional densities

as in Equations 2.4 and 2.5.

$$p(\mathbf{x}, y|\lambda) = p(\mathbf{x}|\lambda)p(y|\mathbf{x}, \lambda) \tag{2.4}$$

$$p(\mathbf{x}, y|\theta) = p(\mathbf{x}|\theta)p(y|\mathbf{x}, \theta) \tag{2.5}$$

We can split up the *iid* assumption into an assumption on the marginal densities (Equation 2.6) and an assumption on the conditional densities (Equation 2.7).

$$p(\mathbf{x}|\lambda) \triangleq p(\mathbf{x}|\theta) \tag{2.6}$$

$$p(y|\mathbf{x}, \lambda) \triangleq p(y|\mathbf{x}, \theta) \tag{2.7}$$

In the *covariate shift* setting, Assumption 2.6 can be violated but 2.7 holds. This means, the input marginals—also known as covariate densities—can be different but training and test data share the same conditional density.

In the more general *multi-task* setting, Assumption 2.6 as well as 2.7 can be violated. In multi-task learning there are several potentially related learning tasks each with a distinct joint input-output distribution. For each task, training data as well as test data may be available governed by the identical task specific distribution. From the point of view of one specific task, the test data from this task may governed by a joint distribution different from the training distributions from all other tasks. The challenge is to effectively share the training data across tasks. The detailed problem settings of learning under covariate shift and multi-task learning are introduced in Sections 3 and 4.1.

3 Learning under Covariate Shift

In some applications the difference between the training and test distributions is only reflected in the distribution over the inputs, in statistics also known as covariates of a prediction problem. Such a covariate shift has the appealing property that for an estimate and compensation of the shift only *unlabeled* test data is required (in addition to the training data). In many cases unlabeled test data is abundant in contrast to labeled data from the test distribution.

In the *covariate shift* problem setting, a training sample is available in matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. This training sample is governed by an unknown distribution $p(\mathbf{x}|\lambda)$. Vector $\mathbf{y}$ with elements $y_1, \ldots, y_m$ are the labels for training examples and are drawn according to an unknown target concept $p(y|\mathbf{x})$. In addition, unlabeled test data becomes available in matrix $\mathbf{X}_T$ with rows $\mathbf{x}_{m+1}, \ldots, \mathbf{x}_{m+n}$. The test data is governed by a different unknown distribution, $p(\mathbf{x}|\theta)$. Training and test distribution may differ arbitrarily, but there is only one unknown target conditional class distribution $p(y|\mathbf{x})$.

A typical cause of a pure covariate shift is a selection process that has *only* access to unlabeled test data. For example, the labeled data for a prediction problem is collected through a clinical study and beforehand doctors select patients for the study based on specific input features. The selection process causes the training distribution to be different from the test distribution. At the time of the selection the outcome (the labels) of the study is unknown to the doctors. The selection and therefore the shift can only be reflected in the input features not the output labels.

In some cases the type of shift is unknown (covariate shift or joint input-output shift) but no labeled data, only unlabeled data from the test and labeled data from the training distribution are available. Even if one knows that there is a joint input-output shift in the data, there is no way to estimate the joint shift because no labeled test data is available. In such a setting one could assume covariate shift and apply a covariate shift model, in order to at least account for this part of the shift. The empirical studies at the end of this chapter show that this procedure can lead to improved prediction performance compared to an *iid* assumption of training and test data.

In this chapter we develop a new framework for learning under covariate shift and

study the relationship to existing methods both theoretically and empirically. The outline of the chapter is as follows. We clarify the relationship between covariate shift, sample selection bias, and propensity scoring in Section 3.1. The prevalent reweighting procedure for learning under covariate shift is reviewed in Section 3.2. In Section 3.3 we derive our discriminative framework for learning under covariate shift and instantiate several new algorithms. The relationship to other direct density ratio estimators is discussed in Section 3.6. In this context we derive a new out-of-sample extension to the existing kernel mean matching model. We discuss parameter tuning procedures for covariate shift models in Section 3.7. An empirical study on spam filtering, text classification, and landmine detection is presented in Section 3.8. Section 3.9 concludes this chapter.

3.1 Covariate Shift vs. Sample Selection Bias and Propensity Scoring

The problem of learning under sample selection bias is very similar to learning under covariate shift. Propensity scoring can be seen as an extension to learning under sample selection bias. In this section we will clarify the relationships between covariate shift, sample selection bias, and propensity scoring.

3.1.1 Sample Selection Bias

A line of work on learning under sample selection bias has meandered from the statistics and econometrics community into machine learning (Heckman, 1979; Zadrozny, 2004). Sample selection bias relies on a model of the data generation process. Test instances are drawn under $p(\mathbf{x}|\theta)$. Training instances are drawn by first sampling $\mathbf{x}$ from the test distribution $p(\mathbf{x}|\theta)$. A binary selector variable[1] s' then decides whether $\mathbf{x}$ is moved into the training set ($s' = 1$) or moved into the rejected set ($s' = -1$). For instances in the training set ($s' = 1$) a label is drawn from $p(y|\mathbf{x})$, for the instances in the rejected set the labels are unknown. A typical scenario for sample selection bias is credit scoring. The labeled training sample consists of customers who where given a loan in the past and the rejected sample are customers that asked for but where not given a loan. New customers asking for a loan reflect the test distribution.

In the so-called *missing at random* case, the selector variable is only dependent on

[1]The prime symbol distinguishes s' from the variable s, that is used later in this thesis with a slightly different meaning.

$\mathbf{x}$, but not on y; that is, $p(s' = 1|\mathbf{x}, y, \theta, \lambda) = p(s' = 1|\mathbf{x}, \theta, \lambda)$. The distribution of the selector variable then maps the test onto the training distribution:

$$p(\mathbf{x}|\lambda) \propto p(\mathbf{x}|\theta)p(s' = 1|\mathbf{x}, \theta, \lambda). \tag{3.1}$$

Proposition 3.1 (Zadrozny, 2004; Bickel and Scheffer, 2007) says that minimizing the loss on instances weighted by $p(s'|\mathbf{x}, \theta, \lambda)^{-1}$ in fact minimizes the expected loss with respect to θ.

Proposition 3.1 *The expected loss with respect to θ is proportional to the weighted expected loss with respect to λ with weights $p(s' = 1|\mathbf{x}, \theta, \lambda)^{-1}$ for the loss incurred by each $\mathbf{x}$, provided that the support of $p(\mathbf{x}|\theta)$ is equal to the support of $p(\mathbf{x}|\lambda)$.*

$$\mathbf{E}_{(\mathbf{x},y)\sim\theta}[\ell(f(\mathbf{x}), y)] \quad \propto \quad \mathbf{E}_{(\mathbf{x},y)\sim\lambda}\left[\frac{1}{p(s' = 1|\mathbf{x}, \theta, \lambda)}\ell(f(\mathbf{x}), y)\right]. \tag{3.2}$$

When the model is implemented, $p(s' = 1|\mathbf{x}, \theta, \lambda)$ is learned by discriminating the training against the rejected examples; in a second step the target model is learned by following Proposition 3.1 and weighting training examples by $p(s'|\mathbf{x}, \theta, \lambda)^{-1}$. No test examples drawn directly from $p(\mathbf{x}|\theta)$ are needed to train the model, only labeled selected and unlabeled rejected examples are required. This is in contrast to the covariate shift model that requires samples drawn from the test distribution, but no selection process is assumed and no rejected examples are needed. Covariate shift models can be applied to learning under sample selection bias in the missing at random setting by treating the selected examples as labeled sample L and the union of selected (ignoring the labels) and rejected examples as unlabeled sample T.

Some authors study learning under sample selection bias in a setting where instead of unlabeled rejected examples only examples directly drawn from the test distribution $p(\mathbf{x}|\theta)$ are available (Smith and Elkan, 2007; Hein, 2008). With a missing at random assumption this is almost identical to the covariate shift problem setting and the same models can be applied. The only marginal difference lies in the relationship between the support of the distributions over labeled and unlabeled data. In sample selection bias one usually assumes that if $p(\mathbf{x}|\theta) > 0$ also $p(s' = 1|\mathbf{x}, \theta, \lambda) > 0$ and therefore the support of the training distribution is *identical* to the test distribution. In learning under covariate shift (see also Section 3.2) the usual assumption is that the support of the test distribution is *contained* in the support of the training distribution. In this sense learning under covariate shift is more general than learning under sample selection bias (in the missing at random case).

Smith and Elkan (2004) systematically analyze different dependency assumptions for sample selection bias. They study selection processes dependent on only observed features (missing at random case, described above), only unobserved features, only output labels, and arbitrary combinations of these. Elkan (2001) and Japkowicz and Stephen (2002) investigate the case of training data that is only biased with respect to the class ratio when the true ratio is known. This can be seen as sample selection bias where the selection only depends on the output labels y.

Maximum entropy density estimation under sample selection bias has been studied by Dudik et al. (2005). Bickel and Scheffer (2007) impose a Dirichlet process prior on several learning problems with related sample selection bias. Cortes et al. (2008) theoretically analyze the error that gets introduced by estimating sample selection bias from data. Their analysis covers the kernel mean matching procedure and a cluster-based estimation technique.

3.1.2 Propensity Scoring

Propensity scores (Rosenbaum and Rubin, 1983; Lunceford and Davidian, 2004) are applied in settings related to sample selection bias; the training data is again assumed to be drawn from the test distribution $p(\mathbf{x}|\theta)$ followed by a selection process. The difference to the setting of sample selection bias is that the selected *and* the rejected examples are labeled. Weighting the selected examples by the inverse of the selection probability $p(s' = 1|\mathbf{x}, \lambda, \theta)^{-1}$ and weighting the rejected examples by the inverse of the rejection probability $p(s' = -1|\mathbf{x}, \lambda, \theta)^{-1}$ results in two unbiased samples with respect to the test distribution $p(\mathbf{x}|\theta)$. Propensity scoring is equivalent to applying the weighting of Equation 3.2 twice, once for the selected, and once for the rejected data. Propensity score is another term for the sample selection probability $p(s' = 1|\mathbf{x}, \lambda, \theta)$ described in Section 3.1.1; likewise, it can be estimated by discriminating the selected against the rejected examples.

Propensity scoring can precede a variety of analysis steps. This can be a statistical analysis of the two reweighted samples or the training of a target model on reweighted data. A typical application for propensity scores is the analysis of the success of a medical treatment. Patients are selected to be given the treatment and some other patients are selected into the control group (rejected). If the selector variable is not independent of $\mathbf{x}$ (patients may be chosen for an experimental therapy only if they meet specific requirements), the outcome (*e.g.,* ratio of cured patients) of the two groups cannot be compared directly, propensity scores have to be applied.

3.2 Compensation of Covariate Shift by Loss Rescaling

In a covariate shift setting, if training and test distributions were known, then the loss on the test distribution could be minimized by weighting the loss on the training distribution with an instance-specific factor. Proposition 3.2 (Shimodaira, 2000) illustrates that the scaling factor has to be $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$.

Proposition 3.2 *The expected loss with respect to θ equals the weighted expected loss with respect to λ with weights $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ for the loss incurred by each $\mathbf{x}$, provided that the support of $p(\mathbf{x}|\theta)$ is contained in the support of $p(\mathbf{x}|\lambda)$:*

$$\mathbf{E}_{(\mathbf{x},y)\sim\theta}[\ell(f(\mathbf{x}),y)] \quad = \quad \mathbf{E}_{(\mathbf{x},y)\sim\lambda}\left[\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}\ell(f(\mathbf{x}),y)\right]. \tag{3.3}$$

After expanding the expected value into its integral $\int \ell(f(\mathbf{x}),y)p(\mathbf{x},y|\theta)d\theta$, the joint distribution $p(\mathbf{x},y|\lambda)$ is decomposed into $p(\mathbf{x}|\lambda)p(y|\mathbf{x},\lambda)$. Since $p(y|\mathbf{x},\lambda) = p(y|\mathbf{x}) = p(y|\mathbf{x},\theta)$ is the global conditional distribution of the class variable given the instance, Proposition 3.2 follows. All instances $\mathbf{x}$ with positive $p(\mathbf{x}|\theta)$ are integrated over. Hence, Equation 3.3 holds as long as each $\mathbf{x}$ with positive $p(\mathbf{x}|\theta)$ also has a positive $p(\mathbf{x}|\lambda)$; otherwise, the denominator vanishes. This shows that covariate shift can only be compensated for as long as the training distribution covers the entire support of the test distribution. If a test instance had zero density under the training distribution, the test-to-training density ratio which it would need to be scaled with would incur a zero denominator.

Shimodaira (2000) studies asymptotic properties of learning probabilistic models under covariate shift. He proves that learning with weighted data by minimizing the right hand side of Equation 3.3 improves the prediction performance over training with uniform weights only if the model is misspecified. In this case the estimation bias of the model is reduced by weighting the training data with the density ratio.

In Shimodaira's analysis the predictive function $f(\mathbf{x})$ is represented by a conditional model $p(y|\mathbf{x},\mathbf{w})$ with model parameters $\mathbf{w}$. If the model is well-specified there exists a $\mathbf{w}^*$ for which the model $p(y|\mathbf{x},\mathbf{w}^*)$ exactly matches the true conditional $p(y|\mathbf{x})$, thus $p(y|\mathbf{x},\mathbf{w}^*) = p(y|\mathbf{x})$. In this case, even if the training distribution is exposed to covariate shift, the expected error is minimized by training with uniformly weighted data. Because these findings are based on asymptotic approximations they do not necessarily carry over to practical settings with a small number of training examples. An asymptotic analysis by Sokolovska et al. (2008) shows that methods for learning under covariate shift can even improve the performance of models in regular semi-supervised

learning without covariate shift if the model is misspecified.

In practice a classifier $f(\mathbf{x})$ is trained by minimizing the expected weighted loss (right hand side of Equation 3.3) on a sample L drawn from the training distribution $p(\mathbf{x}|\theta)$ reweighted by estimates for $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$. In general, applying non-uniform weights to training data (some of which may even be zero) reduces the effective sample size. This leads to a bias-variance trade-off in learning under covariate shift (Shimodaira, 2000): training on unweighted data causes an estimation bias if the model is misspecified, applying non-uniform weights reduces the effective sample size and therefore increases the variance of the estimator. This trade-off can be controlled by using a smoothed rescaling weight $\left(\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}\right)^{\eta}$ with a parameter $\eta \in [0, 1]$. A uniform weighting (higher bias, lower variance) can be obtained by setting $\eta = 0$ and the density ratio without smoothing (lower bias, higher variance) corresponds to $\eta = 1$.

Compensation for covariate shift requires the density ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ (Equation 3.3). Both, $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\lambda)$ are unknown, but $p(\mathbf{x}|\theta)$ is reflected in T, as is $p(\mathbf{x}|\lambda)$ in L. A straightforward approach is to first obtain estimates $\hat{p}(\mathbf{x}|\theta)$ and $\hat{p}(\mathbf{x}|\lambda)$ from the test and training data, respectively, using kernel density estimation (Shimodaira, 2000; Sugiyama and Müller, 2005). In a second step, the estimated density ratio is used to re-sample the training instances, or to train with weighted examples.

This method decouples the problem. First, it estimates training and test distributions. This step is intrinsically model-based and only loosely related to the ultimate goal of accurate classification. In a subsequent step, the classifier is derived given fixed weights. Since the parameters of the final classifier and the parameters that control the weights are not independent, this decomposition into two optimization steps cannot generally find the optimal setting of the *joint* parameter vector.

3.3 Discriminative Learning under Covariate Shift

In discriminative learning tasks such as classification, the classifier's goal is to produce the correct output given the input. It is widely accepted that this is best performed by discriminative learners that directly maximize a quality measure of the produced output. Model-based optimization criteria such as the joint likelihood of input and output, by contrast, additionally assess how well the classifier models the distribution of input values. This amounts to adding a term to the criterion that is irrelevant for the task at hand.

In this section we derive a discriminative model for learning under different training and test distributions. The model directly characterizes the divergence between train-

ing and test distribution, without the intermediate—intrinsically model-based—step of estimating training and test distribution separately. We formulate the search for all model parameters as an integrated optimization problem. This complements the predominant procedure of first estimating the bias of the training sample, and then learning the classifier on a weighted version of the training sample. We show that the integrated optimization can be convex, depending on the model type; it is convex for the exponential model. We derive a Newton gradient descent procedure, leading to a kernel logistic regression and an exponential model classifier for covariate shift.

Section 3.3.1 describes the generative modeling assumption of the integrated model. In Section 3.3.2 we outline a *maximum a posteriori* estimation procedure for the model parameters. Section 3.3.3 depicts the discriminative reformulation of the test-to-training ratio and defines the label likelihood. Section 3.3.4 describes the integrated optimization problem. We derive primal and kernelized classifiers for differing training and test distributions in Section 3.3.5. In Section 3.4, we analyze the convexity of the integrated optimization problem. Section 3.5 describes a two-stage approximation that allows to train virtually any type of classifier under covariate shift.

3.3.1 Integrated Model

Our goal is to find model parameters $\mathbf{w}$ for a probabilistic classification model $f(\mathbf{x}) = \operatorname{argmax}_y p(y|\mathbf{x};\mathbf{w})$. The model should correctly predict labels of the test data $\mathbf{X}_T$ drawn from $p(\mathbf{x}|\theta)$. A regular *maximum a posteriori* estimation $\mathbf{w}' = \operatorname{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}_L;\mathbf{w})p(\mathbf{w})$, would only use the training data $(\mathbf{y},\mathbf{X}_L)$ governed by $p(\mathbf{x}|\lambda)$. By ignoring the test data, this estimate will not generally result in a model that predicts the missing labels of the test data with a minimum error because the training distribution $p(\mathbf{x}|\lambda)$ is different from the test distribution $p(\mathbf{x}|\theta)$.

In the following we devise a probabilistic model that accounts for the difference between training and test distribution. Before we describe the model we define a joint data matrix $\mathbf{X}$ that is a concatenation of the matrices $\mathbf{X}_L$ and $\mathbf{X}_T$. The model is based on a binary selector variable s: Given an instance vector $\mathbf{x}$ from the joint matrix $\mathbf{X}$ of all available instances, selector variable s decides whether $\mathbf{x}$ is drawn into the training data $\mathbf{X}_L$ and $\mathbf{y}$ is determined ($s = 1$) or into the test data $\mathbf{X}_T$ ($s = -1$). The variable s is governed by the distribution $p(s|\mathbf{x};\mathbf{v})$. Parameter $\mathbf{v}$ characterizes the discrepancy between the training and test distribution. Based on the model for s we can now describe the generative process underlying our model:

1. Draw parameter vectors $\mathbf{v}$ and $\mathbf{w}$ from prior distributions $p(\mathbf{v})$ and $p(\mathbf{w})$;

2. For each row $\mathbf{x}$ in matrix $\mathbf{X}$ draw binary variable s from distribution $p(s|\mathbf{x};\mathbf{v})$; accordingly, the likelihood of the vector of all selector variables $\mathbf{s}$ is $p(\mathbf{s}|\mathbf{X};\mathbf{v}) = \prod_{i=1}^{m+n} p(s_i|\mathbf{x}_i;\mathbf{v})$;

3. For all selected training examples (all examples $\mathbf{x}_i$ with $s_i = 1$) draw vector $\mathbf{y}$ of all labels from $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$.

This generative process corresponds to the following factorization of the joint probability of the vector of labels $\mathbf{y}$, vector of selector variables $\mathbf{s}$, and parameter vectors $\mathbf{v}$ and $\mathbf{w}$:

$$p(\mathbf{y},\mathbf{s},\mathbf{w},\mathbf{v}|\mathbf{X}) = p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}). \tag{3.4}$$

3.3.2 Maximum A Posteriori Parameter Inference

For parameter inference we want to find parameters $\mathbf{w}$ that maximize the posterior probability given all available data (Equation 3.5). The available data are the data matrix $\mathbf{X}$, the label vector $\mathbf{y}$, and the selection vector $\mathbf{s}$, that splits the data matrix into training and test data. Because the parameter $\mathbf{v}$ is unknown and is not needed for the final classifier the best we can do is to integrate it out (Equation 3.6).

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X}) \tag{3.5}$$

$$= \operatorname{argmax}_{\mathbf{w}} \int p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X})d\mathbf{v} \tag{3.6}$$

Integrating over $\mathbf{v}$ is computationally infeasible. In Equation 3.7, the integral is therefore approximated by the single assignment of values to the parameters which maximizes the posterior—the *maximum a posteriori* (MAP) estimator. In our case, the MAP estimator naturally assigns values to all parameters, $\mathbf{w}$ and $\mathbf{v}$.

$$(\mathbf{w}_{\text{MAP}},\mathbf{v}_{\text{MAP}}) = \operatorname{argmax}_{\mathbf{w},\mathbf{v}} p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) \tag{3.7}$$

$$= \operatorname{argmax}_{\mathbf{w},\mathbf{v}} p(\mathbf{y},\mathbf{s},\mathbf{w},\mathbf{v}|\mathbf{X}) \tag{3.8}$$

$$= \operatorname{argmax}_{\mathbf{w},\mathbf{v}} p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}) \tag{3.9}$$

Equation 3.8 follows from multiplication with a constant $p(\mathbf{y},\mathbf{s}|\mathbf{X})$ and with the chain rule. Equation 3.9 applies the factorization from the generative process of Equation 3.4.

The class-label posterior $p(y|\mathbf{x};\mathbf{w}_{\text{MAP}})$ is conditionally independent of $\mathbf{v}_{\text{MAP}}$ given $\mathbf{w}_{\text{MAP}}$. However, $\mathbf{w}_{\text{MAP}}$ and $\mathbf{v}_{\text{MAP}}$ are dependent. Assigning a single MAP value to $[\mathbf{w},\mathbf{v}]$ instead of integrating over $\mathbf{v}$ is a common approximation. However, sequential max-

imization of $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ over parameters $\mathbf{v}$ followed by maximization of $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$ over parameters $\mathbf{w}$ with fixed $\mathbf{v}$ would amount to an additional degree of approximation and will not generally coincide with the maximum of the product in Equation 3.9.

In the next sections we will discuss the likelihood functions $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$ and $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ and the optimization problem for parameter inference based on maximization of Equation 3.9.

3.3.3 Label Likelihood and Discriminative Weighting Factors

In order to define the label likelihood we first derive a discriminative expression for $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ which will no longer include any density on instances. When $p(s=-1)>0$, which is implied by the test set not being empty, the definition of s allows us to rewrite the test distribution as $p(\mathbf{x}|\theta)=p(\mathbf{x}|s=-1,\theta)$. Since test instances are only dependent on parameter θ but not on parameter λ, equation $p(\mathbf{x}|s=-1,\theta)=p(\mathbf{x}|s=-1,\theta,\lambda)$ follows. By an analogous argument, $p(\mathbf{x}|\lambda)=p(\mathbf{x}|s=1,\theta,\lambda)$ when $p(s=1)>0$. This implies Equation 3.10.

In Equation 3.11, Bayes' rule is applied twice; the two terms of $p(\mathbf{x}|\theta,\lambda)$ cancel each other out in Equation 3.12. Since $p(s=-1|\mathbf{x},\theta,\lambda)=1-p(s=1|\mathbf{x},\theta,\lambda)$, Equation 3.13 follows.

The conditional $p(s=1|\mathbf{x},\theta,\lambda)$ discriminates training ($s=1$) against test instances ($s=-1$).

$$\begin{aligned}
\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)} &= p(\mathbf{x}|s=-1,\theta,\lambda)\frac{1}{p(\mathbf{x}|s=1,\theta,\lambda)} && (3.10)\\
&= \frac{p(s=-1|\mathbf{x},\theta,\lambda)p(\mathbf{x}|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\frac{p(s=1|\theta,\lambda)}{p(s=1|\mathbf{x},\theta,\lambda)p(\mathbf{x}|\theta,\lambda)} && (3.11)\\
&= \frac{p(s=1|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\frac{p(s=-1|\mathbf{x},\theta,\lambda)}{p(s=1|\mathbf{x},\theta,\lambda)} && (3.12)\\
&= \frac{p(s=1|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\left(\frac{1}{p(s=1|\mathbf{x},\theta,\lambda)}-1\right) && (3.13)
\end{aligned}$$

The significance of Equation 3.13 is that it shows how the optimal example weights, the test-to-training ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$, can be determined without knowledge of either training or test density. The right hand side of Equation 3.13 can be evaluated based on a model that discriminates training against test examples and outputs how much more likely an instance is to occur in the test data than it is to occur in the training data. Instead of potentially high-dimensional densities $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\lambda)$, a conditional distribution of the single binary variable s needs to be modeled.

The expression $p(s|\mathbf{x},\theta,\lambda)$ in Equation 3.13 corresponds to the parametric model

$p(s|\mathbf{x};\mathbf{v})$ of Equation 3.4. With this model we can predict test-to-training density ratios for the training data in $\mathbf{X}_L$ according to Equation 3.13.

Since our goal is discriminative training, the likelihood function $p(\mathbf{y}|\mathbf{w},\mathbf{X}_L)$ (not taking training-test difference $\mathbf{v}$ into account) would be $\prod_i p(y_i|\mathbf{x}_i;\mathbf{w})$. Intuitively, $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ dictates how many times, on average, $\mathbf{x}$ should occur in $\mathbf{X}_L$ if $\mathbf{X}_L$ was governed by the test distribution θ. When the individual conditional likelihood of $\mathbf{x}$ is $p(y|\mathbf{x};\mathbf{w})$, then the likelihood of $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ occurrences of $\mathbf{x}$ is $p(y|\mathbf{x};\mathbf{w})^{\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}}$. Using a parametric model $p(s|\mathbf{x};\mathbf{v})$, according to Equation 3.13 the test-to-training ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ can be expressed as [2]

$$\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{p(s=1|\mathbf{x};\mathbf{v})}-1\right).$$

Therefore, we define the likelihood function as

$$p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v}) = \prod_{i=1}^{m} p(y_i|\mathbf{x}_i;\mathbf{w})^{\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{p(s_i=1|\mathbf{x}_i;\mathbf{v})}-1\right)}. \tag{3.14}$$

As an immediate corollary of Manski and Lerman (1977), the likelihood function of Equation 3.14 has the property that when the true value $\mathbf{v}^*$ is given, its maximizer over $\mathbf{w}$ is a consistent estimator of the true parameter $\mathbf{w}^*$ that has produced labels for the test data under the test distribution θ. That is, as the sample grows, the maximizer of Equation 3.14 converges in probability to the true value $\mathbf{w}^*$ of parameter $\mathbf{w}$.

Shortly after we first published the above derivations, Smith and Elkan (2007) derive Equation 3.13 from the perspective of learning under sample selection bias. For the statistical analysis of case-control studies, Prentice and Pyke (1979) estimate the ratio of two odds ratios with a discriminative model using a formula similar to Equation 3.13. This double odds ratio is a statistical measure of the relative risk of an incidence (e.g., lung cancer) given a specific exposure (e.g., cigarette smoking) based on data from a retrospective study.

3.3.4 Optimization Problem for Integrated Model

The likelihood function $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ resolves to $p(s_i=1|\mathbf{x}_i;\mathbf{v})$ for all training instances and $p(s_i=-1|\mathbf{x}_i;\mathbf{v})$ for all test instances:

$$p(\mathbf{s}|\mathbf{X};\mathbf{v}) = \prod_{i=1}^{m} p(s_i=1|\mathbf{x}_i;\mathbf{v}) \prod_{i=m+1}^{m+n} p(s_i=-1|\mathbf{x}_i;\mathbf{v}). \tag{3.15}$$

[2] For a simplified presentation we drop the conditioning in the prior ratio, *i.e.,* $p(s|\theta,\lambda)=p(s)$.

Equation 3.16 summarizes Equations 3.7 to 3.9. Equation 3.17 inserts the likelihood models (Equations 3.14 and 3.15) and draws constants $p(s=1)$ and $p(s=-1)$ out of the product.

$$p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) \propto p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}) \tag{3.16}$$

$$= \left(\prod_{i=1}^{m} p(y|\mathbf{x}_i;\mathbf{w})^{\frac{1}{p(s_i=1|\mathbf{x}_i;\mathbf{v})}-1}\right)^{\frac{p(s=1)}{p(s=-1)}} \left(\prod_{i=1}^{m} p(s_i=1|\mathbf{x}_i;\mathbf{v}) \prod_{i=m+1}^{m+n} p(s_i=-1|\mathbf{x}_i;\mathbf{v})\right) p(\mathbf{w})p(\mathbf{v}) \tag{3.17}$$

Using a logistic model for $p(s=1|\mathbf{x};\mathbf{v})$, we notice that Equation 3.13 can be simplified as in Equation 3.18.

$$\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{1/(1+\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}))}-1\right) = \frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}) \tag{3.18}$$

Optimization Problem 1 is derived from Equation 3.17 in logarithmic form, using linear models $\mathbf{v}^\mathsf{T}\mathbf{x}_i$ and $\mathbf{w}^\mathsf{T}\mathbf{x}_i$ and a logistic model for $p(s=1|\mathbf{x};\mathbf{v})$. Negative log-likelihoods are abbreviated $\ell_\mathbf{w}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i) = -\log p(y_i|\mathbf{x}_i;\mathbf{w})$ and $\ell_\mathbf{v}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i) = -\log p(s_i|\mathbf{x}_i;\mathbf{v})$, respectively; this notation emphasizes the duality between likelihoods and empirical loss functions. The regularization terms correspond to Gaussian priors on $\mathbf{v}$ and $\mathbf{w}$ with variances $\sigma^2_\mathbf{v}$ and $\sigma^2_\mathbf{w}$.

Optimization Problem 1 *Over all* $\mathbf{w}$ *and* $\mathbf{v}$*, minimize*

$$\sum_{i=1}^{m}\frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i)\ell_\mathbf{w}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i) + \sum_{i=1}^{m+n}\ell_\mathbf{v}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i) + \frac{1}{2\sigma^2_\mathbf{w}}\mathbf{w}^\mathsf{T}\mathbf{w} + \frac{1}{2\sigma^2_\mathbf{v}}\mathbf{v}^\mathsf{T}\mathbf{v}.$$

In Section 3.2 we described how the bias-variance trade-off in learning under covariate shift can be controlled by a smoothed density ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}^\eta$ (Shimodaira, 2000). In Optimization Problem 1 the regularization parameter $\sigma^2_\mathbf{v}$ for the weight model plays the role of Shimodaira's smoothing parameter η. The parameter $\sigma^2_\mathbf{v}$ is the variance of a Gaussian prior on $\mathbf{v}$ and therefore regularizes the L2-norm of $\mathbf{v}$. If we apply η to the density ratio model of Equation 3.18 as shown in Equation 3.19 we observe that scaling $\mathbf{v}^\mathsf{T}\mathbf{x}$ by η would have the same effect as regularizing the L2-norm of $\mathbf{v}$.

$$\left(\frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x})\right)^\eta = \left(\frac{p(s=1)}{p(s=-1)}\right)^\eta \exp(-\eta\mathbf{v}^\mathsf{T}\mathbf{x}) \tag{3.19}$$

Empirically, we find that smoothing of the constant prior ratio with η in the first term on the right hand side of Equation 3.19 is not necessary because $\sigma_{\mathbf{v}}^2$ gives us enough control to the smoothness of the weights.

3.3.5 Primal and Kernelized Learning Algorithm

We derive a Newton gradient descent method that directly minimizes Optimization Problem 1 in the attribute space. To this end, we need to derive the gradient and the Hessian of the objective function. The update rule assumes the form of a set of linear equations that have to be solved for the update vector $[\Delta_{\mathbf{v}}, \Delta_{\mathbf{w}}]^\mathsf{T}$. It depends on the current parameters $[\mathbf{v}, \mathbf{w}]^\mathsf{T}$, all combinations of training and test data, and resulting coefficients. In order to express the update rule as a single equation in matrix form, we define

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_L & \mathbf{X}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_L \end{bmatrix}, \tag{3.20}$$

where $\mathbf{X}_L$ and $\mathbf{X}_T$ are the matrices of training vectors and test vectors, respectively.

Theorem 3.1 *The update step for the Newton gradient descent minimization of Optimization Problem 1 is* $[\mathbf{v}', \mathbf{w}']^\mathsf{T} \leftarrow [\mathbf{v}, \mathbf{w}]^\mathsf{T} + [\Delta_{\mathbf{v}}, \Delta_{\mathbf{w}}]^\mathsf{T}$ *with*

$$(\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T} + S) \begin{bmatrix} \Delta_{\mathbf{v}} \\ \Delta_{\mathbf{w}} \end{bmatrix} = -\mathbf{X}\mathbf{g} - S \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}. \tag{3.21}$$

The definitions of coefficients $\mathbf{\Lambda}$, S, and $\mathbf{g}$—and the proof of the theorem—can be found in Appendix A.

Given the parameter $\mathbf{w}$, a test instance $\mathbf{x}$ is classified as $f(\mathbf{x}; \mathbf{w}) = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})$.

We derive a kernelized version of the integrated classifier for differing training and test distributions. A transformation Φ maps instances into a target space in which a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ calculates the inner product $\Phi(\mathbf{x}_i)^\mathsf{T}\Phi(\mathbf{x}_j)$. The update rule (Equation 3.21) thus becomes

$$(\Phi(\mathbf{X})\mathbf{\Lambda}\Phi(\mathbf{X})^\mathsf{T} + S) \begin{bmatrix} \Delta_{\mathbf{v}} \\ \Delta_{\mathbf{w}} \end{bmatrix} = -\Phi(\mathbf{X})\mathbf{g} - S \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}. \tag{3.22}$$

$\Phi(\mathbf{X})$ is defined by

$$\Phi(\mathbf{X}) = \begin{bmatrix} \Phi(\mathbf{X}_L) & \Phi(\mathbf{X}_T) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Phi(\mathbf{X}_L) \end{bmatrix}. \tag{3.23}$$

According to the Representer Theorem, the optimal separator is a linear combination of examples. Parameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the dual space weight the influence of all examples:

$$\begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \Phi(\mathbf{X}) \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}. \tag{3.24}$$

Equation 3.22 can therefore be rewritten as Equation 3.25. We now multiply $\Phi(\mathbf{X})^\mathsf{T}$ from the left to both sides and obtain Equation 3.26. We replace all resulting occurrences of $\Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})$ by the kernel matrix $\mathbf{K}$ and arrive at Equation 3.27; S is replaced by S' such that $\Phi(\mathbf{X})^\mathsf{T} S \Phi(\mathbf{X}) = \Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})S'$, *i.e.,* $S'_{i,i} = \sigma_\mathbf{v}^{-2}$ for $i = 1, \ldots, m+n$ and $S'_{m+n+i,m+n+i} = \sigma_\mathbf{w}^{-2}$ for $i = 1, \ldots, m$. Equation 3.27 is satisfied when Equation 3.28 is satisfied. Equation 3.28 is the update rule for the dual Newton gradient descent.

$$(\Phi(\mathbf{X})\boldsymbol{\Lambda}\Phi(\mathbf{X})^\mathsf{T} + S)\Phi(\mathbf{X}) \begin{bmatrix} \Delta_{\boldsymbol{\alpha}} \\ \Delta_{\boldsymbol{\beta}} \end{bmatrix} = -\Phi(\mathbf{X})\mathbf{g} - S\Phi(\mathbf{X}) \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{3.25}$$

$$\Phi(\mathbf{X})^\mathsf{T}(\Phi(\mathbf{X})\boldsymbol{\Lambda}\Phi(\mathbf{X})^\mathsf{T} + S)\Phi(\mathbf{X}) \begin{bmatrix} \Delta_{\boldsymbol{\alpha}} \\ \Delta_{\boldsymbol{\beta}} \end{bmatrix} = -\Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})\mathbf{g} - \Phi(\mathbf{X})^\mathsf{T} S \Phi(\mathbf{X}) \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{3.26}$$

$$(\mathbf{K}\boldsymbol{\Lambda}\mathbf{K} + \mathbf{K}S') \begin{bmatrix} \Delta_{\boldsymbol{\alpha}} \\ \Delta_{\boldsymbol{\beta}} \end{bmatrix} = -\mathbf{K}\mathbf{g} - \mathbf{K}S' \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{3.27}$$

$$(\boldsymbol{\Lambda}\mathbf{K} + S') \begin{bmatrix} \Delta_{\boldsymbol{\alpha}} \\ \Delta_{\boldsymbol{\beta}} \end{bmatrix} = -\mathbf{g} - S' \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{3.28}$$

Given the parameters, test instance $\mathbf{x}$ is classified by $f(\mathbf{x}; \boldsymbol{\beta}) = \text{sign}(\sum_{i=1}^{m} \beta_i k(\mathbf{x}, \mathbf{x}_i))$.

3.4 Convexity Analysis and Solving the Optimization Problems

The following theorem specifies sufficient conditions for convexity of Optimization Problem 1. With this theorem we can easily check whether the integrated classifier for co-

variate shift is convex for specific models of the negative log-likelihood functions. The negative log-likelihood function $\ell_{\mathbf{w}}$ itself and its first and second derivatives are needed. Equations A.1 to A.3 in Appendix A define shorthand notation which we will use in the following.

Theorem 3.2 *Optimization Problem 1 is convex if the loss function $\ell_{\mathbf{v}}$ is convex and $\ell_{\mathbf{w}}$ is log-convex and non-negative. The log-convexity condition is equivalent to*

$$\ell_{\mathbf{w}}\ell''_{\mathbf{w}} - \ell'^2_{\mathbf{w}} \geq 0. \tag{3.29}$$

Proof Looking at Optimization Criterion 1 we immediately see that the regularizers are convex. If $\ell_{\mathbf{v}}$ is convex, the second term is convex as well. We therefore only need to analyze the convexity of the term

$$\sum_{i=1}^{m} \frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i)\ell_{\mathbf{w}}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)$$

A sum is convex if the single summands are convex. And a sufficient condition for convexity of a function is that it is non-negative and log-convex. This means we only need to check whether

$$\log \frac{p(s=1)}{p(s=-1)} - \mathbf{v}^\mathsf{T}\mathbf{x}_i + \log \ell_{\mathbf{w},i}$$

is convex. The prior ratio is assumed to be constant. The second term is linear and therefore convex and the third term is the log-convexity condition of $\ell_{\mathbf{w}}$. The second derivative of $\log \ell_{\mathbf{w}}$ is

$$\ell_{\mathbf{w}}^{-1}\ell''_{\mathbf{w}} + \ell_{\mathbf{w}}^{-2}\ell'^2_{\mathbf{w}},$$

thus $\log \ell_{\mathbf{w}}$ is convex if $\ell_{\mathbf{w}}\ell''_{\mathbf{w}} - \ell'^2_{\mathbf{w}}$ is non-negative. ■

In order to check Optimization Criterion 1 for convexity we need to choose models of the negative log-likelihood $\ell_{\mathbf{v}}$ and $\ell_{\mathbf{w}}$ and derive their first and second derivatives. These derivations are also needed to actually minimize Optimization Criterion 1 with the Newton update steps derived in the last section.

We use a logistic model $\ell_{\mathbf{v}}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}) = \log(1 + \exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}))$; the abbreviations of Appendix A can now be expanded:

$$\ell'_{\mathbf{v},i}s_ix_{ij} = -\frac{\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{1+\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}s_ix_{ij}; \quad \ell''_{\mathbf{v},i}x_{ij}x_{ik} = \frac{\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{(1+\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i))^2}x_{ij}x_{ik}. \tag{3.30}$$

For the target classifier, we detail the derivations for logistic and for exponential models of $\ell_{\mathbf{w}}$. For the logistic model the derivatives of $\ell_{\mathbf{w}}$ are the same as for $\ell_{\mathbf{v}}$, only

$\mathbf{v}$ needs to be replaced by $\mathbf{w}$ and s_i by y_i. For an exponential model with $\ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T}\mathbf{x}) = \exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x})$ the abbreviations are expanded as follows:

$$\ell'_{\mathbf{w},i} y_i x_{ij} = -\exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i) y_i x_{ij}; \quad \ell''_{\mathbf{w},i} x_{ij} x_{ik} = \exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i) x_{ij} x_{ik}. \tag{3.31}$$

Using Theorem 3.2 we can now easily check the convexity of the integrated classifier with logistic model and with exponential model for $\ell_{\mathbf{w}}$.

Corollary 3.1 *With a logistic model for $\ell_{\mathbf{w}}$, the condition of Equation 3.29 is violated and therefore Optimization Problem 1 with logistic model for $\ell_{\mathbf{w}}$ is not convex in general.*

Proof Inserting the logistic function into Equation 3.29 we get the following solution.

$$\ell_{\mathbf{w},i} \ell''_{\mathbf{w},i} - \ell'^2_{\mathbf{w},i} = \frac{\exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i)}{(1+\exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}))^2} \left(\log(1+\exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x})) - \exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i)\right) \tag{3.32}$$

The fraction in Equation 3.32 is always positive, the difference term is always negative, thus Optimization Problem 1 with logistic model for $\ell_{\mathbf{w}}$ is non-convex. ■

Empirically, we find that it is a good choice to select the parameters of a regular, *iid* logistic regression classifier as starting point for the Newton gradient search. Since *iid* logistic regression has a convex optimization criterion, this starting point is easily found.

One can easily show that Optimization Problem 1 is non-convex when $\ell_{\mathbf{w}}$ are chosen as hinge loss or quadratic loss.

Corollary 3.2 *Optimization Problem 1 with exponential model for $\ell_{\mathbf{w}}$ is convex.*

Proof Inserting the exponential model into the above criterion results in the nonnegative expression

$$\ell''_{\mathbf{w},i} \ell_{\mathbf{w},i} - \ell'^2_{\mathbf{w},i} = \exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i) \exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i) - (-\exp(-y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i)^2) = 0. \tag{3.33}$$

■

This means the global optimum of Optimization Problem 1 with exponential model for $\ell_{\mathbf{w}}$ can easily be found by Newton gradient descent.

3.5 Two-Stage Approximation to Integrated Model

The previous sections describe a complete solution to the learning problem under covariate shift. Optimization Problem 1 is convex for the exponential model; solving

it using the efficient procedures derived in Section 3.3.5 produces a globally optimal solution.

For the logistic model, unfortunately, it is not convex. Furthermore, the regularized regression classifier is deeply embedded in Optimization Problem 1. It would not be easy to replace it by a different type of classifier such as, for instance, a decision tree. We will now discuss an approximation to Optimization Problem 1 which solves two consecutive optimization problems. The first optimization problem produces example-specific weights; the second step generates a classifier from the weighted examples. Both optimization problems are convex for exponential, logistic, and hinge loss as well as for many other loss functions. But most significantly, the two-stage approximation is *conceptually simple:* the second optimization step can be carried out by any learning procedure that is able to scale the loss incurred by each example using prescribed weight factors. Example-specific weights can easily be incorporated into virtually any learning method. Furthermore, as a result of the decomposition into two optimization problems parameter tuning becomes much easier because cross-validation can be used (cf. Section 3.7).

The derivation in Section 3.3.2 approximates the integral over $\mathbf{v}$ by simultaneously selecting a pair of values which maximize the posterior. This leads to the joint MAP hypothesis over $\mathbf{v}$ and $\mathbf{w}$. In the resulting optimization problem, $\mathbf{v}$ and $\mathbf{w}$ are free parameters. At a higher degree of approximation, one may factorize the posterior (Equation 3.34) and at first approximate the integral over $\mathbf{v}$ by the maximum of $p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X})$ (Equations 3.37 and 3.38). Subsequently, the posterior over $\mathbf{w}$ is maximized given fixed parameters $\mathbf{v}_{\text{MAP}'}$ (Equations 3.35 and 3.36).

$$
\begin{aligned}
\mathbf{w}^* &= \operatorname{argmax}_{\mathbf{w}} \int p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) d\mathbf{v} \\
&= \operatorname{argmax}_{\mathbf{w}} \int p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X};\mathbf{v}) p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) d\mathbf{v} && (3.34)\\
&\approx \operatorname{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X};\mathbf{v}_{\text{MAP}'}) && (3.35)\\
&= \operatorname{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v}_{\text{MAP}'}) p(\mathbf{w}) && (3.36)\\
\text{with}\quad \mathbf{v}_{\text{MAP}'} &= \operatorname{argmax}_{\mathbf{v}} p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) && (3.37)\\
&= \operatorname{argmax}_{\mathbf{v}} p(\mathbf{s}|\mathbf{y},\mathbf{X};\mathbf{v}) p(\mathbf{v}) && (3.38)
\end{aligned}
$$

This results in two optimization problems. Only parameter $\mathbf{v}$ is free in the first stage (Optimization Problem 2). The test-to-training ratio (Equation 3.18) can be derived from the resulting value of $\mathbf{v}$.

Optimization Problem 2 *Over* $\mathbf{v}$*, minimize*

$$\sum_{i=1}^{m+n} \ell_{\mathbf{v}}(s_i \mathbf{v}^\mathsf{T} \mathbf{x}_i) + \frac{1}{2\sigma_{\mathbf{v}}^2} \mathbf{v}^\mathsf{T} \mathbf{v}.$$

In the second stage (Optimization Problem 3), the target model parameters $\mathbf{w}$ are optimized with constant parameters $\mathbf{v}$ and constant example weights. The parameters $\mathbf{v}$ are the result of Optimization Problem 2.

Optimization Problem 3 *Over* $\mathbf{w}$ *(*$\mathbf{v}$ *is constant), minimize*

$$\sum_{i=1}^{m} \frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i) \ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i) + \frac{1}{2\sigma_{\mathbf{w}}^2} \mathbf{w}^\mathsf{T} \mathbf{w}.$$

The criterion of Optimization Problem 3 weights the loss $\ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)$ that each example incurs such that the sample is matched to the test distribution. The last term $\frac{1}{2\sigma_{\mathbf{w}}^2} \mathbf{w}^\mathsf{T} \mathbf{w}$ is the regularizer of the regression. Optimization Problem 3 can easily be adapted to virtually any type of classification mechanism by inserting the appropriate loss function $\ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)$ and regularizer. Operationally, an arbitrary classification procedure is applied to a sample that is either resampled from the training data according to sampling distribution $\frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i)$, or the classifier is applied to the training data with the example-specific loss scaled according to $\frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i)$.

3.6 Kernel Mean Matching and KLIEP

Kernel Mean Matching and KLIEP are both models for directly estimating rescaling weights. Similar to the approximation in Section 3.5 they are two-step approaches that first find weights for the training instances and in the subsequent step a target model is trained over reweighted data. Section 3.6.1 describes kernel mean matching, reveals the relationship to the discriminative model of Section 3.3, and paves the way for a tuning procedure for kernel mean matching. In Section 3.6.2 KLIEP and its log-linear extension are reviewed.

3.6.1 Kernel Mean Matching

Kernel mean matching (Huang et al., 2007) finds weights for the training instances such that the first momentum of training and test sets—*i.e.,* their mean value—matches in feature space (Optimization problem 3.1). $\Phi(\cdot)$ is a mapping into a feature space and

$\sigma_{\mathbf{v}}^2$ is a regularization parameter. Vector α_L denotes all elements α_i with $i = 1, \ldots, m$.

Optimization Problem 3.1

$$\min_{\alpha_L} \quad \left\| \frac{1}{m} \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{x}_i) - \frac{1}{n} \sum_{i=m+1}^{m+n} \Phi(\mathbf{x}_i) \right\|^2$$
$$\text{subject to} \quad \alpha_i \in \left[0, \sigma_{\mathbf{v}}^2\right] \text{ and } \left| \frac{1}{m} \sum_{i=1}^{m} \alpha_i - 1 \right| \leq \epsilon$$

By applying the binomial theorem to the first term in Optimization Problem 3.1 one obtains the equivalent Optimization Problem 3.2 that is a quadratic programming problem and can be solved with standard optimization tools. $K_{(LL)}$ is the kernel matrix between training and $K_{(LT)}$ between training and test instances.

Optimization Problem 3.2

$$\min_{\alpha_L} \quad \frac{1}{2} \alpha_L^\mathsf{T} K_{(LL)} \alpha_L - \frac{m}{n} \alpha_L^\mathsf{T} K_{(LT)} \mathbf{1}$$
$$\text{subject to} \quad \alpha_i \in \left[0, \sigma_{\mathbf{v}}^2\right] \text{ and } \left| \frac{1}{m} \sum_{i=1}^{m} \alpha_i - 1 \right| \leq \epsilon$$

Matching the means in feature space is equivalent to matching all moments of the distributions if a universal kernel is used.

We derive a new interpretation for kernel mean matching that shows its relation to Optimization Problem 2 and the above two-stage approximation to the integrated classifier for covariate shift.

Using a hinge loss for $\ell_{\mathbf{v}}(\mathbf{v}^\mathsf{T}\mathbf{x}_i + b, s_i)$ in Optimization Problem 2 and an explicit offset parameter b we obtain a regular support vector machine. The kernel matrix of this SVM is $\begin{bmatrix} K_{(LL)} & K_{(LT)} \\ K_{(LT)}^\mathsf{T} & K_{(TT)} \end{bmatrix}$ and the target variables are $s_i \in \{1, -1\}$. An SVM can heuristically be simplified by setting the dual parameters α_i for the unlabeled examples to a fixed value $\frac{m}{n}$. This can be interpreted as a mixture between an SVM and a Rocchio classifier. The α_i corresponding to the labeled examples ($s_i = 1$) are trained with an SVM; setting α_i of all unlabeled examples ($s_i = -1$) to $\frac{m}{n}$ approximates the negative class (the unlabeled examples) by their centroid in feature space in accordance with the Rocchio classifier (Joachims, 1997).

The SVM optimization criterion with fixed $\alpha_i = \frac{m}{n}$ for examples with $s_i = -1$ is

$$\min_{\alpha_L} \quad \frac{1}{2}\alpha_L^\mathsf{T} K_{(LL)}\alpha_L - \frac{m}{n}\alpha_L^\mathsf{T} K_{(LT)}\mathbf{1} + \frac{1}{2}\frac{m^2}{n^2}\mathbf{1}K_{(TT)}\mathbf{1} - \alpha_L^\mathsf{T}\mathbf{1} - n\frac{m}{n}$$
$$\text{subject to} \quad \alpha_i \in \left[0, \sigma_\mathbf{v}^2\right] \quad \text{and} \quad \sum_{i=1}^{m}\alpha_i = \sum_{i=1}^{n}\frac{m}{n} = m;$$

again, vector α_L denotes all elements α_i with $i = 1, \ldots, m$. We can drop the constant terms ($\alpha_L^\mathsf{T}\mathbf{1}$ is constant because of the second constraint) and arrive at Optimization Problem 3.3.

Optimization Problem 3.3

$$\min_{\alpha_L} \quad \frac{1}{2}\alpha_L^\mathsf{T} K_{(LL)}\alpha_L - \frac{m}{n}\alpha_L^\mathsf{T} K_{(LT)}\mathbf{1} \quad \text{subject to} \quad \alpha_i \in \left[0, \sigma_\mathbf{v}^2\right] \quad \text{and} \quad \alpha_L^\mathsf{T}\mathbf{1} = m.$$

This is almost identical to the objective of kernel mean matching (Optimization Problem 3.2). The only difference is that Huang et al. (2007) relax the second constraint up to a small constant ϵ, their constraint is $\left|\frac{1}{m}\sum_{i=1}^{m}\alpha_i - 1\right| \leq \epsilon$. Empirically we find that setting ϵ to zero has no impact on the performance. In order to solve the second

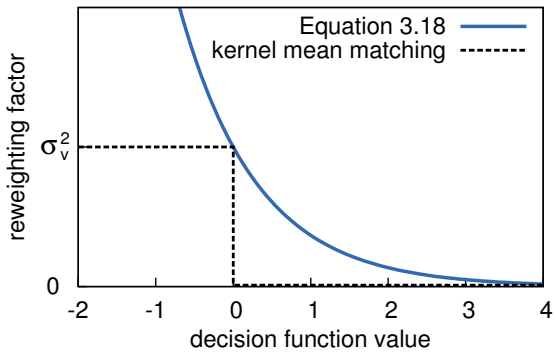

Figure 3.1: Relationship between decision function value and rescaling factor of labeled examples for Equation 3.18 and kernel mean matching, the $\sigma_\mathbf{v}^2$ label on the vertical axis refers only to kernel mean matching.

stage (Optimization Problem 3), kernel mean matching does not use rescaling factors $\frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i - b)$ but directly uses the dual α_i parameters as weights. We want to find out what the relationship between $\frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i - b)$ and α_i is; we derive the Lagrangian of kernel mean matching (Optimization Problem 3.2) and analyze the Karush-Kuhn-Tucker conditions. This results in a similar interpretation as for a regular SVM with three cases shown in Equations 3.39; the difference to a regular SVM

is a threshold on the decision function value of 0 instead of 1. The decision function is $g(\mathbf{x};\alpha,b) = \frac{1}{m}\sum_{j=1}^{m}\alpha_j k(\mathbf{x},\mathbf{x}_j) - \frac{1}{n}\sum_{j=m+1}^{m+n} k(\mathbf{x},\mathbf{x}_j) + b$; the first term is the SVM part and the second the Rocchio part of the decision function. The derivations can be found in Appendix B.

$$\begin{array}{lrcl} \text{case 1:} & g(\mathbf{x}_i;\alpha,b) \geq 0 & \Rightarrow & \alpha_i = 0 \\ \text{case 2:} & g(\mathbf{x}_i;\alpha,b) \leq 0 & \Rightarrow & \alpha_i = \sigma_{\mathbf{v}}^2 \\ \text{case 3:} & g(\mathbf{x}_i;\alpha,b) = 0 & \Rightarrow & 0 < \alpha_i < \sigma_{\mathbf{v}}^2 \end{array} \tag{3.39}$$

With Figure 3.1, we graphically compare the relationship between decision function value and rescaling factor of labeled examples for Equation 3.18 and kernel mean matching (Equations 3.39). For the curve of Equation 3.18 the decision function is $\mathbf{v}^\mathsf{T}\mathbf{x}_i$ (or its dual counterpart). For kernel mean matching the decision function is $g(\mathbf{x};\alpha,b)$ as defined above. The intuition of Equation 3.18 and a discriminative model of the bias between labeled and unlabeled examples is the following. If a labeled example $\mathbf{x}$ is very similar to the unlabeled examples it receives a low decision function value $\mathbf{v}^\mathsf{T}\mathbf{x}$ and a large rescaling factor $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)} \approx \frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x})$. In Figure 3.1 we can see that this intuition roughly also holds for kernel mean matching but the step function causes binary rescaling factors for examples not on the decision boundary. With Equation 3.18 examples never get a rescaling factor of 0, with kernel mean matching this is possible and these examples are completely excluded from the training of the target classifier in the second step.

To sum up, kernel mean matching can be interpreted as an approximated variant of Optimization Problem 2: it uses a partially Rocchio-style approximation to the SVM optimization criterion, instead of the exponential function for the relationship between decision function value and rescaling factor, kernel mean matching uses a step function.

The regularization parameter and the kernel parameters (in kernelized variants) of Optimization Problem 2 and of the KLIEP method (described in the next section) can be easily tuned by cross-validation because these models can be directly applied to out-of-sample data. For a description of such a tuning procedure see Section 5.3.2 and Sugiyama et al. (2008a). In a transductive manner, kernel mean matching estimates only weights for training examples. Until now, there is no out-of-sample extension known and it is unclear how to tune regularization and kernel parameters of kernel mean matching. Based on the relationship to SVMs derived in this section one can develop a tuning procedure for kernel mean matching. The above described decision function $g(\mathbf{x};\alpha,b)$ is an out-of-sample extension for kernel mean matching and can be

used to evaluate tuning data in a cross-validation-based tuning procedure.

3.6.2 KLIEP

KLIEP (Kullback-Leibler importance estimation procedure) estimates rescaling weights by minimizing the Kullback-Leibler divergence between the test and the reweighted training distribution (Sugiyama et al., 2008a). The rescaling weights $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ are modeled by a weighted sum over kernel functions centered at the test instances, $r(\mathbf{x}|\alpha) = \sum_{i=m+1}^{m+n} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$. The goal is to estimate model parameters α by minimizing the Kullback-Leibler divergence between the test and the reweighted training distribution (Equation 3.40). In Equation 3.41 all terms independent of α_i are ignored and in Equation 3.42 the integral is approximated with sum over the test examples.

$$\begin{aligned} \mathrm{KL}\left[p(\mathbf{x}|\theta)||r(\mathbf{x}|\alpha)p(\mathbf{x}|\lambda)\right] &= \int p(\mathbf{x}|\theta) \log \frac{p(\mathbf{x}|\theta)}{r(\mathbf{x}|\alpha)p(\mathbf{x}|\lambda)} && (3.40) \\ &\propto -\int p(\mathbf{x}|\theta) \log r(\mathbf{x}|\alpha) && (3.41) \\ &\approx -\frac{1}{n} \sum_{i=m+1}^{m+n} \log r(\mathbf{x}|\alpha) && (3.42) \end{aligned}$$

This leads to Optimization Problem 3.4. The first constraint enforces that the reweighted empirical distribution over the training examples is a proper distribution and sums to one. For optimization Sugiyama et al. (2008a) propose gradient decent steps interleaved with projection steps to satisfy the constraints.

Optimization Problem 3.4

$$\begin{aligned} \min_{\alpha} \quad & -\frac{1}{n} \sum_{i=m+1}^{m+n} \log r(\mathbf{x}|\alpha) \\ & \text{subject to } \frac{1}{m} \sum_{i=1}^{m} r(\mathbf{x}|\alpha) = 1 \quad \text{and} \quad \alpha_i \geq 0. \end{aligned}$$

Sugiyama et al. (2008b) analyze asymptotic properties of KLIEP and prove the convergence to the true rescaling weights. A log-linear extension (Tsuboi et al., 2008) to KLIEP with $r'(\mathbf{x}|\alpha) \propto \exp(\sum_{i=m+1}^{m+n} \alpha_i K(\mathbf{x}, \mathbf{x}_i))$ leads to an unconstrained optimization problem and has computational advantages.

3.7 Parameter Tuning

Optimization Problem 1 relies on hyper-parameters $\sigma_{\mathbf{v}}^2$ and $\sigma_{\mathbf{w}}^2$ that need to be tuned. For the two-stage approximation of Section 3.5 and reference methods like kernel mean matching two similar parameters need to be specified. In addition to the regularization parameters kernel parameters need to be tuned for non-linear kernels. Parameter tuning for covariate shift models is much more difficult than for regular prediction models because in the covariate shift setting there is no labeled data available drawn from the test distribution. Parameter tuning by regular cross-validation on the labeled training data is inappropriate because the labeled training data is not governed by the test distribution.

In the following paragraphs we describe different tuning procedures; two procedures require prior knowledge and one does not require prior knowledge on the hyper-parameters. The tuning procedures with prior knowledge can be used for all described models. The one without prior knowledge cannot be used for kernel mean matching and the one-stage model of Optimization Problem 1.

A typical setting with prior knowledge on the hyper-parameters is when the difference between training and test data is introduced by a covariate shift over time and the input distribution shifts constantly over time. The most recent data is the unlabeled test data and the older data has been labeled and is the training data. In this setting the parameters can be tuned by splitting the labeled training data into two consecutive parts. The tuning models are learned on the part with earlier timestamps and the hyper-parameters $\sigma_{\mathbf{v}}^2$ and $\sigma_{\mathbf{w}}^2$ and kernel parameters are optimized on the part with later timestamps.

Another setting with prior knowledge is when in addition to the pair of training and test set an additional pair of training and fully labeled test set from a different domain with a similar magnitude of covariate shift is available. This additional set can be used to tune the parameters. Due to the similar magnitude of the covariate shift the optimal parameters for the additional domain are assumed to be a good choice for the parameters of the target domain.

For some two-stage models for covariate shift there is no prior knowledge necessary to tune hyper-parameters. Sugiyama et al. (2008a) propose to tune the regularizer of the KLIEP model with cross-validation. In this manner the first stage parameter $\sigma_{\mathbf{v}}^2$ (and kernel parameters) of the two-stage model of Section 3.5 can be tuned as follows. The training and the test data are both split into training and tuning folds and the hold-out likelihood of the tuning folds is optimized with grid search on $\sigma_{\mathbf{v}}^2$ (and

kernel parameters). The hold-out likelihood measures the predictive performance of the model $p(s|\mathbf{x};\mathbf{v})$ with respect to predicting the selector variable s of the hold out examples. Once the regularizer of the first stage is tuned, the second stage parameter $\sigma^2_{\mathbf{w}}$ (and kernel parameters) can be tuned with cross-validation on weighted training data (Sugiyama and Müller, 2005). The data of training folds as well as the data of tuning folds are weighted with the estimated training-to-test ratio.

Kernel mean matching does not provide out-of-sample predictions and it is therefore difficult to tune the regularization parameter $\sigma^2_{\mathbf{v}}$ with cross-validation. The one-stage model of Optimization Problem 1 is also difficult to tune with cross-validation because there is a bidirectional influence between the parameters $\sigma^2_{\mathbf{v}}$ and $\sigma^2_{\mathbf{w}}$.

In order to compare the one-stage model and kernel mean matching to the other two-stage models we use tuning procedures based on prior knowledge in the empirical studies in the next section.

3.8 Empirical Results

We study the benefit of two versions of the integrated classifier for covariate shift and other reference methods on spam filtering, text classification, and landmine detection problems.

The first integrated classifier uses a logistic model for $\ell_{\mathbf{w}}$ ("integrated log model"), the second an exponential model for $\ell_{\mathbf{w}}$ ("integrated exp model"); $\ell_{\mathbf{v}}$ is a logistic model in both cases.

3.8.1 Reference Methods and Experimental Setup

The first baseline is a classifier trained under *iid* assumption with logistic $\ell_{\mathbf{w}}$. All other reference methods consist of a two-stage procedure: first, the difference between training and test distribution is estimated, the classifier is trained on weighted data in a second step. The second method is kernel mean matching (Section 3.6.1); we set $\epsilon = \sqrt{m-1}/\sqrt{m}$ as proposed by Huang et al. (2007). In the third method, separate density estimates for $p(\mathbf{x}|\lambda)$ and $p(\mathbf{x}|\theta)$ are obtained using kernel density estimation (Shimodaira, 2000), the bandwidth of the kernel is chosen according to the rule-of-thumb of Silverman (1986).

The last two reference methods rely on the two-stage approximation of Optimization Problems 2 and 3 with a logistic regression ("two-stage LR") and an exponential model classifier ("two-stage exp model") as their second stages. The example weights are computed according to Equation 3.18 using a logistic model in the first stage, $p(s =$

$1|\mathbf{x};\mathbf{v})$ is estimated by training a logistic regression that discriminates training from test examples.

The baselines differ in the first stage, the second stage is based on a logistic regression classifier with weighted examples in all cases but the two-stage exponential model baseline. We use a maximum likelihood estimate of $\frac{\frac{m}{m+n}}{\frac{n}{m+n}} = \frac{m}{n}$ for $\frac{p(s=1)}{p(s=-1)}$. We use tuning procedures that rely on prior knowledge (cf. Section 3.7). Short descriptions of the respective tuning data can be found below. For all experiments we tune the regularization parameters of all methods (and the variance parameter of the RBF kernels for the landmine experiments) by maximizing AUC on the tuning set.

3.8.2 Spam Filtering

We use the spam filtering data of Bickel et al. (2007); the collection contains nine different inboxes with test emails (5270 to 10964 emails, depending on inbox) and one set of training emails compiled from various different sources. We use a fixed set of 1000 emails as training data. We randomly select between 32 and 2048 emails from one of the original inboxes. We repeat this process 10 times for 2048 test emails and 20 to 640 times for 1024 to 32 test emails. As tuning data we use the labeled emails from an additional inbox different from the test inboxes. The performance measure is the rate by which the 1-AUC risk is reduced over the *iid* baseline (Bickel and Scheffer, 2007); it is computed as $1 - \frac{1-AUC}{1-AUC_{iid}}$. We use linear kernels for all methods. We analyze the rank of the kernel matrix and find that it fulfills the universal kernel requirement of kernel mean matching; this is due to the high-dimensionality of the data.

Figure 3.2 (top row) shows the results for various numbers of unlabeled examples. The left column of Figure 3.2 compares the integrated classifiers for covariate shift to the kernel mean matching and kernel density estimation baselines. The right column compares the integrated classifiers (Optimization Problem 1) with the two-stage approximations (Optimization Problems 2 and 3). The results for a specific number of unlabeled examples are averaged over 10 to 640 random test samples and averaged over all nine inboxes. Averaged over all users and inbox sizes the absolute AUC of the *iid* classifier is 0.994. Error bars indicate standard errors of the 1-AUC risk.

The integrated and two-step logistic regression and exponential models and kernel mean matching perform similarly well. The differences to the *iid* baseline are highly significant. For 1048 examples the 1-AUC risk is even reduced by an average of 30% with the integrated exponential model classifier! The kernel density estimation procedure is not able to beat the *iid* baseline. The convex integrated exponential model performs

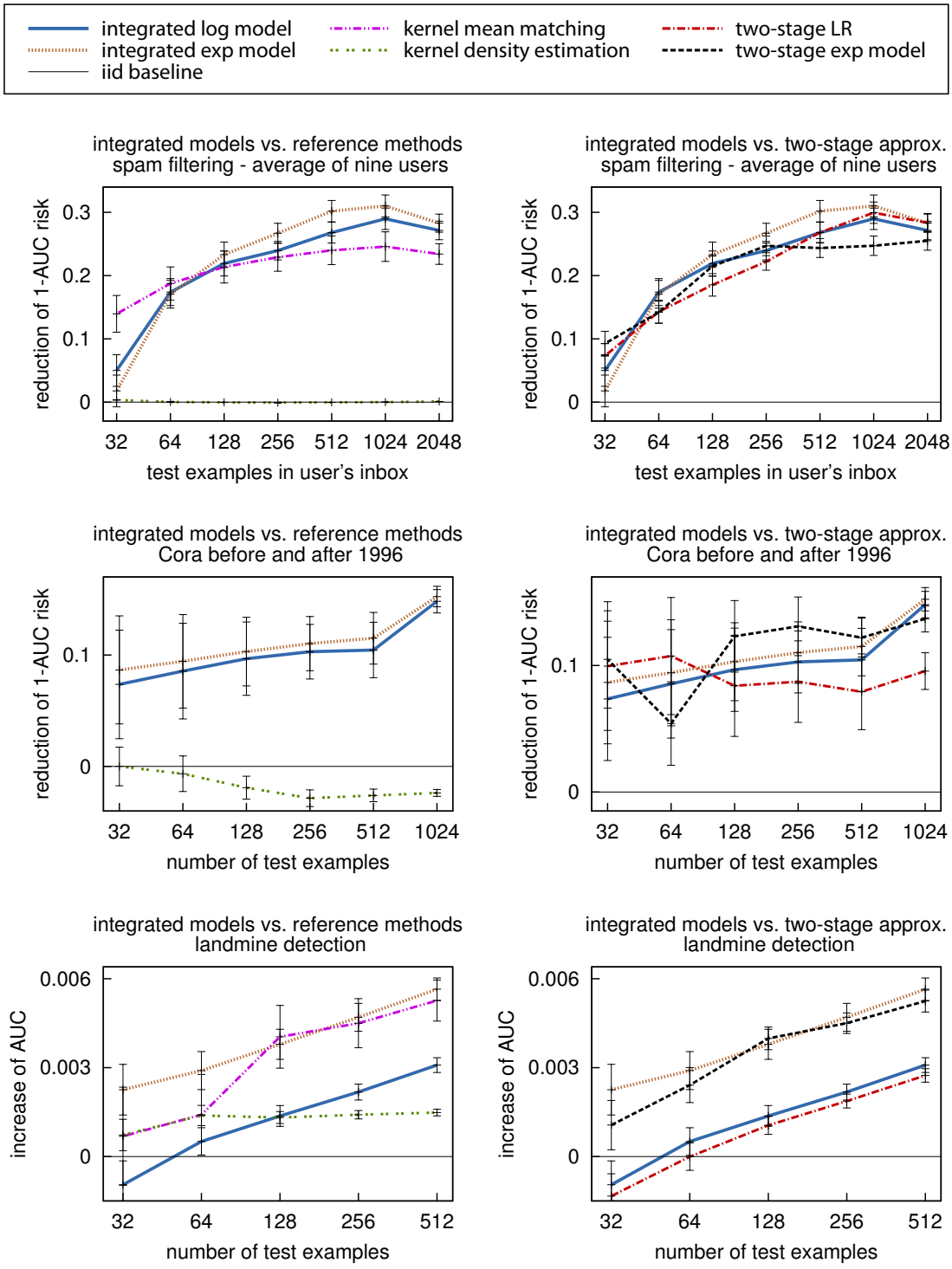

Figure 3.2: Average reduction of 1-AUC risk over nine users for spam filtering (top row) and Cora *Machine Learning/Networking* classification before and after 1996 (second row) and average increase of AUC for landmine detection over 812 pairs of mine fields (bottom row) depending on the number of unlabeled test examples. The left column displays the integrated models together with the kernel mean matching and kernel density estimation baselines, the right column compares the integrated models with their two-stage approximations.

slightly better than its two-stage approximation; for larger number of test examples (512 to 2048) this difference is statistically significant according to a paired t-test with significance level of 5%. For the logistic model, the two-stage optimization performs similarly well as the integrated version.

3.8.3 Text Classification

We now study text classification using computer science papers from the Cora data set. The task is to discriminate Machine Learning from Networking papers. We select 812 papers written before 1996 from both classes as training examples and 1285 papers written after 1996 as test examples. For parameter tuning we apply an additional time split on the training data; we train on the papers written before 1995 and tune on papers written 1995. Parameters are tuned by maximizing AUC. Title and abstract are transformed into *tfidf* vectors, the number of distinct words is 40,000. We again use linear kernels (rank analysis verifies the universal kernel property) and average the results over 20 to 640 random test samples for different sizes (1024 for 20 samples to 32 for 640 samples) of test sets. The resulting 1-AUC risk is shown in Figure 3.2 (second row). The average absolute AUC of the *iid* classifier is 0.998. The methods based on discriminative density estimates significantly outperform all other methods. Kernel mean matching is not displayed because its average performance lies far below the *iid* baseline. The integrated models reduce the 1-AUC risk by 15% for 1024 test examples.

3.8.4 Landmine Detection

In a third set of experiments we study the problem of detecting landmines using the data set of Xue et al. (2007). The collection contains data of 29 mine fields in different regions. Binary labels (landmine or safe ground) and nine dimensional feature vectors extracted from radar images are provided. There are about 500 examples for each mine field. Each of the fields has a distinct distribution of input patterns, varying from highly foliated to desert areas.

We enumerate all 29×28 pairs of mine fields, using one field as training, and the other as test data. For tuning we hold out 4 of the 812 pairs. Parameters are tuned by maximizing AUC. Results are increases over the *iid* baseline, averaged over all $29 \times 28 - 4$ combinations. We use RBF kernels for all methods. The results are displayed in Figure 3.2 (bottom row). The average absolute AUC of the *iid* baseline is 0.64 with a standard deviation of 0.07; note, that the error bars are much smaller than the absolute standard deviation because they indicate the standard error of the *differences* to the *iid* baseline.

For this problem, the exponential model classifiers and kernel mean matching significantly outperform all other methods on average. Considering only methods with logistic target model, kernel mean matching is better than all other methods. Integrated logistic regression and two-stage logistic regression are still significantly better than the *iid* baseline except for 32 and 64 test examples. The integrated classifiers are slightly better than the two-stage variants.

3.9 Conclusion

We derived a discriminative model for learning under covariate shift. The contribution of each training instance to the optimization problem ideally needs to be weighted with its test-to-training density ratio. We show that this ratio can be expressed—without modeling either training or test density—by a discriminative model that characterizes how much more likely an instance is to occur in the test sample than it is to occur in the training sample.

We described a generative model whose parameters can be estimated with a joint MAP hypothesis of both the parameters of the test-to-training model and the final classifier. Optimizing these dependent parameters sequentially incurs an additional approximation compared to solving the joint optimization problem.

We derived a primal and a kernelized Newton gradient descent procedure for the joint optimization problem. Theorem 3.2 specifies the condition for the convexity of Optimization Problem 1. Checking the condition using popular loss functions as models of the negative log-likelihoods reveals that Optimization Problem 1 is only convex with exponential loss.

We gave a new interpretation for kernel mean matching as an approximation to Optimization Problem 2. This finding also led to an out-of-sample extension for kernel mean matching that can be used for tuning of its kernel and regularization parameters.

Empirically, we found that the integrated and the two-stage models as well as kernel mean matching outperform the *iid* baseline and the kernel density estimation model in almost all cases. In some cases, the integrated models perform slightly better than their two-stage counterparts. The performance of kernel mean matching depends on the problem; for one out of three problems it did not beat the *iid* baseline, for the others it yielded comparable results to the integrated models.

The two-stage model is conceptually simpler than the integrated model, and may in some cases have the greatest practical utility. The main advantage compared to the integrated model is that regularization parameters can be tuned without prior

knowledge by cross-validation. Another advantage of the two-stage model is that in the second stage, after the example-specific weights have been derived, virtually any learning mechanism can be employed to produce the final classifier from the weighted training sample. This comes at the cost of only a marginal loss of performance compared to the integrated model.

4 Multi-Task Learning

In the multi-task setting the divergence between training and test distributions is reflected in the joint distribution of inputs and outputs across tasks, and is not restricted to the inputs, as in the covariate shift setting. The term "multi-task" indicates that there are several (usually more than two) distinct prediction tasks each with its own distribution over inputs and outputs. Within each task the training and the test data are governed by the identical task specific distribution; across tasks, the joint distributions may differ.

Existing methods for multi-task learning often assume that all tasks share one common model structure. For example, hierarchical Bayesian models assume that the parameters of all tasks are drawn from one common prior distribution. If there are tasks that do not have anything in common with all other tasks, or if the common prior is not flexible enough to capture the relationship between tasks, such models might not work well in practice.

In this chapter we contribute a new multi-task learning model that can handle arbitrarily different data distributions between tasks without making assumptions about the data generation process or the relation between tasks. We show that by appropriately weighting each instance in the pool of all examples, one can match the distribution that governs the pool of examples of all tasks to each of the single task distributions. We show how appropriate weights can be obtained by discriminating the labeled sample for a given task against the pooled sample from all other tasks.

The structure of the chapter is as follows. We begin with a definition of the problem setting in Section 4.1. In Section 4.2 we give an introduction to hierarchical Bayesian methods for multi-task learning and we show that well known feature mappings for multi-task learning and taxonomy classification can be directly derived from hierarchical Bayesian models. Furthermore, we develop a nested hierarchical Bayesian model for Gaussian processes. Other related work on multi-task learning is reviewed in Section 4.3. In Section 4.4, we devise our new multi-task learning method based on distribution matching. A case study on HIV therapy screening is presented in Section 4.5. Section 4.6 concludes this chapter.

4.1 Problem Setting

In supervised *multi-task learning*, each of several tasks z is characterized by an unknown joint distribution $p(\mathbf{x}, y|z)$ of features $\mathbf{x}$ and label y given the task z. The joint distributions of different tasks may differ arbitrarily but usually some tasks have similar distributions. A training sample $L = \langle(\mathbf{x}_1, y_1, z_1), \ldots, (\mathbf{x}_m, y_m, z_m)\rangle$ collects examples from all tasks. There may be tasks with little or even no training data. For each example, input attributes $\mathbf{x}_i$, class label y_i, and the originating task z_i are known. The entire sample L is governed by the mixed joint density $p(z)p(\mathbf{x}, y|z)$. The prior $p(z)$ specifies the task proportions.

The goal is to learn a hypothesis $f_z : \mathbf{x} \mapsto y$ for each task z. This hypothesis $f_z(\mathbf{x})$ should correctly predict the true label y of unseen examples drawn from $p(\mathbf{x}|z)$ for all z. That is, it should minimize the expected loss

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|z)}[\ell(f_z(\mathbf{x}), y)]$$

with respect to the unknown joint distribution $p(\mathbf{x}, y|z)$ for each individual z.

In some application settings we may have prior knowledge on the similarity of tasks in addition to training data. We assume that this knowledge is encoded in a kernel function $k(z, z')$ for a pair of tasks z and z'.

4.2 Hierarchical Bayesian Learning

Hierarchical Bayesian models constitute the traditional statistical approach to multi-task learning. They are an extension to simple Bayesian models. The building blocks of a simple Bayesian model are a prior probability function $p(\mathbf{w}|\phi)$ on model parameters $\mathbf{w}$ given hyperparameters ϕ and a likelihood function $p(L|\mathbf{w})$ that measures how likely the training data L is under model parameters $\mathbf{w}$. With Bayes' rule the posterior probability is given by $p(\mathbf{w}|L, \phi) \propto p(L|\mathbf{w})p(\mathbf{w}|\phi)$. For full Bayesian predictions the model parameters are integrated out, $y^* = \operatorname{argmax}_y \int p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w}|L, \phi)d\mathbf{w}$. Usually, the integral is only tractable if the prior conjugates to the likelihood function. If this is not the case one can resort to a *maximum a posteriori* (MAP) estimation by choosing the model parameters that maximize the posterior, $\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} p(\mathbf{w}|L, \phi)$. Based on this MAP point estimate the prediction function is $y^* = \operatorname{argmax}_y p(y|\mathbf{x}, \mathbf{w}^*)$.

In the *simple* Bayesian model the hyperparameters ϕ are fixed. In a *hierarchical* Bayesian model the hyperparameters are variable and are included in the inference process (Gelman et al., 2004). A hyperprior $p(\phi)$ encodes our prior knowledge on the

hyperparameters. The joint posterior of model parameters and hyperparameters in a hierarchical Bayesian model is shown in Equation 4.1.

$$p(\mathbf{w}, \phi|L) \quad \propto \quad p(L|\mathbf{w})p(\mathbf{w}|\phi)p(\phi) \tag{4.1}$$

For full Bayesian predictions parameters $\mathbf{w}$ and hyperparameters ϕ are integrated out (Equation 4.2).

$$y^* = \underset{y}{\text{argmax}} \int p(y|\mathbf{x}, \mathbf{w}) \left(\int p(\mathbf{w}, \phi|L) d\phi \right) d\mathbf{w} \tag{4.2}$$

Similar to the simple Bayesian case, if the integrals are intractable Equation 4.2 can be approximated by a MAP point estimate of both parameters, $(\mathbf{w}^*, \phi^*) = \text{argmax}_{\mathbf{w},\phi}\, p(\mathbf{w}, \phi|L)$. Again, the prediction function is $y^* = \text{argmax}_y\, p(y|\mathbf{x}, \mathbf{w}^*)$.

Hierarchical Bayesian models can play to their strength in a multi-task setting when several related model parameters $\mathbf{w}_1, \ldots, \mathbf{w}_l$ (l is the number of tasks) share the same prior probability $p(\mathbf{w}_z|\phi)$. This is based on the generative assumption that all task-specific model parameters $\mathbf{w}_z$ are drawn from a common prior. For each task z, training data $L_z = \{(\mathbf{x}_i, y_i, z_i) \in L : z_i = z\}$ is available and with conditional independence of the task parameters given the prior $p(\mathbf{w}_z|\phi)$ the posterior of all model parameters can be expressed as in Equation 4.3.

$$p(\mathbf{w}_1, \ldots, \mathbf{w}_l, \phi|L_1, \ldots, L_l) \quad \propto \quad p(\phi) \prod_z p(L_z|\mathbf{w}_z)p(\mathbf{w}_z|\phi) \tag{4.3}$$

Again, if the integrals are tractable the parameters can be integrated out for prediction, otherwise a MAP approximation can be used to obtain a point estimate of the parameters.

Because the inference in a hierarchical Bayesian model also includes the prior parameters ϕ, the tasks can "exchange information" via the prior. Optimally, the prior $p(\mathbf{w}_z|\phi)$ captures parameter structures that are common to all tasks. For tasks with only a few or no training examples the prior dominates the posterior and therefore the task inherits the prior parameters learned from all other tasks. The more training data is available for a task, the less influence the prior has on the posterior. If there is no true common structure over all tasks or if the prior is not flexible enough to capture this structure a hierarchical Bayesian model is likely to be inferior to separately trained models without any interaction.

Hierarchical Bayesian modeling for multi-task learning is well studied in the machine

learning community. Heskes (2000) imposes a Gaussian prior on the parameters of related neural network models. A Dirichlet process can serve as prior in a hierarchical Bayesian model and cluster the tasks (Xue et al., 2007; Roy and Kaelbling, 2007); all tasks in one cluster share the same model parameters. More flexibility gives a hierarchical Dirichlet process prior (Teh et al., 2006). Two (or even more) layers of Dirichlet process priors are nested in a way that the single instances belonging to each task are clustered and the cluster models can be shared across tasks.

Maximum entropy models sharing a common Laplacian prior are studied by Dudik et al. (2007). Liu et al. (2008) derive a chain structured hierarchical Bayesian model for multi-task learning on ordered tasks. The model parameters for each task are assumed to be drawn from a mixture over the parameters of all previous tasks. Evgeniou and Pontil (2004) discover a link between hierarchical Bayes and kernel learning with support vector machines. In Section 4.2.1 we give more details on this link and show that the hierarchical Bayesian kernel gets a clean Bayesian interpretation by choosing a logistic or a ridge regression instead of a support vector machine model.

Yu et al. (2005) and Schwaighofer et al. (2005) impose a normal-inverse Wishart hyperprior on the mean and covariance of a Gaussian process prior that is shared by all task-specific regression functions. Mean and covariance of the Gaussian process are estimated using the EM algorithm. In Section 4.2.3 we give more details on this model. The same framework with t-processes instead of Gaussian processes leads to a multi-task model that is more robust to outlier tasks (Yu et al., 2007). In Section 4.2.4 we derive a nested version of the hierarchical Bayesian Gaussian process model for grouped tasks.

Multi-task feature selection aims in sharing a sparse feature representation over all tasks (Obozinski et al., 2007; Argyriou et al., 2007). Bi et al. (2008) show that these methods can be casted as a hierarchical Bayesian model with Gaussian prior on the task parameters and Wishart hyperprior.

4.2.1 Hierarchical Bayesian Kernel Learning

Evgeniou and Pontil (2004) describe a feature mapping and a kernel function for multi-task learning with support vector machines. They argue that their model follows the intuition of hierarchical Bayes but do not provide a probabilistic derivation. In this section we show that a modified version of their model, with a logistic or squared loss instead of a hinge loss, is equivalent to a hierarchical Bayesian model. In order to make the derivations more concrete we choose a logistic loss in the following argumentation.

The derivations carry over to the squared loss by replacing the logistic with a Gaussian model for the data likelihood. For each task z we choose a logistic model $p(y|\mathbf{x}, \mathbf{w}_z) = 1/(1 + \exp(-y\mathbf{w}_z^\mathsf{T}\mathbf{x}))$ as likelihood function with binary label $y \in \{-1, 1\}$. We assume the model parameters $\mathbf{w}_z$ for each task z are drawn from a Gaussian prior $\mathbf{w}_z \sim N(\mathbf{w}_0, \sigma_\mathbf{w}^2\mathbf{I})$ shared by all tasks. A Gaussian hyperprior on the mean $\mathbf{w}_0 \sim N(\mathbf{0}, \sigma_{\mathbf{w}_0}^2\mathbf{I})$ is imposed. The hyperparameters $\mathbf{w}_0$ correspond to ϕ of Equation 4.1. Prior and hyperprior both have a scalar covariance matrix with fixed scalars $\sigma_{\mathbf{w}_0}^2$ and $\sigma_\mathbf{w}^2$. The following generative process summarizes the model:

1. Draw prior mean once from Gaussian hyperprior, $\mathbf{w}_0 \sim N(\mathbf{0}, \sigma_{\mathbf{w}_0}^2\mathbf{I})$;
2. For all tasks z draw parameters from Gaussian prior, $\mathbf{w}_z \sim N(\mathbf{w}_0, \sigma_\mathbf{w}^2\mathbf{I})$;
3. For each input vector $\mathbf{x}$ and task z independently draw label y using a logistic model $p(y|\mathbf{x}, \mathbf{w}_z) = \frac{1}{1+\exp(-y\mathbf{w}_z^\mathsf{T}\mathbf{x})}$.

In accordance with the generative process and Equation 4.3 the log-posterior of the hyperparameters and all task parameters given the data is proportional to the log-factorization of Equation 4.4. Equation 4.5 expands the density functions and drops constant terms. We can express each task parameter $\mathbf{w}_z$ by the prior mean $\mathbf{w}_0$ plus an offset vector $\mathbf{v}_z$. We rewrite the log-posterior in terms of offset vectors $\mathbf{v}_z$ with the replacement $\mathbf{w}_z = \mathbf{w}_0 + \mathbf{v}_z$ (Equation 4.6). In Equation 4.7 we introduce a new vector $\mathbf{v}_0$ by replacing $\mathbf{w}_0$ with $\frac{\sigma_{\mathbf{w}_0}}{\sigma_\mathbf{w}}\mathbf{v}_0$. Thus, the vector $\mathbf{v}_0$ is just the prior mean $\mathbf{w}_0$ rescaled by a constant.

$$
\begin{aligned}
&\log p(\mathbf{w}_1, \ldots, \mathbf{w}_l, \mathbf{w}_0 | L_1, \ldots, L_l, \sigma_\mathbf{w}^2, \sigma_{\mathbf{w}_0}^2) \\
&\quad\propto\quad \log N(\mathbf{w}_0|\mathbf{0}, \sigma_{\mathbf{w}_0}^2\mathbf{I}) + \sum_{z=1}^{l} \log N(\mathbf{w}_z|\mathbf{w}_0, \sigma_\mathbf{w}^2\mathbf{I}) + \sum_{z=1}^{l} \sum_{(\mathbf{x},y)\in L_z} \log p(y|\mathbf{x}, \mathbf{w}_z) && (4.4)\\
&\quad\propto\quad -\frac{||\mathbf{w}_0||^2}{2\sigma_{\mathbf{w}_0}^2} - \sum_{z=1}^{l} \frac{||\mathbf{w}_z - \mathbf{w}_0||^2}{2\sigma_\mathbf{w}^2} - \sum_{z=1}^{l} \sum_{(\mathbf{x},y)\in L_z} \log(1 + \exp(-y\mathbf{w}_z^\mathsf{T}\mathbf{x})) && (4.5)\\
&\quad=\quad -\frac{||\mathbf{w}_0||^2}{2\sigma_{\mathbf{w}_0}^2} - \sum_{z=1}^{l} \frac{||\mathbf{v}_z||^2}{2\sigma_\mathbf{w}^2} - \sum_{z=1}^{l} \sum_{(\mathbf{x},y)\in L_z} \log(1 + \exp(-y(\mathbf{w}_0^\mathsf{T}\mathbf{x} + \mathbf{v}_z^\mathsf{T}\mathbf{x}))) && (4.6)\\
&\quad=\quad -\frac{||\mathbf{v}_0||^2}{2\sigma_\mathbf{w}^2} - \sum_{z=1}^{l} \frac{||\mathbf{v}_z||^2}{2\sigma_\mathbf{w}^2} - \sum_{z=1}^{l} \sum_{(\mathbf{x},y)\in L_z} \log\left(1 + \exp\left(-y\left(\frac{\sigma_{\mathbf{w}_0}}{\sigma_\mathbf{w}}\mathbf{v}_0^\mathsf{T}\mathbf{x} + \mathbf{v}_z^\mathsf{T}\mathbf{x}\right)\right)\right) && (4.7)\\
&\quad=\quad -\frac{||\mathbf{v}||^2}{2\sigma_\mathbf{w}^2} - \sum_{(\mathbf{x},y,z)\in L} \log(1 + \exp(-y\mathbf{v}^\mathsf{T}\Phi(\mathbf{x}, z))) && (4.8)
\end{aligned}
$$

We introduce a vector $\mathbf{v}$ that is a concatenation of all parameter vectors, $\mathbf{v} = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_l]$. For all training examples we map the pairs $(\mathbf{x}, z)$ of input features and task identifier into a new feature space using the feature mapping $\Phi(\mathbf{x}, z)$ defined in Equation 4.9 (Evgeniou and Pontil, 2004).

$$\Phi(\mathbf{x}, z) = \left[\frac{\sigma_{\mathbf{w}_0}}{\sigma_{\mathbf{w}}}\mathbf{x}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{1,\ldots,z-1}, \underbrace{\mathbf{x}}_{z}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{z+1,\ldots,l}\right] \tag{4.9}$$

With this feature mapping each task receives its individual section in the new feature space. For a training example from task z the original feature vector $\mathbf{x}$ is copied to the range in the new feature space that belongs to task z, all other regions except the first one are filled with zero vectors. All tasks share the first section in the new feature space, task $z = 1$ receives the second segment, task $z = 2$ receives the third segment, and so on. Applying the feature mapping and the substitution of $\mathbf{v}$ we reach Equation 4.8. This is the objective function of a regular logistic regression with input examples transformed by feature mapping $\Phi(\mathbf{x}, z)$.

To sum up, the maximum of Equation 4.8 with respect to parameter vector $\mathbf{v}$ is a MAP estimate for the parameters of a hierarchical Bayesian model with Gaussian prior and hyperprior with fixed scalar covariance matrices. This maximum can be obtained using a standard logistic regression with feature mapping $\Phi(\mathbf{x}, z)$. Once the logistic regression is trained, parameter $\mathbf{v}^*$ and thereby the joint MAP estimate for all parameters $\mathbf{w}_1^*, \ldots, \mathbf{w}_l^*, \mathbf{w}_0^*$ is available. A prediction for instance $\mathbf{x}$ of task z can be obtained by $y^* = \text{argmax}_y \, 1/(1 + \exp(-y\mathbf{v}^{*\mathsf{T}}\Phi(\mathbf{x}, z)))$.

The parameter vector $\mathbf{v}$ in Equation 4.8 has the dimensionality of $\mathbf{x}$ times $l + 1$. Instead of optimizing over this potentially very high-dimensional space one can resort to optimization in the dual space using a standard kernel logistic regression (Zhu and Hastie, 2002). This reduces the number of parameters to the number of training examples. In Equation 4.10 the kernel function for the kernel logistic regression is directly derived from Equation 4.9.

$$k_{hBayes}((\mathbf{x}, z), (\mathbf{x}', z')) = \Phi(\mathbf{x}, z)^{\mathsf{T}}\Phi(\mathbf{x}', z') = \left(\frac{\sigma_{\mathbf{w}_0}^2}{\sigma_{\mathbf{w}}^2} + \delta(z, z')\right)\mathbf{x}^{\mathsf{T}}\mathbf{x}' \tag{4.10}$$

$\delta(z, z')$ is the Kronecker delta. One can replace the linear kernel $\mathbf{x}^{\mathsf{T}}\mathbf{x}'$ on the right hand side of Equation 4.10 with any non-linear kernel function $k(\mathbf{x}, \mathbf{x}')$.

4.2.2 Hierarchical Bayes for Taxonomy Classification

In this section we prove that a well known model for taxonomy classification is a nested hierarchical Bayesian model and explain how taxonomy classification is related to multi-task learning.

Lafferty et al. (2001) introduce conditional random fields and thereby extend logistic regression for structured and interdependent output variables. Conditional random fields are based on a conditional exponential model of an output structure $\mathbf{y}$ given input $\mathbf{x}$ and model parameters $\mathbf{v}$,

$$p(\mathbf{y}|\mathbf{x},\mathbf{v}) = \frac{\exp(\Phi(\mathbf{x},\mathbf{y})^\mathsf{T}\mathbf{v})}{\sum_{\mathbf{y}'}\exp(\Phi(\mathbf{x},\mathbf{y}')^\mathsf{T}\mathbf{v})},$$

with an application specific input-output mapping $\Phi(\mathbf{x},\mathbf{y})$. The standard training procedure is a MAP estimation with isotropic Gaussian prior on $\mathbf{v}$. Optimization Problem 4.1 trains a conditional random field by maximizing the log-posterior of the model parameters $\mathbf{v}$ given training data and the prior variance σ^2. The variable i indexes the training examples.

Optimization Problem 4.1 *Over parameters* $\mathbf{v}$*, maximize*

$$-\frac{||\mathbf{v}||^2}{2\sigma^2} + \sum_i \left(\Phi(\mathbf{x}_i,\mathbf{y}_i)^\mathsf{T}\mathbf{v} - \log\sum_{\mathbf{y}'}\exp\Phi(\mathbf{x}_i,\mathbf{y}')^\mathsf{T}\mathbf{v} \right).$$

In taxonomy classification the goal is to assign objects with feature vector $\mathbf{x}$ to nodes y in a label hierarchy. A conditional random field model is instantiated for taxonomy classification by using the input-output mapping $\Phi_{\text{tax}}(\mathbf{x},y)$ of Equation 4.11 (Tsochantaridis et al., 2005). In taxonomy classification the output structure $\mathbf{y}$ reduces to a node label y. $\Phi(\mathbf{x})$ is any suitable input mapping and $\otimes$ denotes the tensor product.

$$\Phi_{\text{tax}}(\mathbf{x},y) \quad = \quad \Phi(\mathbf{x}) \otimes \Lambda(y). \tag{4.11}$$

The function $\Lambda(y)$ maps each node label to a binary vector with one element for each node in the taxonomy. We denote such a single element for node y' with $\Lambda_{y'}(y)$ and define it in Equation 4.12:

$$\Lambda_{y'}(y) \quad = \quad \begin{cases} 1 & \text{if } y' \text{ lies on path from } y \text{ to a root node} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

The mapping $\Phi_{\text{tax}}(\mathbf{x}, y)$ has a segment for each node in the taxonomy and copies the vector $\Phi(\mathbf{x})$ to all segments corresponding to the nodes on the path from a root node to y (including the segment for the root node and y itself). We assume that each node has exactly one parent node in the taxonomy except for the root nodes.

We will now show that Optimization Problem 4.1 together with mapping $\Phi_{\text{tax}}(\mathbf{x}, y)$ can be derived from a nested hierarchical Bayesian model with the following generative process, where R is the set of root nodes and l is the total number of nodes in the taxonomy:

1. For all root nodes $y \in R$ in the taxonomy draw parameter vector $\mathbf{w}_y$ from an isotropic Gaussian prior with zero mean, $\mathbf{w}_y \sim N(\mathbf{0}, \sigma^2\mathbf{I})$;

2. For all non-root nodes $y \notin R$ (nodes are assumed to be ordered by decreasing distance to root nodes) draw parameter vector $\mathbf{w}_y$ from a Gaussian prior with the mean vector equal to the parents' parameters, $\mathbf{w}_y \sim N(\mathbf{w}_{\pi(y)}, \sigma^2\mathbf{I})$, $\pi(y)$ is the parent node of y;

3. For each input vector $\mathbf{x}$ draw node label y using an exponential model $p(y|\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_l) = \frac{\exp(\Phi(\mathbf{x})^\mathsf{T}\mathbf{w}_y)}{\sum_{y'} \exp(\Phi(\mathbf{x})^\mathsf{T}\mathbf{w}_{y'})}$ and a feature mapping $\Phi(\mathbf{x})$.

Following this generative process the log-posterior of all node parameters given training data L and variance σ^2 is proportional to the log-factorization as shown in Equation 4.13.

$$
\begin{aligned}
&\log p(\mathbf{w}_1, \ldots, \mathbf{w}_l | L, \sigma^2) \\
&\propto \log \sum_{y \in R} N(\mathbf{w}_y | \mathbf{0}, \sigma^2\mathbf{I}) + \sum_{y \notin R} \log N(\mathbf{w}_y | \mathbf{w}_{\pi(y)}, \sigma^2\mathbf{I}) + \sum_i p(y_i | \mathbf{x}_i, \mathbf{w}_1, \ldots, \mathbf{w}_l) \qquad (4.13) \\
&\propto -\sum_{y \in R} \frac{||\mathbf{w}_y||^2}{2\sigma^2} - \sum_{y \notin R} \frac{||\mathbf{w}_y - \mathbf{w}_{\pi(y)}||^2}{2\sigma^2} \qquad (4.14) \\
&\quad + \sum_i \left(\Phi(\mathbf{x}_i)^\mathsf{T}\mathbf{w}_{y_i} - \log \sum_{y'} \exp(\Phi(\mathbf{x}_i)^\mathsf{T}\mathbf{w}_{y'}) \right) \\
&= -\sum_{y \in R} \frac{||\mathbf{v}_y||^2}{2\sigma^2} - \sum_{y \notin R} \frac{||\mathbf{v}_y||^2}{2\sigma^2} \qquad (4.15) \\
&\quad + \sum_i \left(\sum_{j \in \text{path}(y_i)} \Phi(\mathbf{x}_i)^\mathsf{T}\mathbf{v}_j - \log \sum_{y'} \exp \left(\sum_{j \in \text{path}(y')} \Phi(\mathbf{x}_i)^\mathsf{T}\mathbf{v}_j \right) \right) \\
&= -\frac{||\mathbf{v}||^2}{2\sigma^2} + \sum_i \left(\Phi_{\text{tax}}(\mathbf{x}_i, y_i)^\mathsf{T}\mathbf{v} - \log \sum_{y'} \exp \Phi_{\text{tax}}(\mathbf{x}_i, y')^\mathsf{T}\mathbf{v} \right) \qquad (4.16)
\end{aligned}
$$

Equation 4.14 follows by expanding the density functions and dropping the constant normalization terms. In Equation 4.15 we rewrite the expression in terms of offset vectors $\mathbf{v}_y$ with the replacement $\mathbf{w}_y = \mathbf{w}_{\pi(y)} + \mathbf{v}_y$ for all non-root nodes $y \notin R$. Remember, that $\pi(y)$ denotes the parent node of y. Within the sum over the training data likelihoods this recursive replacement results in a sum over nodes $j \in \text{path}(y)$. $\text{path}(y)$ is the set of nodes on the path from a root node to node y (including the root and y). For a unified notation we also replace all parameters $\mathbf{w}_y$ of root nodes $y \in R$ with $\mathbf{v}_y$.

We exemplify the recursive replacement for a specific node y two levels above from a root node: In Equations 4.17 and 4.18 the first and second (recursive) replacements are applied. Equation 4.19 replaces the parameter symbol for the root node and Equation 4.20 collects all parameters on the path from the root node to y.

$$\begin{aligned}
\mathbf{w}_y &= \mathbf{w}_{\pi(y)} + \mathbf{v}_y && (4.17)\\
&= \mathbf{w}_{\pi(\pi(y))} + \mathbf{v}_{\pi(y)} + \mathbf{v}_y && (4.18)\\
&= \mathbf{v}_{\pi(\pi(y))} + \mathbf{v}_{\pi(y)} + \mathbf{v}_y && (4.19)\\
&= \sum_{j \in \text{path}(y)} \mathbf{v}_j && (4.20)
\end{aligned}$$

Equation 4.16 follows by applying the feature mapping of Equation 4.11 and by introducing a vector $\mathbf{v}$ that is just the concatenation of all node parameter vectors, $\mathbf{v} = [\mathbf{v}_1, \ldots, \mathbf{v}_l]$.

We observe that Equation 4.16 is identical to the objective function of Optimization Problem 4.1 in combination with the input-output mapping of Equation 4.11. This shows that training a conditional random field with the standard feature mapping for taxonomy classification by Tsochantaridis et al. (2005) is equivalent to MAP estimation of a hierarchical Bayesian model with nested Gaussian priors.

Based on this finding, hierarchical Bayesian models for taxonomy classification with more flexible prior structures can be developed. For example, one can deviate from using one fixed covariance matrix $\sigma^2\mathbf{I}$ for all priors and learn different scalar parameters σ^2 for each node by imposing an inverse Gamma prior on the individual σ^2.

The relationship of taxonomy classification to multi-task learning is the following. In taxonomy classification a related set of node parameters, in multi-task learning a related set of task parameters are estimated. The main difference is that for the inference over one instance $\mathbf{x}$ in multi-task learning only the associated task parameters of one task are used. In taxonomy classification the parameters of all nodes are involved and needed

for normalization of the conditional node probability (see step 3 of the above generative process). The second difference is that multi-task learning is usually based on a *flat* set of nodes in contrast to taxonomy classification with a *hierarchical* node structure.

The generative process of the taxonomy classification model is similar to the generative process of the multi-task model described in Section 4.2.1. The taxonomy classification model can be considered as an extension of this multi-task model to a hierarchical structure and with a different data likelihood function.

4.2.3 Hierarchical Bayes with Gaussian Processes

A stochastic process is called Gaussian process if any finite set or subset of random variables belonging to the stochastic process are governed by a multivariate Gaussian distribution. Gaussian processes can be used to define prior distributions over functions of input variables. If a function is assumed to be drawn from a Gaussian process prior and data drawn from this function is available, the resulting posterior over functions is again a Gaussian process. Using a Gaussian process posterior, predictions for new data can be obtained easily. Such a model can be used for nonparametric regression. A thorough introduction to Gaussian processes is beyond the scope of this thesis. More details on Gaussian processes can be found in Rasmussen and Williams (2006).

Schwaighofer et al. (2005) follow the hierarchical Bayesian paradigm and impose one Gaussian process prior on the functions of several related tasks. They want to find a MAP estimate of the shared prior and observe that if this estimate is based on a finite number of training examples, the Gaussian process prior over a continuous input domain can be replaced by a Gaussian process prior over a finite number of input locations and this is equivalent to a regular multivariate Gaussian distribution $N(\mu, \mathbf{K})$ as common prior. For MAP inference only the mean μ and covariance matrix $\mathbf{K}$ of this Gaussian distribution need to be estimated and they are assumed to be governed by a normal-inverse Wishart (NIW) hyperprior. The NIW distribution is a natural choice because it is the conjugate prior for the multivariate Gaussian distribution.

The function for task z drawn from the shared Gaussian prior $N(\mu, \mathbf{K})$ is a vector $\mathbf{f}_z$ over function values for all training examples in L. For task z the function value for training instance i is denoted by f_{zi}. Analogously, μ and $\mathbf{K}$ are mean vector and covariance matrix over all input locations $\mathbf{x}$ in L. The generative process underlying the hierarchical Bayesian model for a given set of training inputs is the following:

1. Draw mean vector and covariance matrix once from a normal-inverse Wishart hyperprior, $(\mu, \mathbf{K}) \sim N(\mu|\mathbf{0}, \frac{1}{\pi}\mathbf{K})IW(\mathbf{K}|\tau, \kappa^{-1})$, given fixed scalars π, τ, and

matrix κ;

2. For each task z, independently draw vector with function values $\mathbf{f}_z$ from common prior, $\mathbf{f}_z \sim N(\mu, \mathbf{K})$;

3. For all tasks z and all input vectors $\mathbf{x}_i$ from L_z, $y_i = f_{zi} + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$.

Schwaighofer et al. use an expectation maximization (EM) algorithm (Dempster et al., 1977) to obtain a MAP estimate for the prior parameters, $(\mu^*, \mathbf{K}^*) = \text{argmax}_{\mu,\mathbf{K}}\, p(\mu, \mathbf{K}|L, \sigma^2, \pi, \tau, \kappa)$. Besides the three scalar parameters σ^2, π, and τ the scale matrix κ needs to be specified. This matrix can be any suitable positive definite kernel matrix over the training data, e.g.: linear, polynomial, rbf, etc. With this hierarchical Bayesian model the learned Gaussian process prior is only defined over a discrete set of inputs. If a test instance is not included in this discrete set, predictions are not directly available. In a transductive manner one can add the test instances to the training data with all labels set to zero so that the labels do not influence the learned prior but are included in the discrete input domain of the Gaussian process prior (Schwaighofer et al., 2005). Yu et al. (2005) introduce a modification to the model that can handle out-of-sample predictions.

The derivation of an EM algorithm for an extended version of the model with an additional layer can be found in the next section and in Appendix C.

4.2.4 Nested Hierarchical Bayes with Gaussian Processes

The multi-task model of Schwaighofer et al. (2005) described in the previous section can be used to model related regression functions with homogeneous relationships between the functions/tasks. In some applications the tasks are grouped, for example, in collaborative filtering each user can be modeled as a separate task (Schwaighofer et al., 2005) and the tasks can be grouped by gender. Users in the male group might be more similar to other males than to females. Another example is the prediction of the outcome of similar therapies (modeled as tasks) for patients that can be grouped by country. More details on this application can be found in Section 4.5.

In the following, we derive a new nested hierarchical Bayesian model for multi-task learning with grouped tasks. In this model the tasks in each group share a group-specific prior and all group priors share one global prior. The model extends the model of Schwaighofer et al. (2005) by inserting an additional layer for the group prior. The parameters $(\mu, \mathbf{K})$ of a global Gaussian process prior are again drawn from a normal-inverse Wishart hyperprior. Following the same reasoning as in the previous section,

functions are defined over a discrete input domain of all training input locations. Such functions can be described by vectors over function values. For each group k, a vector of function values $\mathbf{g}_k$ is drawn from the global prior distribution $N(\mu, \mathbf{K})$. This function $\mathbf{g}_k$ is used as mean function for a group-specific Gaussian process prior $N(\mathbf{g}_k, \mathbf{K})$. From the latter the task functions $\mathbf{f}_{kz}$ are drawn. Single elements in this function vector are denoted by f_{kzi}. The complete generative model is the following:

1. Draw mean vector and covariance matrix once from a normal-inverse Wishart hyperprior, $(\mu, \mathbf{K}) \sim N(\mu|\mathbf{0}, \frac{1}{\pi}\mathbf{K})IW(\mathbf{K}|\tau, \kappa^{-1})$, given fixed scalars π, τ, and matrix κ;

2. For each group k independently draw vector with function values $\mathbf{g}_k \sim N(\mu, \mathbf{K})$;

3. For each task z in group k independently draw vector with function values $\mathbf{f}_{kz} \sim N(\mathbf{g}_k, \mathbf{K})$;

4. For all tasks z and all input vectors $\mathbf{x}_i$ from L_z, $y_i = f_{kzi} + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$.

We want to find the model parameters with maximum posterior probability given the data and the hyperprior parameters. This posterior is shown on the left hand side of Equation 4.21. The posterior is proportional to the joint distribution of prior parameters and the vector over all outputs $\mathbf{y}$ given the matrix over all input vectors $\mathbf{X}$ and the hyperprior parameters (right hand side of Equation 4.21). Equation 4.22 expands this joint distribution according to the generative process. For parameter inference we maximize Equation 4.22 with respect to the prior parameters by using the expectation maximization updates given in Theorem 4.1.

$$
\begin{aligned}
&\log p(\mu, \mathbf{g}_k, \mathbf{K}|L, \sigma^2, \pi, \tau, \kappa) \\
&\quad\propto\quad \log p(\mu, \mathbf{g}_k, \mathbf{K}, \mathbf{y}|\mathbf{X}, \sigma^2, \pi, \tau, \kappa) && (4.21) \\
&\quad=\quad \sum_k \sum_{\substack{z\in \\ \text{group } k}} \log \int N(\mathbf{f}_{kz}|\mathbf{g}_k, \mathbf{K}) \prod_{i\in L_z} N(y_i|f_{kzi}, \sigma^2) d\mathbf{f}_{kz} && (4.22) \\
&\quad+\quad \log N(\mu|0, \frac{1}{\pi}\mathbf{K}) + \log IW(\mathbf{K}|\tau, \kappa^{-1}) + \sum_k \log N(\mathbf{g}_k|\mu, \mathbf{K})
\end{aligned}
$$

Theorem 4.1 *The EM update steps for finding a local optimum of Equation 4.22 are the following:*

- ***E-step**: Compute expectation and covariance matrix of hidden variables for all*

groups k and tasks z,

$$\begin{aligned}\mathbf{E}\left[\mathbf{f}_{kz}\right] &= \mathbf{K}_{*,kz}\left(\mathbf{K}_{kz,kz}+\sigma^2\mathbf{I}\right)^{-1}\left(\mathbf{y}_{kz}-\mathbf{g}_{kz}\right)+\mathbf{g}_k,\\ \mathrm{Cov}[\mathbf{f}_{kz}] &= \mathbf{K}-\mathbf{K}_{*,kz}\left(\mathbf{K}_{kz,kz}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{K}_{*,kz}^{\mathsf{T}}.\end{aligned}$$

- ***M-step**: Compute new mean functions for all groups k,*

$$\mathbf{g}_k = \frac{\sum\limits_{z\in\text{group } k}\mathbf{E}\left[\mathbf{f}_{kz}\right]+\mu}{|z\in\text{group } k|+1},$$

and compute new prior parameters,

$$\begin{aligned}\mu &= \frac{\sum_k \mathbf{g}_k}{|\text{groups}|+\pi},\\ \mathbf{K} &= \left(\sum_k\sum_{z\in\text{group } k}\left(\mathrm{Cov}[\mathbf{f}_{kz}]+\left(\mathbf{E}[\mathbf{f}_{kz}]-\mathbf{g}_k\right)\left(\mathbf{E}[\mathbf{f}_{kz}]-\mathbf{g}_k\right)^{\mathsf{T}}\right)\right.\\ &\quad \left.+\tau\kappa+\sum_k\left(\mathbf{g}_k-\mu\right)\left(\mathbf{g}_k-\mu\right)^{\mathsf{T}}\right)\frac{1}{\tau+|\text{tasks}|+|\text{groups}|}.\end{aligned}$$

The proof can be found in Appendix C.

As discussed in the previous section, predictions can be obtained in a transductive manner by including the test examples in the inference process with labels set to zero or by using the out-of-sample extension proposed in Yu et al. (2005). The model can be easily extended to deeper hierarchies of tasks.

4.3 Overview on other Multi-Task Models

One obvious strategy for multi-task learning is to learn independent models for each task z by minimizing an appropriate loss function on the portion of $L_z = \{(\mathbf{x}_i, y_i, z_i) \in L : z_i = z\}$. The other extreme could be a one-size-fits-all model $f_*(\mathbf{x})$ trained on the entire sample L.

Beyond these two simple approaches and beside hierarchical Bayesian models several other methods for multi-task learning are studied in the machine learning community. In one of the earliest papers Caruana (1997) studies neural network models in which tasks share hidden nodes. He also describes a regression model that is learned by optimizing an objective on the target task's data combined with a down-weighted objective over data from auxiliary tasks. In a similar approach by Wu and Dietterich (2004) a

fixed scalar weight controls the influence of examples from one auxiliary task in the objective of a support vector machine. The differences to the model that we derive in Section 4.4 are that we introduce weights on the instance-level instead of the task-level and we derive weights to match the distribution of the target task.

The model of Liao et al. (2005) decreases the influence of instances from an auxiliary task that incur a large loss in a model trained over data from both (target and auxiliary) tasks. Intuitively, this heuristic should down-weight auxiliary instances that disagree with the target instances. Kaski and Peltonen (2007) make the assumption that the target data are entirely drawn from a target model and the data from auxiliary tasks are drawn from a mixture of the target model and auxiliary task specific models. The mixture weights are estimated along with all model parameters. This model has the advantage that each auxiliary task can have a different influence on the target model depending on the similarity of the respective auxiliary task to the target task. Very dissimilar auxiliary tasks can even have no influence on the target model.

In many applications, task-level descriptions or prior knowledge on task similarity encoded in a task kernel are available. Bonilla et al. (2007) study an extension of the one-size-fits-all model and find that training with a kernel defined as the multiplication of an input feature kernel and a task-level kernel outperforms a gating network. Task-level features have also been utilized for task clustering and for a task-dependent prior on the model parameters (Bakker and Heskes, 2003). Instead of relying on prior knowledge on task similarity, the model of Bonilla et al. (2008) explicitly estimates inter-task correlations in a Gaussian process prior.

In another line of research a common transformation matrix is learned that maps the input features from all tasks into a latent space (Ando and Zhang, 2005; Zhang et al., 2005). In the transformed space each task is treated independently as a separate learning problem.

4.4 Multi-Task Learning by Distribution Matching

All models for multi-task learning described in the previous sections make assumptions on the relationship between tasks. Usually, some shared prior or other shared model structure is assumed. In this section we develop a new model that does not require any assumption on the relationship between tasks. The model is based on rescaling weights that match the mixture distribution over all tasks to the distribution of any specific task (Section 4.4.1). A reformulation of the rescaling weights results in a discriminative expression (Section 4.4.2) that can be estimated with logistic regression (Section 4.4.3).

After the estimation of rescaling weights a target model on the weighted data from all tasks is learned (Section 4.4.4).

4.4.1 Definition of Rescaling Weights

In learning a classifier $f_t(\mathbf{x})$ for target task t, we seek to minimize the loss function with respect to $p(\mathbf{x}, y|t)$. Both, t and z are values of the random variable *task*; value t identifies the current target task. Simply pooling the available data for all tasks would create a sample governed by $\sum_z p(z)p(\mathbf{x}, y|z)$. Our approach now is to create a task-specific rescaling weight $r_t(\mathbf{x}, y)$ for each element of the pool of examples. The rescaling weights match the pool to the target distribution $p(\mathbf{x}, y|t)$. The weighted sample is governed by the correct target distribution, but is still larger as it draws from the sample pool over all tasks. Instead of rescaling each example in the pool, one can resample the pool based on the rescaling weights. The expected weighted loss with respect to the mixture distribution that governs the pool equals the loss with respect to the target distribution $p(\mathbf{x}, y|t)$. Equation 4.23 defines the rescaling weights.

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t)}[\ell(f(\mathbf{x},t),y)] \quad = \quad \mathbf{E}_{(\mathbf{x},y)\sim\sum_z p(z)p(\mathbf{x},y|z)}\left[r_t(\mathbf{x},y)\ell(f(\mathbf{x},t),y)\right]$$

In the following, we will show that

$$r_t(\mathbf{x}, y) = \frac{p(\mathbf{x}, y|t)}{\sum_z p(z)p(\mathbf{x}, y|z)}$$

satisfies Equation 4.23. Equation 4.23 expands the expectation and introduces a fraction that equals one. Equation 4.24 expands the sum over z in the numerator to run over the entire expression because the integral over $(\mathbf{x}, y)$ is independent of z. Equation 4.25 is the expected loss over the distribution of all tasks weighted by $\frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$.

$$\begin{aligned}
&\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t)}[\ell(f(\mathbf{x},t),y)] \\
&= \int \frac{\sum_z p(z)p(\mathbf{x},y|z)}{\sum_{z'} p(z')p(\mathbf{x},y|z')} p(\mathbf{x},y|t)\ell(f(\mathbf{x},t),y)d\mathbf{x}dy && (4.23)\\
&= \int \sum_z \left(p(z)p(\mathbf{x},y|z)\frac{p(\mathbf{x},y|t)}{\sum_{z'} p(z')p(\mathbf{x},y|z')}\ell(f(\mathbf{x},t),y)\right) d\mathbf{x}dy && (4.24)\\
&= \mathbf{E}_{(\mathbf{x},y)\sim\sum_z p(z)p(\mathbf{x},y|z)}\left[\frac{p(\mathbf{x},y|t)}{\sum_{z'} p(z')p(\mathbf{x},y|z')}\ell(f(\mathbf{x},t),y)\right] && (4.25)
\end{aligned}$$

Equation 4.25 signifies that we can train a hypothesis for task t by minimizing the expected loss over the distribution of all tasks weighted by $r_t(\mathbf{x}, y)$. This amounts to minimizing the expected loss with respect to the target distribution $p(\mathbf{x}, y|t)$.

Equation 4.25 leaves us with the problem of estimating the joint density ratio $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$. One might be tempted to train density estimators for $p(\mathbf{x}, y|t)$ and $\sum_z p(z)p(\mathbf{x}, y|z)$. However, obtaining estimators for potentially high-dimensional densities is unnecessarily difficult because ultimately only a scalar weight is required for each example.

4.4.2 Discriminative Formulation of Weights

In this section, we derive a discriminative model that directly estimates the rescaling weights $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ without estimating the individual densities. We reformulate the density ratio $\frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ in terms of a conditional model $p(t|\mathbf{x}, y)$. This conditional has the following intuitive meaning: Given that an instance $(\mathbf{x}, y)$ has been drawn at random from the pool $\bigcup_z L_z = L$ of samples for all tasks (including L_t); the probability that $(\mathbf{x}, y)$ originates from L_t is $p(t|\mathbf{x}, y)$. The following equations assume that the prior on the size of the target sample is greater than zero, $p(t) > 0$. In Equation 4.27 Bayes' rule is applied twice and in Equation 4.28 $p(\mathbf{x}, y)$ and $p(z)$ are canceled out. Equation 4.29 follows by $\sum_z p(z|\mathbf{x}, y) = 1$.

$$
\begin{aligned}
r_t(\mathbf{x}, y) &= \frac{p(\mathbf{x}, y|t)}{\sum_z p(z)p(\mathbf{x}, y|z)} && (4.26)\\
&= \frac{p(t|\mathbf{x}, y)p(\mathbf{x}, y)}{p(t)} \frac{1}{\sum_z p(z)\frac{p(z|\mathbf{x},y)p(\mathbf{x},y)}{p(z)}} && (4.27)\\
&= \frac{p(t|\mathbf{x}, y)}{p(t)\sum_z p(z|\mathbf{x}, y)} && (4.28)\\
&= \frac{p(t|\mathbf{x}, y)}{p(t)} && (4.29)
\end{aligned}
$$

The significance of Equation 4.29 is that it shows how the rescaling weights $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ can be determined without knowledge of any of the task densities $p(\mathbf{x}, y|z)$. The right hand side of Equation 4.29 can be evaluated based on a model $p(t|\mathbf{x}, y)$ that discriminates labeled instances of the target task against labeled instances of the pool of examples for all tasks. Intuitively, $p(t|\mathbf{x}, y)$ characterizes how much more likely $(\mathbf{x}, y)$ is to occur in the target distribution than it is to occur in the mixture distribution of all tasks. Instead of potentially high-dimensional densities $p(\mathbf{x}, y|t)$ and $p(\mathbf{x}, y|z)$, a conditional distribution with a single variable needs to be modeled. One can apply any probabilistic classifier to model this conditional distribution.

4.4.3 Logistic Model for Weights

We model $p(t|\mathbf{x}, y)$ of Equation 4.29 for all tasks jointly with a multinomial logistic model (the multi-class generalization of the logistic model, also known as soft-max model) with model parameters $\mathbf{v}$, displayed in Equation 4.30. The parameter vector $\mathbf{v}$ is a concatenation of task-specific subvectors $\mathbf{v}_z$, one for each task z. With this model an estimate for $p(t|\mathbf{x}, y)$ is given by $p(z = t|\mathbf{x}, y, \mathbf{v})$; this is the evaluation of the multinomial logistic model with respect to task t.

$$p(z|\mathbf{x}, y, \mathbf{v}) \quad = \quad \frac{\exp(\mathbf{v}_z^\mathsf{T}\Phi(\mathbf{x}, y))}{\sum_{z'} \exp(\mathbf{v}_{z'}^\mathsf{T}\Phi(\mathbf{x}, y))} \tag{4.30}$$

Equation 4.30 requires a problem-specific feature mapping $\Phi(\mathbf{x}, y)$. We define this mapping for binary labels $y \in \{1, -1\}$ in Equation 4.31; δ is the Kronecker delta. In the absence of prior knowledge about the similarity of classes, input features $\mathbf{x}$ of examples with different class labels y are mapped to disjoint subsets of the feature vector.

$$\Phi(\mathbf{x}, y) \quad = \quad \begin{bmatrix} \delta(y, 1)\Phi(\mathbf{x}) \\ \delta(y, -1)\Phi(\mathbf{x}) \end{bmatrix} \tag{4.31}$$

With this feature mapping the models for positive and negative examples do not interact and can be trained independently.

For training of the multinomial logistic model we maximize the regularized log-likelihood of the data. Prior knowledge on the similarity of tasks in the form of a positive semi-definite kernel function $k(z, z')$ can be be encoded in the covariance matrix of a Gaussian prior $N(0, \Sigma)$ on parameter vector $\mathbf{v}$. We set all main diagonal entries of Σ to the scalar parameter $\sigma_\mathbf{v}^2$ and set the secondary diagonal entries corresponding to the covariances between $\mathbf{v}_z$ and $\mathbf{v}'_z$ to $k(z, z')\rho\sigma_\mathbf{v}^2$ (assuming kernel values $0 \leq k(z, z') \leq 1$). Parameter $\sigma_\mathbf{v}^2$ specifies the variance of each element in $\mathbf{v}$. $k(z, z')\rho$ is the correlation coefficient between elements of subvectors $\mathbf{v}_z$ and $\mathbf{v}'_z$; parameter ρ specifies the strength of this correlation. The covariance matrix Σ is required to be invertible and therefore $0 \leq \rho < 1$. All other entries of Σ are set to zero. When prior knowledge on the task similarities is encoded in the prior on the model parameters, then this prior knowledge dominates the optimization criterion for small samples while the data-driven portion of the criterion becomes dominant and overrides prior beliefs as more data arrives.

Optimization Problem 4.2 *Over parameters* $\mathbf{v}$*, maximize*

$$\sum_{(\mathbf{x}_i,y_i,z_i)\in L} \log p(z_i|\mathbf{x}_i,y_i,\mathbf{v}) - \mathbf{v}^\mathsf{T}\Sigma^{-1}\mathbf{v}.$$

The solution of Optimization Problem 4.2 is a *maximum a posteriori* estimation of the multinomial logistic model (Equation 4.30) over the model parameters $\mathbf{v}$ using a log-Gaussian prior with covariance matrix Σ. Tasks with no training examples are covered naturally in Optimization Problem 4.2. In this case, the Gaussian prior with the task kernel $k(z,z')$ encoded in the covariance matrix determines the model.

For our experiments we use a kernelized variant of Optimization Problem 4.2 by applying the representer theorem. Details on the kernelization of multi-class logistic regression can be learned from Zhu and Hastie (2002).

4.4.4 Weighted Empirical Loss and Target Model

The multi-task learning procedure first determines rescaling weights $r_z(\mathbf{x},y)$ for all tasks and instances by solving Optimization Problem 4.2. In this section we describe the second step of training an array of target models, one for each task, using weighted examples. This two-stage procedure for multi-task learning corresponds to the two-stage approximation for learning under covariate shift described in Section 3.5.

With the results of Optimization Problem 4.2 the discriminative expression for the weights of Equation 4.29 can be estimated. Using these weights we can evaluate the expected loss over the weighted training data as displayed in Equation 4.32. It is the regularized empirical counterpart of Equation 4.25.

$$\mathbf{E}_{(\mathbf{x},y)\sim L}\left[\frac{p(t|\mathbf{x},y,\mathbf{v})}{p(t)}\ell(f(\mathbf{x},t),y)\right] + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2} \tag{4.32}$$

An instance of Optimization Problem 4.3 is solved for each task independently to produce a separate model for this task. Optimization Problem 4.3 minimizes Equation 4.32, the weighted regularized loss over the training data using a standard log-Gaussian prior with variance $\sigma_\mathbf{w}^2$ on the parameters $\mathbf{w}_t$. Each example is weighted by the discriminatively estimated density fraction from Equation 4.29 using the solution of Optimization Problem 4.2.

Optimization Problem 4.3 *For task* t*: over parameters* $\mathbf{w}_t$*, minimize*

$$\frac{1}{|L|}\sum_{(\mathbf{x}_i,y_i)\in L}\frac{p(t|\mathbf{x}_i,y_i,\mathbf{v})}{p(t)}\ell(f(\mathbf{x}_i,\mathbf{w}_t),y_i) + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2}.$$

4.5 Case Study: HIV Therapy Screening

In this section we conduct a case study on multi-task learning for the problem of predicting the therapeutic success of a given combination of drugs for a given strain of the *Human Immunodeficiency Virus-1* (HIV-1). HIV is associated with the *acquired immunodeficiency syndrome* (AIDS). Being a disease that claimed more than 25 million lives since 1981, AIDS is one of the most destructive epidemics in recorded history. Currently there are more than 33 million people infected with HIV (UNAIDS/WHO, 2007).

Antiretroviral therapy is hampered by HIV's strong ability to mutate and develop viral quasi-species that can quickly be dominated by resistant variants. In order to decide on a course of therapy, virus samples taken from each individual patient are tested for a set of resistance-relevant mutations. Given this set of identified mutations together with the patient's medication history, a medical practitioner needs to decide which combination of drugs to administer. The large number of genetic mutations and the wide array of available drug combinations render the process of predicting the success of a potential therapy difficult, at best, for a human doctor.

Historic treatment records of HIV patients cover only a small portion of all possible drug combinations. For many of these combinations, only few treatments have been recorded. This scarceness of training data precludes separate training of a powerful prediction model for each combination from only records of treatments which used the same drug combination. Distinct combinations can have similar effects when they intersect in jointly contained drugs, or when they include drugs that use similar mechanisms to affect the virus. Therefore, in order to predict the outcome of a given drug combination, it is desirable to exploit data from related combinations and thereby achieve generalization over both virus mutations and combinations of drugs.

Prior approaches to HIV therapy screening do not account for the similarities of drug combinations in a principled way. Larder et al. (2007) tackle the problem of predicting virological response to a given HIV drug combination with neural networks. Lathrop and Pazzani (1999) apply combinatorial optimization to the same problem using features extracted from the viral genotype and the drugs in the combination. Altmann et al. (2007) approach the problem by including various phenotypic information and an estimate of future evolutionary development of the virus in the learning process.

The application problem of HIV therapy screening instantiates the abstract problem setting of *multi-task learning* as follows. Input $\mathbf{x}$ describes the genotype of the virus that

a patient carries, together with the patient's treatment history. Genotype information is encoded as a binary vector indicating the presence and absence of each out of a predefined set of resistance-relevance mutations, respectively. The treatment history can be represented as a binary vector indicating which drugs have been administered over the course of past treatments. A candidate combination of drugs plays the role of the task z: each task has an associated binary vector that indicates a set of drugs that a medical practitioner is currently giving consideration. The binary class label y indicates whether the therapy will be successful.

In addition to training data, we have prior knowledge on the similarity of tasks which is encoded in a kernel function $k(z, z')$. Prediction models for different drug combinations can be similar because the sets of drugs intersect (we will later refer to this as the drug feature kernel), or because similar sets of mutations in the virus render the drugs in the set ineffective (mutation table kernel).

In the next subsections we describe the data sets (Section 4.5.1), reference methods (Section 4.5.2), experimental setup (Section 4.5.3), and the empirical results of our study (Section 4.5.4).

4.5.1 Data Sets and Prior Knowledge on Task Similarity

We use data from the EuResist project (Rosen-Zvi et al., 2008). The data set comprises a total number of 52846 treatment records from the treatment histories of 16999 HIV patients treated in hospitals in the period of 1977 through 2007.

We use two different definitions of therapeutic success and failure to tag the data: *virus load labeling* and *multi-conditional labeling*.

According to our *virus load labeling* definition a therapy is successful if the viral load (number of virus copies per ml blood plasma, cp/ml) drops below the established level of virus detection of 400 cp/ml during the time of the treatment. Otherwise the treatment is a failure. In *multi-conditional labeling*, a therapy is successful if the viral load measured in the time range between 28 and 84 days after the start of the therapy decreases by at least 2 orders of magnitude compared to the most recent viral load measured one to three months before the start of the therapy, or the viral load drops below 400 cp/ml 56 days after the start of the therapy. A drawback of this definition is that due to the strict time intervals it imposes on the measurements, class labels that adhere to this labeling are only available for a small number of records. The *virus load labeling* does not require these strict time intervals by making use of any viral load measurement during the course of therapy to label it.

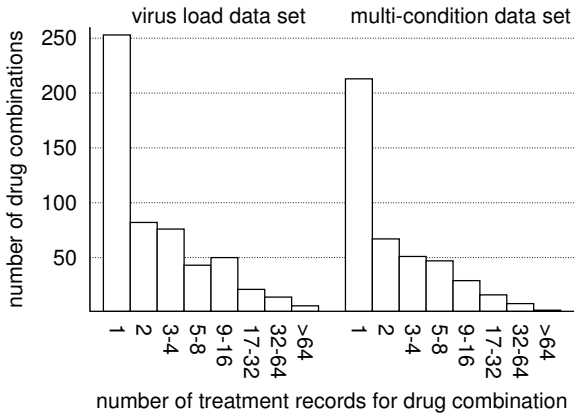

Figure 4.1: Histogram over number of treatment records for drug combinations (tasks) in the virus load data set (left) and multi-condition data set (right).

Out of all available treatment records we extract two different data sets using the two labelings. With the virus load labeling we extract 3260 and with the multi-conditional labeling 2011 treatment records with corresponding ratios of 65.7% and 64.1% successful treatments. The size of these data sets is much smaller than the size of the original data due to missing viral load measurements, or missing virus sequence information.

A number of 545 distinct drug combinations (tasks z) occur at least once in the virus load data set; 433 occur in the the multi-conditional data set. The histogram over sample sizes per task is displayed in Figure 4.1. For many combinations, only a few examples occur in the data. For instance, in the virus load data set we observe 253 out of 545 drug combinations with only one data point and 411 with less than 5 instances. Similarly, the multi-conditional data set has 213 out of 433 drug combinations with a single data point and 331 with less than 5 observations.

We extract two types of features for each instance: a genotypic description of the virus and information about the treatment history of the patient. We use the viral genotype taken from the patient shortly before the treatment and represent it by a binary vector indicating the presence of resistance-relevant mutations of the viral sequence (Johnson et al., 2007). Drug-resistant viral quasi-species evolve during the course of the treatment due to selective pressure imposed by the drug. As they remain in the patient's body, the treatment history plays an important role for predicting the outcome of a potential treatment. Hence, we extract all drugs given to the patient in previous treatments and use a binary vector representation with a one entry for each drug given to the patient in the treatment history. The 82-dimensional feature vector $\mathbf{x}$ for each data point results from the concatenation of 65 genotypic and 17 historic

treatment features.

We have prior knowledge about the similarity of combinations and encode this knowledge into two different task similarity kernels $k(z, z')$. The binary drug indicator vector has an entry for each drug; entries of one indicate the presence of a drug in the combination. The *drug indicator kernel* is the inner product between the normalized drug indicator vectors of two combinations. The *mutation table kernel* is based on tables about the resistance-associated mutations of single drugs (Johnson et al., 2007). We construct binary vectors indicating resistance-relevant mutations for the set of drugs occurring in a combination. The kernel computes the normalized inner product between such binary vectors for two drug combinations.

4.5.2 Reference Methods

The first reference method is training of a separate logistic regression model for each task without any interaction ("separate"). Tasks without any training examples get a constant classifier that assigns each test example with 50% to each of both classes.

The next baseline is a one-size-fits-all model; all examples are pooled and only one common logistic regression is trained for all tasks ("pooled"). For the experiments with prior knowledge on task similarity we multiply the feature kernel with the task kernel values $k(\mathbf{x}, \mathbf{x}')(k(z, z') + 1)$ and train one model using this kernel (Bonilla et al., 2007). We include a "+1" term to ensure that the feature kernel does not vanish.

The third reference method ("hier. Bayes kernel") is a logistic regression with the hierarchical Bayesian kernel of Equation 4.10 described in Section 4.2.1. For the experiments with task similarity kernel the hierarchical Bayesian and the task kernel are multiplied. As second hierarchical Bayesian method ("hier. Bayes Gauss. proc.") we use the Gaussian process regression of Section 4.2.3.

Last but not least we apply our new nested hierarchical Bayesian model for Gaussian processes ("nested hier. Bayes Gauss. proc.") that requires prior knowledge on task groups (Section 4.2.4). We use the country of residence of patients as grouping criterion. Thus, the three groups in our data are Italy, Germany, and Sweden. Our intuition is that the effect of treatments on the virus and the health of patients differs between countries because of different overall health care conditions, different education of doctors, or different climate conditions. We also considered the reversed structure with one group for each drug combination and each group has three sub-tasks, one for each country. We do not report on these results because initial experiments revealed that the model structure with grouping by country achieves better results.

4.5.3 Experimental Setup

In our experiments we study the benefit of distribution matching for HIV therapy screening compared to the reference methods described in Section 4.5.2. Optimization Problem 4.2 is solved with limited-memory BFGS and Optimization Problem 4.3 with Newton gradient descent using a logistic loss. For the prior term $p(t)$ required in Optimization Problem 4.3 we use a MAP estimate $\frac{|L_t|+\gamma}{\sum_z(|L_z|+\gamma)}$ with a symmetric Dirichlet prior. We use RBF kernels for all methods.

We apply a training-test split of the data consistent with the dates of the treatment records. We sort the treatment records by date and use the first 80% of the records as training data and the last 20% as test data. This procedure yields 653 and 403 test examples for the virus load and multi-conditional data set, respectively. The date consistent split is necessary because new drugs get approved over time, and under pressure of new drugs the viral population evolves. In such environments, the prediction models should be able to learn from data seen in the *past* and perform well on unseen data in the *future*.

We tune the prior and regularization parameters of all methods, the Dirichlet parameter γ, and the variance of the RBF kernels on tuning data resulting from a date consistent split of the training data.

4.5.4 Results

The evaluation measure is the accuracy of predicting the correct label (success or failure of a treatment) on the test set. Table 4.1 shows the results of the prediction accuracy for all methods over both data sets without and with two different types of prior knowledge on combination similarity. The columns "se.Δ" placed next to the accuracy columns display the standard error of the differences to the distribution matching method.

Multi-task learning by distribution matching outperforms, or is as good as, the best alternative method in all cases. The improvement over the separate model baseline is about 10-14%. We can reject the null hypothesis that the pooled and the hierarchical Bayesian kernel baseline is at least as accurate as distribution matching in four and five cases respectively out of six according to a paired t-test at $\alpha = 0.05$. The differences between the hierarchical Bayesian Gaussian process and our new nested variant with grouping by country are not significant. We assume the differences between countries are not distinct enough.

For distribution matching, prior knowledge does not improve the accuracy. The pooled baseline benefits from prior knowledge for the multi-condition data set. For

Table 4.1: Classification accuracies with standard errors of differences to distribution matching method (se.Δ). Three different types of prior knowledge are used (none, drug.feat., or mut.table). Symbols (•,○,∗,◇,▷) indicate statistical significance according to a paired t-test with significance level $\alpha = 0.05$, (•) compared to separate baseline, (○) compared to pooled baseline, (∗) compared to hierarchical Bayesian kernel baseline, (◇) compared to hierarchical Bayesian Gaussian process baseline, (▷) compared to nested hierarchical Bayesian Gaussian process baseline. Results for the Gaussian process models with prior knowledge are not reported because it is unclear how to exploit this knowledge in a principled way for these models.

virus load data set

	prior knowledge					
method	none	se.Δ	drug.feat.	se.Δ	mut.table	se.Δ
separate	67.87%	1.80	67.87%	1.76	67.87%	1.78
pooled	75.00%	1.47	75.46%	1.39	75.61%	1.37
hier. Bayes kernel	76.69%	1.39	75.31%	1.34	76.84%	1.16
hier. Bayes Gauss. proc.	75.92%	1.32	–	–	–	–
nested hier. Bayes Gauss. proc.	76.07%	1.30	–	–	–	–
distribution matching	**79.14%** • ○ ∗ ◇ ▷		**77.91%** • ○ ∗		**79.29%** • ○ ∗	

multi-condition data set

	prior knowledge					
method	none	se.Δ	drug.feat.	se.Δ	mut.table	se.Δ
separate	64.64%	2.41	64.64%	2.29	64.64%	2.38
pooled	76.67%	1.13	**78.41%**	1.63	**78.66%**	1.11
hier. Bayes kernel	77.17%	1.29	75.19%	1.44	**77.42%**	1.24
hier. Bayes Gauss. proc.	76.43%	1.35	–	–	–	–
nested hier. Bayes Gauss. proc.	76.43%	1.44	–	–	–	–
distribution matching	**79.40%** • ○ ∗ ◇ ▷		**78.16%** •	∗	**79.16%** •	

the case without prior knowledge we do not observe a statistically significant difference between the three hierarchical Bayesian methods, but they are all significantly worse than distribution matching according to the paired t-test. We do not report on results of the Gaussian process models with prior knowledge on task similarity because it is unclear how to exploit this knowledge in a principled way. Note that the Gaussian process methods are regression models; all other methods are classification models.

Figure 4.2 displays the accuracy over the combinations in the test set grouped by the number of available examples for the settings without and with the mutation table kernel. For instance, an accuracy of 74% for the first group "0-2" means, that only test

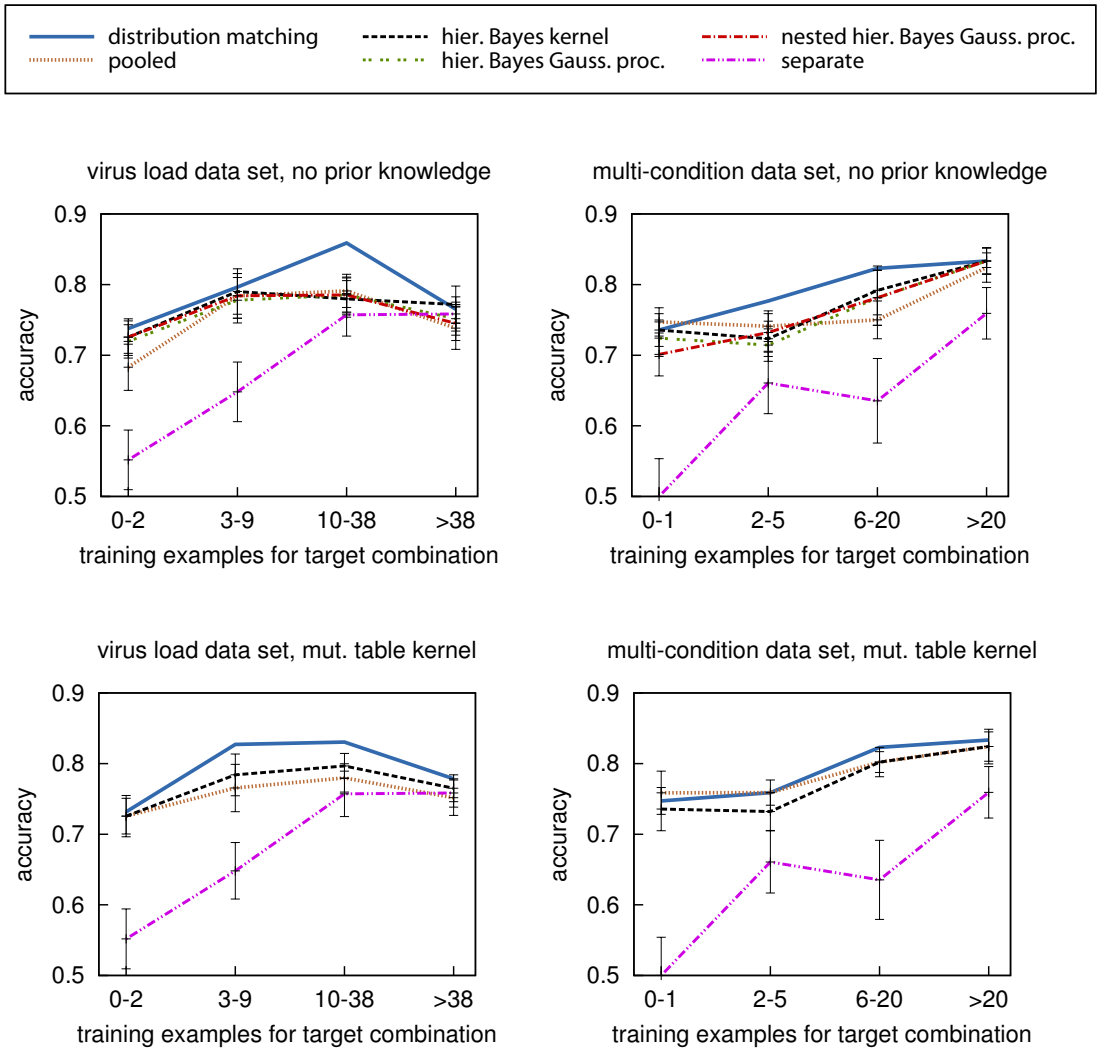

Figure 4.2: Accuracy over different number of training examples for target combination; virus load data set (left), multi-condition data set (right). Error bars indicate the standard error of the differences to distribution matching. The key can be found in the box right above the diagrams.

examples from combinations are selected that have zero, one, or two training examples each, and the accuracy on this subset of the test examples is 74%. Each of the four groups covers about the same number of test examples. The error bars indicate the standard error of the differences to the distribution matching method. Note, that the statistical tests described above are based on all test data and are not directly related to the group-specific error bars in the diagrams.

All methods benefit from larger numbers of training examples per drug combination. The slightly decreasing accuracy for the virus load data set with ">38" training exam-

ples is surprising. Further analysis reveals that in this case there is an accumulation of test examples with history profiles very different from the training examples of the same combination.

For all methods that generalize over the tasks the benefit compared to the separate model baseline is the largest for the smallest group ("0-2" and "0-1" training examples respectively).

4.6 Conclusion

We showed that the well known multi-task model of Evgeniou and Pontil (2004) can be directly derived from a hierarchical Bayesian model when the hinge loss is replaced with a logistic or squared loss. Similarly, we discovered that a standard taxonomy classification model based on a conditional random field can be derived from a nested hierarchical Bayesian model. This derivation opens up the way for more flexible taxonomy classification models. We derived a nested hierarchical Bayesian model for Gaussian processes that can be applied to multi-task learning with grouped tasks.

We devised a multi-task learning method that centers around rescaling weights which match the distribution of the pool of examples of multiple tasks to the target distribution for a given task at hand. The method creates a weighted sample that reflects the desired target distribution and exploits the entire corpus of training data for all tasks. We showed how appropriate weights can be obtained by discriminating the labeled sample for a given task against the pooled sample. After weighting the pooled sample, a classifier for the given task can be trained. In our experiments on HIV therapy screening we found that the distribution matching method improves on the prediction accuracy over independently trained models by 10-14%. According to a paired t-test, distribution matching is significantly better than the reference methods for 19 out of 22 experiments.

A combination of drugs is the standard way of treating HIV patients. The accuracy to which the likely outcome of a combination therapy can be anticipated can therefore directly impact the quality of HIV treatments.

5 Multi-Task Learning under Covariate Shift

In the previous chapter we studied multi-task learning where the training and the test data for each task are both drawn from an identical task-specific distribution. We now consider the case when the training data for each task is drawn from a training distribution distinct from the test distribution. For each task, small—possibly even empty—labeled samples and large unlabeled samples are available. While the unlabeled samples reflect the target distribution, the labeled samples may be biased. This setting naturally arises when the labels for each task are collected by a selective process such as by questionnaires that can be ignored or refused by potential participants of a survey.

We derive a solution that produces rescaling weights which match the pool of all training examples to the test distribution of any given task. The conceptual idea is similar to the distribution matching methods of Chapters 3 and 4 but here, covariate shift within each task and joint input-output shift across tasks need to be accounted for.

The outline of this chapter is the following. We begin with a definition of the problem setting in Section 5.1. In Section 5.2 we devise a distribution matching method that accounts for covariate shift and joint input-output shift. A case study on targeted advertising is presented in Section 5.3. Section 5.4 concludes the chapter.

5.1 Problem Setting

We consider the following multi-task learning scenario. Each of several tasks z is characterized by an unknown joint distribution $p(\mathbf{x}, y|z, \theta) = p(\mathbf{x}|z, \theta)p(y|\mathbf{x}, z, \theta)$ over features $\mathbf{x}$ and labels y given the task z. The joint distributions of different tasks may differ arbitrarily but usually some tasks have similar distributions. An unlabeled test sample $T = \langle(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_m, z_m)\rangle$ with examples from different tasks is available. For each test example, attributes $\mathbf{x}_i$ and the originating task z_i are known. The test data for task z are governed by $p(\mathbf{x}|z, \theta)$.

A labeled training set $L = \langle(\mathbf{x}_{m+1}, y_{m+1}, z_{m+1}), \ldots, (\mathbf{x}_{m+n}, y_{m+n}, z_{m+n})\rangle$ collects examples from several tasks. In addition to $\mathbf{x}_i$ and z_i, the label y_i is known for each example. The training data for task z is drawn from a joint distribution $p(\mathbf{x}, y|z, \lambda) = p(\mathbf{x}|z, \lambda)p(y|\mathbf{x}, z, \lambda)$ that may differ from the test distribution in terms of the marginal distribution $p(\mathbf{x}|z, \lambda)$. The training and test marginals may differ arbitrarily, as long as each $\mathbf{x}$ with positive $p(\mathbf{x}|z, \theta)$ also has a positive $p(\mathbf{x}|z, \lambda)$. This guarantees that the training distribution covers the entire support of the test distribution for each task. The conditional distributions of test and training data are identical for a given task z, $p(y|\mathbf{x}, z, \theta) = p(y|\mathbf{x}, z, \lambda) = p(y|\mathbf{x}, z)$, but conditionals can differ arbitrarily between tasks. The entire training set over all tasks is governed by the mixed density $p(z|\lambda)p(\mathbf{x}, y|z, \lambda)$. The prior $p(z|\lambda)$ specifies the task proportions. There may be tasks with only a few or no labeled data.

The goal is to learn a hypothesis $f_z : \mathbf{x} \mapsto y$ for each task z. This hypothesis $f_z(\mathbf{x})$ should correctly predict the true label y of unseen examples drawn from $p(\mathbf{x}|z)$ for all z. That is, it should minimize the expected loss

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|z,\theta)}[\ell(f_z(\mathbf{x}), y)]$$

with respect to the unknown distribution $p(\mathbf{x}, y|z, \theta)$ for each individual z.

One reference strategy is to learn individual models for each target task z by minimizing an appropriate loss function on the portion of $L_z = \{(\mathbf{x}_i, y_i, z_i) \in L : z_i = z\}$. This procedure minimizes the loss with respect to $p(\mathbf{x}, y|z, \lambda)$; the minimum of this optimization problem will not generally coincide with the minimal loss on $p(\mathbf{x}, y|z, \theta)$ and if L_z is small or empty learning of the correct function is impossible. The other extreme is a one-size-fits-all model $f_*(\mathbf{x})$ trained on the entire training sample L.

In order to describe the following model accurately, we use the same selector variable s as in Section 3.3.1 which distinguishes training ($s=1$) from test distribution ($s=-1$). Symbol $p(\mathbf{x}, y|z, s=1)$ is a shorthand for $p(\mathbf{x}, y|z, s=1, \lambda) = p(\mathbf{x}, y|z, \lambda)$ (cf. Section 3.3.3); likewise, $p(\mathbf{x}, y|z, s=-1) = p(\mathbf{x}, y|z, \theta)$.

As far as we know we are the first to define this problem setting and to derive and study a solution. A related problem setting is semi-supervised multi-task learning (Liu et al., 2008).

5.2 Multi-Task Learning under Covariate Shift by Distribution Matching

In this section we derive a distribution matching method that is based on rescaling weights which match the pool of all examples to the test distribution of the target task. The rescaling weights are defined in Section 5.2.1. Section 5.2.2 provides a reformulation of the weights in terms of two discriminative expressions that can be estimated with logistic regression models (Section 5.2.3). The final target model with a weighted empirical loss over all training examples is described in Section 5.2.4.

5.2.1 Definition of Rescaling Weights

In learning a classifier $f_t(\mathbf{x})$ for target task t, we seek to minimize the loss function with respect to $p(\mathbf{x}, y|t, \theta) = p(\mathbf{x}, y|t, s=-1)$. Both, t and z are values of the random variable *task*; value t identifies the current target task. Simply pooling the available data for all tasks would create a sample governed by $\sum_z p(z|s=1)p(\mathbf{x}, y|z, s=1)$. Our approach is to create a task-specific rescaling weight $r_t(\mathbf{x}, y)$ for each element of the pool of examples. The rescaling weights match the pool distribution to the target distribution $p(\mathbf{x}, y|t, s=-1)$. The rescaled pool is governed by the correct target distribution, but is larger than the labeled sample of the target task. The following derivation of the rescaling weights is similar to Section 4.4.1 but the resulting weights are different because of the covariate shift in addition to the joint input-output shift.

The expected weighted loss with respect to the mixture distribution that governs the pool equals the loss with respect to the target distribution $p(\mathbf{x}, y|t, s=-1)$. Equation 5.1 defines the condition that the rescaling weights have to satisfy.

$$\begin{aligned} &\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t,s=-1)}[\ell(f(\mathbf{x},t),y)] \\ &= \mathbf{E}_{(\mathbf{x},y)\sim\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)}\left[r_t(\mathbf{x},y)\ell(f(\mathbf{x},t),y)\right] \end{aligned} \tag{5.1}$$

In the following, we will show that

$$r_t(\mathbf{x}, y) = \frac{p(\mathbf{x}, y|t, s=1)}{\sum_z p(z|s=1)p(\mathbf{x}, y|z, s=1)} \frac{p(\mathbf{x}|t, s=-1)}{p(\mathbf{x}|t, s=1)} \tag{5.2}$$

satisfies Equation 5.1. Equation 5.3 expands the expectation and introduces two fractions that equal one. We can factorize $p(\mathbf{x}, y|t, s=-1)$ and expand the sum over z in the numerator to run over the entire expression because the integral over $(\mathbf{x}, y)$ is independent of z (Equation 5.4). Equation 5.5 rearranges some terms and Equation 5.6

is the expected loss over the distribution of all tasks weighted by $r_t(\mathbf{x}, y)$.

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t,s=-1)}[\ell(f(\mathbf{x},t),y)]$$

$$= \int \frac{\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)}{\sum_{z'} p(z'|s=1)p(\mathbf{x},y|z',s=1)} \frac{p(\mathbf{x}|t,s=1)}{p(\mathbf{x}|t,s=1)} p(\mathbf{x},y|t,s=-1)\ell(f(\mathbf{x},t),y)d\mathbf{x}dy \tag{5.3}$$

$$= \int \sum_z \left(\frac{p(z|s=1)p(\mathbf{x},y|z,s=1)}{\sum_{z'} p(z'|s=1)p(\mathbf{x},y|z',s=1)} \frac{p(\mathbf{x}|t,s=1)}{p(\mathbf{x}|t,s=1)} p(\mathbf{x}|t,s=-1)p(y|\mathbf{x},t) \right. \tag{5.4}$$

$$\left. \ell(f(\mathbf{x},t),y) \right) d\mathbf{x}dy$$

$$= \int \sum_z \left(p(z|s=1)p(\mathbf{x},y|z,s=1) \frac{p(\mathbf{x}|t,s=1)p(y|\mathbf{x},t)}{\sum_{z'} p(z'|s=1)p(\mathbf{x},y|z',s=1)} \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)} \right. \tag{5.5}$$

$$\left. \ell(f(\mathbf{x},t),y) \right) d\mathbf{x}dy$$

$$= \mathbf{E}_{(\mathbf{x},y)\sim \sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)} \left[\frac{p(\mathbf{x},y|t,s=1)}{\sum_{z'} p(z'|s=1)p(\mathbf{x},y|z',s=1)} \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)} \ell(f(\mathbf{x},t),y) \right] \tag{5.6}$$

Equation 5.6 signifies that we can train a hypothesis for task t by minimizing the expected loss over the distribution of all tasks weighted by $r_t(\mathbf{x}, y)$. This amounts to minimizing the expected loss with respect to the target distribution $p(\mathbf{x}, y|t, s=-1)$. The rescaling weights of Equation 5.2 have the following intuitive interpretation. The first fraction accounts for the difference in the joint distributions across tasks and corresponds to the rescaling weights for multi-task learning introduced in Chapter 4. The second fraction accounts for the covariate shift within the target task and corresponds to the weights described in Chapter 3.

In order to apply Equation 5.6 we need to estimate the product of two density ratios $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t,s=1)}{\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)} \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)}$.

5.2.2 Discriminative Formulation of Weights

In this section, we derive a discriminative model that directly estimates the two factors $r_t^1(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t,s=1)}{\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)}$ and $r_t^2(\mathbf{x}) = \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)}$ of the rescaling weights $r_t(\mathbf{x}, y) = r_t^1(\mathbf{x}, y) r_t^2(\mathbf{x})$ without estimating the individual densities. The term $r_t^1(\mathbf{x}, y)$ captures the input-output shift and $r_t^2(\mathbf{x})$ the covariate shift between training and test distributions. Accordingly, the following derivation for $r_t^1(\mathbf{x}, y)$ resembles Section 4.4.2 and the one for $r_t^2(\mathbf{x})$ is similar to Section 3.3.3.

We reformulate the first density ratio $r_t^1(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t,s=1)}{\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)}$ in terms of a conditional model $p(t|\mathbf{x}, y, s=1)$. This conditional has the following intuitive meaning: Given that an instance $(\mathbf{x}, y)$ has been drawn at random from the pool distribution

$\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)$ over all tasks (including target task t); the probability that $(\mathbf{x},y)$ originates from $p(\mathbf{x},y|t,s=1)$ is $p(t|\mathbf{x},y,s=1)$. The following equations assume that the prior on the size of the target sample is greater than zero, $p(t|s=1)>0$. In Equation 5.7 Bayes' rule is applied to the numerator and z is summed out in the denominator. Equation 5.8 follows by dropping the normalization factor $p(t|s=1)$ and by canceling $p(\mathbf{x},y|s=1)$.

$$r_t^1(\mathbf{x},y) = \frac{p(\mathbf{x},y|t,s=1)}{\sum_z p(z|s=1)p(\mathbf{x},y|z,s=1)} = \frac{p(t|\mathbf{x},y,s=1)p(\mathbf{x},y|s=1)}{p(t|s=1)p(\mathbf{x},y|s=1)} \quad (5.7)$$

$$\propto p(t|\mathbf{x},y,s=1) \quad (5.8)$$

The significance of Equation 5.8 is that it shows how the first factor of the rescaling weights $r_t^1(\mathbf{x},y)$ can be determined without knowledge of any of the task densities $p(\mathbf{x},y|z,s=1)$. The right hand side of Equation 5.8 can be evaluated based on a model $p(t|\mathbf{x},y,s=1)$ that discriminates labeled instances of the target task against labeled instances of the pool of examples for all non-target tasks.

Similar to the first density ratio, the second density ratio $r_t^2(\mathbf{x})=\frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)}$ can be expressed using a conditional model $p(s=1|\mathbf{x},t)$. In Equation 5.9 Bayes' rule is applied twice. The two terms of $p(\mathbf{x}|t)$ cancel each other out, $p(s=1|t)/p(s=-1|t)$ is just a normalization factor, and since $p(s=-1|\mathbf{x},t)=1-p(s=1|\mathbf{x},t)$, Equation 5.10 follows.

$$r_t^2(\mathbf{x}) = \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)} = \frac{p(s=-1|\mathbf{x},t)p(\mathbf{x}|t)}{p(s=-1|t)}\frac{p(s=1|t)}{p(s=1|\mathbf{x},t)p(\mathbf{x}|t)} \quad (5.9)$$

$$\propto \frac{1}{p(s=1|\mathbf{x},t)}-1 \quad (5.10)$$

The significance of the above derivations is that instead of the four potentially high-dimensional densities in $r_t(\mathbf{x},y)$, only two conditional densities with binary variables (Equations 5.8 and 5.10) need to be estimated. One can apply any probabilistic classifier to this estimation.

5.2.3 Logistic Models for Weights

For the estimation of $r_t^1(\mathbf{x},y)$ we model $p(t|\mathbf{x},y,s=1)$ of Equation 5.8 with a logistic regression model

$$p(t|\mathbf{x},y,s=1,\mathbf{u}_t)=\frac{1}{1+\exp(-\mathbf{u}_t^\mathsf{T}\Phi(\mathbf{x},y))}$$

over model parameters $\mathbf{u}_t$ using a problem-specific feature mapping $\Phi(\mathbf{x},y)$. We define this mapping for binary labels as in Section 4.4.3, $\Phi(\mathbf{x},y)=\begin{bmatrix}\delta(y,1)\Phi(\mathbf{x})\\ \delta(y,-1)\Phi(\mathbf{x})\end{bmatrix}$. In the absence of prior knowledge about the similarity of classes, input features $\mathbf{x}$ of examples

with different class labels y are mapped to disjoint subsets of the feature vector. With this feature mapping the models for positive and negative examples do not interact and can be trained independently. Any suitable mapping $\Phi(\mathbf{x})$ can be applied. In Section 4.4.3, $p(t|\mathbf{x}, y, s=1)$ is modeled for all tasks jointly in single optimization problem with a multinomial logistic model. Empirically, we observe that a separate binary logistic regression model (as described above) for each task yields more accurate results with the drawback of an increased overall training time.

Optimization Problem 5.1 *For task t: over parameters $\mathbf{u}_t$, maximize*

$$\sum_{(\mathbf{x},y)\in L_t} \log(p(t|\mathbf{x}, y, s=1, \mathbf{u}_t)) + \sum_{(\mathbf{x},y)\in L\setminus L_t} \log(1 - p(t|\mathbf{x}, y, s=1, \mathbf{u}_t)) - \frac{\mathbf{u}_t^\mathsf{T}\mathbf{u}_t}{2\sigma_\mathbf{u}}.$$

The solution of Optimization Problem 5.1 is a MAP estimate of the logistic regression using a Gaussian prior on $\mathbf{u}_t$. The estimated vector $\mathbf{u}_t$ leads to the first part of the weighting factor $\hat{r}_t^1(\mathbf{x}, y) \propto p(t|\mathbf{x}, y, s=1, \mathbf{u}_t)$ according to Equation 5.8. $\hat{r}_t^1(\mathbf{x}, y)$ is normalized so that the weighted empirical distribution over the pool L sums to one, $\frac{1}{|L|}\sum_{(\mathbf{x},y)\in L} \hat{r}_t^1(\mathbf{x}, y) = 1$.

According to Equation 5.10 density ratio $r_t^2(\mathbf{x}) = \frac{p(\mathbf{x}|t,s=-1)}{p(\mathbf{x}|t,s=1)} \propto \frac{1}{p(s=1|\mathbf{x},t)} - 1$ can be inferred from $p(s=1|\mathbf{x}, t)$ which is the likelihood that a given $\mathbf{x}$ for task t originates from the training distribution, as opposed to from the test distribution. A model of $p(s=1|\mathbf{x}, t)$ can be obtained by discriminating a sample governed by $p(\mathbf{x}|t, s=1)$ against a sample governed by $p(\mathbf{x}|t, s=-1)$ using a probabilistic classifier. Unlabeled test data T_t is governed by $p(\mathbf{x}|t, s=-1)$. The labeled pool L over all training examples weighted by $r_t^1(\mathbf{x}, y)$ can serve as a sample governed by $p(\mathbf{x}|t, s=1)$; the labels y can be ignored for this step. Empirically, we find that using the weighted pool L instead of just L_t (as used in Section 3.3) achieves better results because the former sample is larger. We model $p(s=1|\mathbf{x}, \mathbf{v}_t)$ of Equation 5.10 with a regularized logistic regression on target variable s with parameters $\mathbf{v}_t$ (Optimization Problem 5.2). Labeled examples L are weighted by the estimated first factor $\hat{r}_t^1(\mathbf{x}, y)$ using the outcome of Optimization Problem 5.1.

Optimization Problem 5.2 *For task t: over parameters $\mathbf{v}_t$, maximize*

$$\sum_{(\mathbf{x},y)\in L} \hat{r}_t^1(\mathbf{x}, y) \log(p(s=1|\mathbf{x}, \mathbf{v}_t)) + \sum_{\mathbf{x}\in T_t} \log(p(s=-1|\mathbf{x}, \mathbf{v}_t)) - \frac{\mathbf{v}_t^\mathsf{T}\mathbf{v}_t}{2\sigma_\mathbf{v}}.$$

With the result of Optimization Problem 5.2 the estimate for the second factor is $\hat{r}_t^2(\mathbf{x}) \propto \frac{1}{p(s=1|\mathbf{x},\mathbf{v}_t)} - 1$, according to Equation 5.10. $\hat{r}_t^2(\mathbf{x})$ is normal-

ized so that the final weighted empirical distribution over the pool sums to one, $\frac{1}{|L|}\sum_{(\mathbf{x},y)\in L}\hat{r}_t^1(\mathbf{x},y)\hat{r}_t^2(\mathbf{x}) = 1$.

5.2.4 Weighted Empirical Loss and Target Model

The learning procedure first determines rescaling weights $\hat{r}_t(\mathbf{x},y) = \hat{r}_t^1(\mathbf{x},y)\hat{r}_t^2(\mathbf{x})$ by solving Optimization Problems 5.1 and 5.2. These weights can now be used to reweight the labeled pool over all tasks and train the target model for task t. Using the weights we can evaluate the expected loss over the weighted training data as displayed in Equation 5.11. It is the regularized empirical counterpart of Equation 5.6.

$$\mathbf{E}_{(\mathbf{x},y)\sim L}\left[\hat{r}_t^1(\mathbf{x},y)\hat{r}_t^2(\mathbf{x})\ell(f(\mathbf{x},t),y)\right] + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2} \tag{5.11}$$

Optimization Problem 5.3 minimizes Equation 5.11, the weighted regularized loss over the training data using a standard Gaussian log-prior with variance $\sigma_\mathbf{w}^2$ on the parameters $\mathbf{w}_t$. Each example is weighted by the two discriminatively estimated density fractions from Equations 5.8 and 5.10 using the solution of Optimization Problems 5.1 and 5.2.

Optimization Problem 5.3 *For task t: over parameters $\mathbf{w}_t$, minimize*

$$\frac{1}{|L|}\sum_{(\mathbf{x},y)\in L}\hat{r}_t^1(\mathbf{x},y)\hat{r}_t^2(\mathbf{x})\ell(f(\mathbf{x},\mathbf{w}_t),y) + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2}.$$

In order to train target models for all tasks, instances of Optimization Problems 5.1 to 5.3 are solved for each task.

5.3 Case Study: Targeted Advertising

In this section we study targeted advertising where the goal is to predict sociodemographic features (such as gender, age, or marital status) of web users, based on their surfing history. Many types of products are specifically targeted at clearly defined market segments, and marketing organizations seek to disseminate their message under minimal costs per delivery to a targeted individual. When sociodemographic attributes can be identified, delivering advertisements to users outside the target segment can be avoided. For some campaigns, clicks and resulting online purchases constitute an ultimate success criterion. However, for many campaigns—including campaigns for products that are not typically purchased on the web—the sole goal is to deliver the advertisement to customers in the target segment.

The targeted advertising problem instantiates the abstract problem of *multi-task learning under covariate shift* as follows. The feature vector $\mathbf{x}$ encodes the web surfing behavior of a user of web portal z (task). For a small number of users the sociodemographic target label y (e.g., gender of user) is collected through web surveys. For new portals the number of such labeled training instances is initially small. The sociodemographic labels for all users of all portals are to be predicted. The joint distribution $p(\mathbf{x}, y|z, \theta)$ can be different between portals since they attract specific populations of users. The training distribution differs from the test distribution because the response to the web surveys is not uniform with respect to the test distribution. Since the completion of surveys cannot be enforced, it is intrinsically impossible to obtain labeled samples that are governed by the test distribution. Therefore, a possible difference between the conditionals $p(y|\mathbf{x}, z, \theta)$ and $p(y|\mathbf{x}, z, \lambda)$ cannot be reflected in the model.

In the following sections we study the benefit of distribution matching and other reference methods on targeted advertising. After describing the data sets in Section 5.3.1, we review the experimental setup in Section 5.3.2, and finally, we discuss the results in Section 5.3.3.

5.3.1 Data Sets

We include four web portals (corresponding to tasks) in our study and manually assign topic labels, out of a fixed set of 373 topics, to all web pages on all portals. For each user the topics of the surfed pages are tracked and the topic counts are stored in cookies of the user's web browser. The average number of surfed topics per user over all portals is 17. The feature vector $\mathbf{x}$ of a specific surfer is the normalized 373 dimensional vector of topic counts.

A small proportion of users is asked to fill out a web questionnaire that collects sociodemographic user profiles. About 25% of the questionnaires get completely filled out (accepted) and in 75% of the cases the user rejects to fill out the questionnaire. The accepted questionnaires constitute the training data L. The completion of the questionnaire cannot be enforced and it is therefore not possible to obtain labeled data that is governed by the test distribution of all users that surf the target portal. In order to evaluate the model, we approximate the distribution of users who reject the questionnaire as follows. We take users who have answered the very first survey question (gender) but have then discontinued the survey as an approximation of the reject set. We add the correct proportion (25%) of users who have taken the survey, and thereby construct a sample that is governed by an approximation of the test distribution. Con-

Table 5.1: Portal statistics: number of accepted, partially rejected, and test examples (mix of all partial reject (=75%) and 25% accept); ratio of male users in training (accept) and test set.

portal	# accept	# partial reject	# test	% male training	% male test
family	8073	2035	2713	53.8%	46.6%
TV channel	8848	1192	1589	50.5%	50.1%
news 1	3051	149	199	79.4%	76.7%
news 2	2247	143	191	73.0%	76.0%

sequently, in our experiments we use the binary target label $y \in \{\text{male}, \text{female}\}$. Table 5.1 gives an overview of the data set.

5.3.2 Experimental Setup

We compare distribution matching on labeled and unlabeled data (Optimization Problems 5.1 to 5.3) and distribution matching only on labeled data by setting $\hat{r}_t^2(\mathbf{x}) = 1$ in Optimization Problem 5.3 (corresponds to Optimization Problem 4.3) to the following reference models. The first baseline is a one-size-fits-all model that directly trains a logistic regression on L (setting $\hat{r}_t^1(\mathbf{x}, y)\hat{r}_t^2(\mathbf{x}) = 1$ in Optimization Problem 5.3). The second baseline is a logistic regression trained only on L_t, the training examples of the target task. Training only on the reweighted target task data and correcting for marginal shift with respect to the unlabeled test data is the third baseline (Section 3.3).

The last reference method is the hierarchical Bayesian kernel of Evgeniou and Pontil (2004) described in Section 4.2.1. We evaluate the methods using all training examples from non-target tasks and different numbers of training examples from the target task. From all available accept examples of the target task we randomly select a certain number (0-1600) of training examples. From the remaining accept examples of the target task we randomly select an appropriate number and add them to all partial reject examples of the target task so that the evaluation set has the right proportions as described above. We repeat this process ten times and report the average accuracies of all methods.

We use a logistic loss as the target loss of distribution matching in Optimization Problem 5.3 and all reference methods. We compare kernelized variants of Optimization Problems 5.1 to 5.3 with RBF, polynomial, and linear kernels and find the linear kernel to achieve the best performance on our data set. All reported results are based on

models with linear kernels. For the optimization of the logistic regression models we use trust region Newton descent (Lin et al., 2008).

We tune parameter $\sigma_{\mathbf{u}}$ using ten-fold nested cross-validation on the training examples of the target task. With the examples of the training folds and with all data of non-target tasks we solve Optimization Problem 5.1 in the outer cross-validation loop. In the inner cross-validation loop we temporarily tune $\sigma_{\mathbf{w}}$ by solving Optimization Problem 5.3 (fixing $\hat{r}_t^2(\mathbf{x}) = 1$) and select a $\sigma_{\mathbf{u}}$ so that the accuracy on the tuning folds is maximized. Parameter $\sigma_{\mathbf{v}}$ is tuned by ten-fold likelihood cross-validation over labeled and unlabeled data as follows. The labels of the labeled data are ignored for this procedure. Test data T_t of the target task as well as the weighted pool L (weighted by $\hat{r}_t^1(\mathbf{x}, y)$) are split into ten folds. With the nine training folds of the test data and the nine training folds of the weighted pool L, Optimization Problem 5.2 is solved ($\hat{r}_t^1(\mathbf{x}, y)$ is fixed based on the model with the previously tuned $\sigma_{\mathbf{u}}$ parameter). Parameter $\sigma_{\mathbf{v}}$ is chosen to maximize the log-likelihood

$$\sum_{(\mathbf{x},y)\in L^{tune}} \hat{r}_t^1(\mathbf{x}, y) \log(p(s=1|\mathbf{x}, \mathbf{v}_t)) + \sum_{\mathbf{x}\in T_t^{tune}} \log(p(s=-1|\mathbf{x}, \mathbf{v}_t))$$

on the tuning folds of test data and weighted pool (denoted by L^{tune} and T_t^{tune}) over all ten cross-validation loops.

Shimodaira (2000) shows that weighting the training data with the true training-to-test density ratio is not necessarily the best way of learning under covariate shift because the expected loss of the target model can be large due to a large variance of the estimator of the target model (cf. Section 3.2). Intuitively, if the weights for many training examples are very small and only a few are large the effective sample size of the weighted training sample is small and training on a small sample decreases the expected performance of the model on unseen data. We follow Shimodaira (2000) and smooth the estimated weights by $\hat{r}_t^2(\mathbf{x})^\eta$ before including them into Optimization Problem 5.3. Without looking at the test data of the target task we tune η on the non-target tasks so that the accuracy of the distribution matching method is maximized. This procedure usually results in η values around 0.3. The last parameter $\sigma_{\mathbf{w}}$ is tuned by regular ten-fold cross-validation on the weighted pool L using the weights resulting from the previously tuned parameters $\sigma_{\mathbf{u}}$, $\sigma_{\mathbf{v}}$, and η. The following list summarizes the tuning procedure:

1. Tuning of $\sigma_{\mathbf{u}}$, outer loop: Accuracy cross-validation on L_t, in each loop solve Optimization Problem 5.1.

- Inner loop: Temporarily tuning of $\sigma_{\mathbf{w}}$ by accuracy cross-validation on rescaled $L_{\neg t}$ merged with the rescaled current training folds of L_t, in each loop solve Optimization Problem 5.3 with fixed $\hat{r}_t^2(\mathbf{x}) = 1$.

2. Tuning of $\sigma_{\mathbf{v}}$: Likelihood cross-validation on $T_t \cup L$, L is rescaled by $\hat{r}_t^1(\mathbf{x}, y)$, in each loop solve Optimization Problem 5.2.

3. Tuning of $\sigma_{\mathbf{w}}$: Accuracy cross-validation on L rescaled by $\hat{r}_t^1(\mathbf{x}, y)\hat{r}_t^2(\mathbf{x})$, in each loop solve Optimization Problem 5.3.

5.3.3 Results

Figure 5.1 displays the accuracies over different numbers of labeled data for the four different target portals. The error bars are the standard errors of the differences to the distribution matching method on labeled data (solid line).

For the "family" and "TV channel" portals the distribution matching method on labeled and unlabeled data outperforms all other methods in almost all cases. The distribution matching method on labeled data outperforms the baselines trained only on the data of the target task for all portals and all data set sizes and it is at least as good as the one-size-fits-all model in almost all cases. The hierarchical Bayes method yields low accuracies for smaller number of training examples but gets comparable to the distribution matching method on labeled data with increasing training set sizes of the target portal. The simple covariate shift model with training only on labeled and unlabeled data of the target task does not improve on the model that only trains on the labeled data of the target task. This indicates that the marginal shift between training and test distributions is rather small, or could indicate that the approximation of the reject distribution which we use in our experimentation is not sufficiently close. Either reason also explains why accounting for the marginal shift in the distribution matching method does not always improve over distribution matching using only labeled data.

5.4 Conclusion

We derived a multi-task learning method that is based on the insight that the expected loss with respect to the unbiased test distribution of the target task is equivalent to the expected loss over the biased training examples of all tasks weighted by a task specific rescaling weight. This led to an algorithm that discriminatively estimates these rescaling weights by training two simple conditional models. After weighting the pooled examples over all tasks the target model for a specific task can be trained.

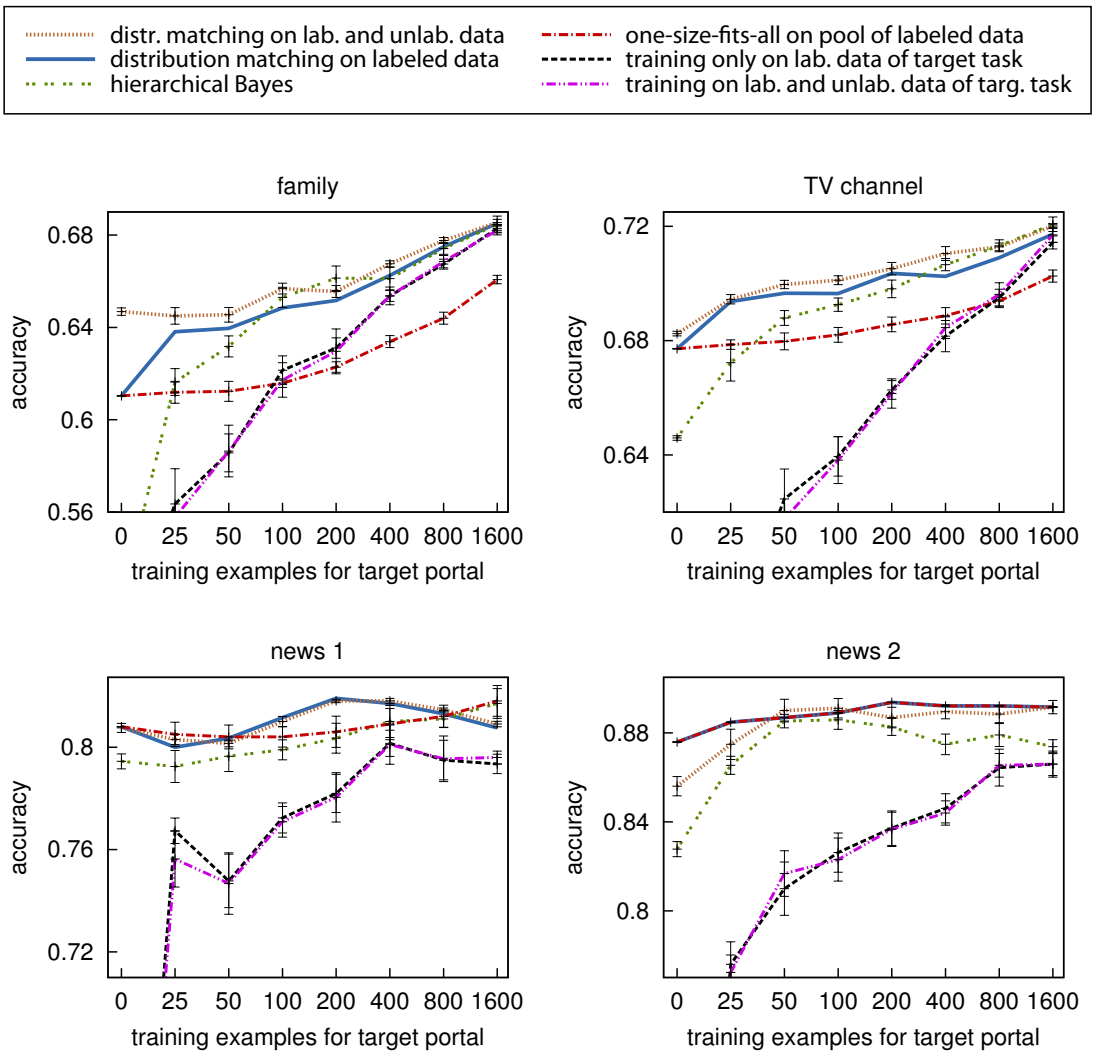

Figure 5.1: Accuracy over different number of training examples for target portal. Error bars indicate the standard error of the differences to distribution matching on labeled data.

In our empirical study on targeted advertising, we found that distribution matching using labeled data outperforms all reference methods in almost all cases; the differences are particularly large for small sample sizes. Distribution matching with labeled and unlabeled data outperforms the reference methods and distribution matching with only labeled data in two out of four portals. Even with no labeled data of the target task the performance of the distribution matching method is comparable to training on 1600 examples of the target task for all portals.

6 Conclusions

In this thesis we systematically studied learning under differing training and test distributions. We considered different assumptions on the relationship between training and test distributions. This led to three different problem settings: learning under covariate shift, multi-task learning, and multi-task learning under covariate shift.

For learning under covariate shift we derived a discriminative expression for rescaling weights that match the rescaled training distribution to the test distribution. Density ratios can be directly estimated with this discriminative model and frees us from the burden of estimating the potentially high dimensional distributions of the single densities separately, which is prevalent in the existing literature. We combined the model for the rescaling weights and the target model and obtained one single integrated optimization problem for learning under covariate shift. An analysis of the convexity of the integrated problem showed that it is only convex for an exponential loss function. A two-stage approximation to the integrated model makes the problem convex for all convex target loss functions and led to a procedure for extending almost any predictive model to learning under covariate shift. We provided a new discriminative view of the well known kernel mean matching procedure and pointed out the relationship to our discriminative model. These findings result in an out-of-sample extension of kernel mean matching that can be used for parameter tuning.

Empirically, the discriminative model outperforms the *iid* baseline and a kernel density estimated model on spam filtering, text classification, and landmine detection. The two-stage approximation yields comparable results to the integrated model. To our knowledge this is the first empirical study on learning under covariate shift that is based on real data with a natural shift in the distributions. All existing studies artificially introduce covariate shift into standard data sets.

In the problem of multi-task learning the difference between the task distributions is not restricted to the covariates but is reflected in the joint distribution over inputs and outputs. We derived a new model for multi-task learning that does not rely on any assumption on the relationship between the task distributions as all existing models for multi-task learning. It is based on rescaling weights that match the distribution

of the training data over all tasks to the distribution of the test data from a target task. Similar to the covariate shift model, the rescaling weights have a discriminative reformulation that can be estimated with logistic regression. A target model is trained on the rescaled or resampled training data over all tasks.

We derived a nested hierarchical Bayesian model for Gaussian processes that can be applied in settings with grouped tasks. We showed that two well known feature mappings for multi-task learning and taxonomy classification directly arise from hierarchical Bayesian models. These findings may lead to more sophisticated models for taxonomy classification. In a case study on HIV therapy screening, multi-task learning by distribution matching outperforms two *iid* baselines and three hierarchical Bayesian models in most of the cases.

We introduced the new problem setting of multi-task learning under covariate shift. This problem naturally arises in a multi-task setting when the collection of labeled training data is exposed to a selection process that leads to a biased training distribution. We devised a solution based on rescaling weights that match the mixture distribution over all tasks to the distribution of the test data for a specific target task. The rescaling weights can be factorized into a term that accounts for the covariate shift and a term that accounts for the divergence of distributions across tasks. These two terms can be reformulated in terms of discriminative models and estimated with two logistic regressions. Analogously to the other distribution matching methods, the target model is trained over reweighted training data from all tasks. A case study on targeted advertising shows that the distribution matching methods improve over the reference methods in almost all cases.

We are convinced that our findings are inspiring to future work in the field of learning under differing training and test distributions. We can think of several extensions and new applications of our work. An intrinsic effect of active learning is that the training distribution differs from the test distribution. In active learning the labeling process is guided towards informative examples. By this selection process covariate shift is introduced. Despite the broad interest of the machine learning community in active learning models, the covariate shift in active learning has been mostly neglected. The appealing property of active learning is that we can be certain that there is a pure covariate shift, not a joint input-output shift because the active learning procedure does not have access to the labels at the time of a labeling request. A related problem setting that has not been studied is active learning in a multi-task setting. The active learning process could query labels that help the model to better estimate the difference of distributions across tasks and thereby improve the transfer of knowledge

between tasks. In some cases the active labeling for different tasks may involve different task-specific costs and a cost-sensitive active learning framework could minimize the expected prediction performance under a limited labeling budget.

Our multi-task model based on distribution matching does not need any assumption on the relationship between the task distributions. This is in contrast to the existing multi-task models that make such assumption. For example, hierarchical Bayesian models assume a common prior distribution over all tasks. There may be settings when these assumptions are partially justified and hybrids between our distribution matching model and, e.g., a hierarchical Bayesian model can lead to better performance then the single models on their own.

A further related challenge is the construction of models for transfer learning that can exploit application specific prior knowledge on the type of transfer or type of distribution shift in the training data. For example, a face recognition model is trained from indoor pictures and applied to outdoor pictures. In this case one has prior knowledge that the main difference between training and test distributions are the lighting conditions and one could directly encode this knowledge in a transfer model.

Our models on learning under covariate shift rely on unlabeled data drawn from the test distribution. Semi-supervised learning is a related problem setting where training and test distributions are assumed to be identical and unlabeled data is available at the training time of the model. Typically, semi-supervised algorithms make some cluster assumption on the shape of the data distributions. One could combine covariate shift and semi-supervised models and construct new models that account for covariate shift in the training data and make a cluster assumption on the data distributions at the same time.

Appendix A

Newton Updates for Integrated Covariate Shift Model

In this Appendix, we derive Newton gradient descent updates for Optimization Problem 1 and thereby prove Theorem 3.1. We abbreviate

$$\ell_{\mathbf{v},i}=\ell_{\mathbf{v}}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i);\quad \ell'_{\mathbf{v},i}s_ix_{ij}=\frac{\partial\ell_{\mathbf{v}}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{\partial v_j};\quad \ell''_{\mathbf{v},i}x_{ij}x_{ik}=\frac{\partial^2\ell_{\mathbf{v}}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{\partial v_jv_k}; \tag{A.1}$$

$$\ell_{\mathbf{w},i}=\ell_{\mathbf{w}}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i);\quad \ell'_{\mathbf{w},i}y_ix_{ij}=\frac{\partial\ell_{\mathbf{w}}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)}{\partial w_j};\quad \ell''_{\mathbf{w},i}x_{ij}x_{ik}=\frac{\partial^2\ell_{\mathbf{w}}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)}{\partial w_jw_k}; \tag{A.2}$$

$$\omega_i=\frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i) \tag{A.3}$$

and denote the objective function of Optimization Problem 1 by

$$F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T) \;=\; \sum_{i=1}^{m}\omega_i\ell_{\mathbf{w},i}+\sum_{i=1}^{m+n}\ell_{\mathbf{v},i}+\frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^\mathsf{T}\mathbf{w}+\frac{1}{2\sigma_{\mathbf{v}}^2}\mathbf{v}^\mathsf{T}\mathbf{v}. \tag{A.4}$$

We compute the gradient with respect to $\mathbf{v}$ and $\mathbf{w}$.

$$\frac{\partial F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T)}{\partial v_j} \;=\; -\sum_{i=1}^{m}\omega_i\ell_{\mathbf{w},i}x_{ij}+\sum_{i=1}^{m+n}\ell'_{\mathbf{v},i}s_ix_{ij}+\frac{1}{\sigma_{\mathbf{v}}^2}v_j \tag{A.5}$$

$$\frac{\partial F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T)}{\partial w_j} \;=\; \sum_{i=1}^{m}\omega_i\ell'_{\mathbf{w},i}y_ix_{ij}+\frac{1}{\sigma_{\mathbf{w}}^2}w_j \tag{A.6}$$

The Hessian is the matrix of second derivatives.

$$\frac{\partial^2 F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T)}{\partial v_j \partial v_k} = \sum_{i=1}^{m} \omega_i \ell_{\mathbf{w},i} x_{ij} x_{ik} + \sum_{i=1}^{m+n} \ell''_{\mathbf{v},i} x_{ij} x_{ik} + \frac{1}{\sigma^2_{\mathbf{v}}} \delta_{jk} \tag{A.7}$$

$$\frac{\partial^2 F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T)}{\partial v_j \partial w_k} = -\sum_{i=1}^{m} \omega_i \ell'_{\mathbf{w},i} y_i x_{ij} x_{ik} \tag{A.8}$$

$$\frac{\partial^2 F(\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{X}_L,\mathbf{X}_T)}{\partial w_j \partial w_k} = \sum_{i=1}^{m} \omega_i \ell''_{\mathbf{w},i} x_{ij} x_{ik} + \frac{1}{\sigma^2_{\mathbf{w}}} \delta_{jk} \tag{A.9}$$

We can rewrite gradient as $\mathbf{X}\mathbf{g} + S\left[\mathbf{v},\mathbf{w}\right]^\mathsf{T}$ and Hessian as $\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T} + S$ using the following definitions, $\mathbf{g} = \left[\mathbf{g}^{(1)},\mathbf{g}^{(2)},\mathbf{g}^{(3)}\right]^\mathsf{T}$, $S = \begin{bmatrix} S^{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & S^{\mathbf{w}} \end{bmatrix}$ with

$$g_i^{(1)} = -\omega_i \ell_{\mathbf{w},i} + \ell'_{\mathbf{v},i} \quad \text{for} \quad i = 1,\ldots,m, \tag{A.10}$$

$$g_i^{(2)} = -\ell'_{\mathbf{v},m+i} \quad \text{for} \quad i = 1,\ldots,n, \tag{A.11}$$

$$g_i^{(3)} = \omega_i \ell'_{\mathbf{w},i} y_i \quad \text{for} \quad i = 1,\ldots,m, \tag{A.12}$$

$$S^{\mathbf{v}}_{i,i} = \sigma_{\mathbf{v}}^{-2} \quad \text{for} \quad i = 1,\ldots,\dim(\mathbf{X}_T), \tag{A.13}$$

$$S^{\mathbf{w}}_{i,i} = \sigma_{\mathbf{w}}^{-2} \quad \text{for} \quad i = 1,\ldots,\dim(\mathbf{X}_L), \tag{A.14}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i \ell_{\mathbf{w},i} + \ell''_{\mathbf{v},i}\right) & \mathbf{0} & -\operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i \ell'_{\mathbf{w},i} y_i\right) \\ \mathbf{0} & \operatorname*{diag}_{i=1,\ldots,n}\left(\ell''_{\mathbf{v},m+i}\right) & \mathbf{0} \\ -\operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i \ell'_{\mathbf{w},i} y_i\right) & \mathbf{0} & \operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i \ell''_{\mathbf{w},i}\right) \end{bmatrix}. \tag{A.15}$$

The update step for the Newton gradient descent minimization of Optimization Problem 1 is $[\mathbf{v}',\mathbf{w}']^\mathsf{T} \leftarrow [\mathbf{v},\mathbf{w}]^\mathsf{T} + [\Delta_{\mathbf{v}},\Delta_{\mathbf{w}}]^\mathsf{T}$ with

$$(\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T} + S) \begin{bmatrix} \Delta_{\mathbf{v}} \\ \Delta_{\mathbf{w}} \end{bmatrix} = -\mathbf{X}\mathbf{g} - S \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}.$$

■

Appendix B

Optimality Conditions of Kernel Mean Matching

We analyze the Karush-Kuhn-Tucker conditions of kernel mean matching (Optimization Problem 3.2, Huang et al. (2007)):

$$\begin{aligned} \min_{\alpha_L} \quad & \frac{1}{2}\alpha_L^\mathsf{T} K_{(LL)}\alpha_L - \frac{m}{n}\alpha_L^\mathsf{T} K_{(LT)}\mathbf{1} \\ & \text{subject to } \ \alpha_i \in \left[0, \sigma_{\mathbf{v}}^2\right] \quad \text{and} \quad m(1-\epsilon) \leq \alpha_L^\mathsf{T}\mathbf{1} \leq m(1+\epsilon). \end{aligned} \tag{B.1}$$

Kernel mean matching uses the α_i, the results of the optimization problem, as rescaling factors for the target classifier (Optimization Problem 3). In order to find out how we can interpret α_i we analyze the optimality conditions of kernel mean matching; we divide the objective by m and construct the Lagrangian in Equation B.2.

$$\begin{aligned} \mathcal{L} \ = \ & \frac{1}{2m}\alpha_L^\mathsf{T} K_{(LL)}\alpha_L - \frac{1}{n}\alpha_L^\mathsf{T} K_{(LT)}\mathbf{1} - \gamma_1(\alpha_L^\mathsf{T}\mathbf{1} - m(1-\epsilon)) \\ & + \gamma_2(\alpha_L^\mathsf{T}\mathbf{1} - m(1+\epsilon)) - \sum_{i=1}^{m}\phi_i\alpha_i + \sum_{i=1}^{m}\mu_i(\alpha_i - \sigma_{\mathbf{v}}^2) \end{aligned} \tag{B.2}$$

The Karush-Kuhn-Tucker conditions are:

$$\frac{\partial\mathcal{L}}{\partial\alpha_i} = \frac{1}{m}\sum_{j=1}^{m}\alpha_j k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{n}\sum_{j=m+1}^{m+n} k(\mathbf{x}_i, \mathbf{x}_j) - \gamma_1 + \gamma_2 - \phi_i + \mu_i = 0 \tag{B.3}$$

$$\gamma_1 \geq 0 \tag{B.4}$$

$$\gamma_1(\alpha_L^\mathsf{T}\mathbf{1} - m(1-\epsilon)) = 0 \tag{B.5}$$

$$\gamma_2 \geq 0 \tag{B.6}$$

$$\gamma_2(\alpha_L^\mathsf{T}\mathbf{1} - m(1+\epsilon)) = 0 \tag{B.7}$$

$$\phi_i \geq 0 \tag{B.8}$$

$$\phi_i\alpha_i = 0 \tag{B.9}$$

$$\mu_i \geq 0 \tag{B.10}$$

$$\mu_i(\alpha_i - \sigma_{\mathbf{v}}^2) = 0 \tag{B.11}$$

The decision function of kernel mean matching is $g(\mathbf{x};\alpha,b) = \frac{1}{m}\sum_{j=1}^{m}\alpha_j k(\mathbf{x},\mathbf{x}_j) - \frac{1}{n}\sum_{j=m+1}^{m+n} k(\mathbf{x},\mathbf{x}_j) + b$. The first term is the SVM part and the second the Rocchio part of the decision function, b is the offset parameter. Similar to the regular SVM we can simplify the Karush-Kuhn-Tucker conditions by considering three cases, we set $-\gamma_1 + \gamma_2 = b$:

$$\begin{array}{lllll} \text{case 1:} & g(\mathbf{x}_i;\alpha,b) \geq 0, & \phi_i \geq 0,\ \mu_i = 0 & \Rightarrow & \alpha_i = 0 \\ \text{case 2:} & g(\mathbf{x}_i;\alpha,b) \leq 0, & \phi_i = 0,\ \mu_i \geq 0 & \Rightarrow & \alpha_i = \sigma_{\mathbf{v}}^2 \\ \text{case 3:} & g(\mathbf{x}_i;\alpha,b) = 0, & \phi_i = 0,\ \mu_i = 0 & \Rightarrow & 0 < \alpha_i < \sigma_{\mathbf{v}}^2. \end{array} \tag{B.12}$$

Appendix C

EM Updates for Nested Hierarchical Bayes with Gaussian Processes

In this appendix we prove Theorem 4.1 and derive EM updates for maximizing the posterior of Equation 4.22. We use the function values $\mathbf{f}_{kz}$ as hidden variables and derive the so called Q-function (Dempster et al., 1977). In case of a MAP estimation the Q-function is the expectation over the log-joint distribution of model parameters $\phi = \{\mathbf{g}_k, \mu, \mathbf{K}\}$, data y, and hidden variables $\mathbf{f}_{kz}$ given the parameters of the last iteration $\phi^{(j-1)}$ (Equation C.1). The prior is independent of $\mathbf{f}_{kz}$ and can be drawn out of the expectation (Equation C.2)

$$
\begin{aligned}
Q(\phi|\phi^{(j-1)}) &= \mathbf{E}[p(\mathbf{g}_k, \mu, \mathbf{K}, \mathbf{y}, \mathbf{f}_{kz}|\mathbf{X}, \sigma^2, \pi, \tau, \kappa)|\phi^{(j-1)}] && \text{(C.1)} \\
&= \mathbf{E}[p(\mathbf{y}, \mathbf{f}_{kz}|\mathbf{g}_k, \mathbf{K}, \mathbf{X}, \sigma^2)|\phi^{(j-1)}] + p(\mathbf{g}_k, \mu, \mathbf{K}|\pi, \tau, \kappa) && \text{(C.2)}
\end{aligned}
$$

The first term of Equation C.2 is expanded in Equation C.3 and further expanded in Equation C.4. For a simplified presentation we drop the conditioning on $\phi^{(j-1)}$ in the notation of the expectations.

$$
\begin{aligned}
&\mathbf{E}[p(\mathbf{y}, \mathbf{f}_{kz}|\mathbf{g}_k, \mathbf{K}, \mathbf{X}, \sigma^2)] \\
&= \mathbf{E}\left[\sum_k \sum_{\substack{z\in \\ \text{group } k}} \left(\log N(\mathbf{f}_{kz}|\mathbf{g}_k, \mathbf{K}) + \sum_{i\in L_{kz}} \log N(y_i|f_{kzi}, \sigma^2)\right)\right] && \text{(C.3)} \\
&= -\frac{1}{2}\sum_k \sum_{\substack{z\in \\ \text{group } k}} \left(\mathbf{E}\left[(\mathbf{f}_{kz} - \mathbf{g}_k)^\mathsf{T} K^{-1}(\mathbf{f}_{kz} - \mathbf{g}_k)\right] + \frac{1}{2\sigma^2}\mathbf{E}\left[||\mathbf{y}_{kz} - \mathbf{f}_{kz}||^2\right]\right) && \text{(C.4)} \\
&\quad -|L|\log\sigma - \frac{1}{2}|L|\log(2\pi) - \frac{1}{2}\sum_k \sum_{\substack{z\in \\ \text{group } k}} (|L|\log(2\pi) + \log|\mathbf{K}|)
\end{aligned}
$$

The binomial theorem applied to the first expectation of Equation C.4 yields Equation C.5. Equation C.6 uses the well known relationship between quadratic forms and trace, and the decomposition property of a covariance matrix, $\mathbf{E}[\mathbf{f}_{kz}^{\mathsf{T}} K^{-1} \mathbf{f}_{kz}] = \mathbf{E}[\mathrm{tr}(\mathbf{f}_{kz}\mathbf{f}_{kz}^{\mathsf{T}} K^{-1})] = \mathrm{tr}(\mathbf{E}[\mathbf{f}_{kz}\mathbf{f}_{kz}^{\mathsf{T}}]K^{-1}) = \mathrm{tr}((\mathrm{Cov}[\mathbf{f}_{kz}] + \mathbf{E}[\mathbf{f}_{kz}]\mathbf{E}[\mathbf{f}_{kz}]^{\mathsf{T}})K^{-1}) = \mathrm{tr}(\mathrm{Cov}[\mathbf{f}_{kz}]K^{-1}) + \mathbf{E}[\mathbf{f}_{kz}]^{\mathsf{T}} K^{-1} \mathbf{E}[\mathbf{f}_{kz}]$. The last three terms of Equation C.6 are proportional to a log-Gaussian density function (Equation C.7).

$$
\begin{aligned}
&\mathbf{E}\left[(\mathbf{f}_{kz} - \mathbf{g}_k)^{\mathsf{T}} K^{-1} (\mathbf{f}_{kz} - \mathbf{g}_k)\right] \\
&= \mathbf{E}\left[\mathbf{f}_{kz}^{\mathsf{T}} K^{-1} \mathbf{f}_{kz} - 2\mathbf{f}_{kz}^{\mathsf{T}} K^{-1} \mathbf{g}_k\right] + \mathbf{g}_k^{\mathsf{T}} K^{-1} \mathbf{g}_k && \text{(C.5)} \\
&= \mathrm{tr}(\mathrm{Cov}[\mathbf{f}_{kz}]K^{-1}) + \mathbf{E}\left[\mathbf{f}_{kz}\right]^{\mathsf{T}} K^{-1} \mathbf{E}\left[\mathbf{f}_{kz}\right] - 2\mathbf{E}\left[\mathbf{f}_{kz}\right]^{\mathsf{T}} K^{-1} \mathbf{g}_k + \mathbf{g}_k^{\mathsf{T}} K^{-1} \mathbf{g}_k && \text{(C.6)} \\
&\propto \mathrm{tr}(\mathrm{Cov}[\mathbf{f}_{kz}]K^{-1}) - \log N(\mathbf{E}\left[\mathbf{f}_{kz}\right] | \mathbf{g}_k, K) && \text{(C.7)}
\end{aligned}
$$

For the evaluation of the Q-function of Equations C.1, C.4, and C.7 the expectation and covariance matrix of $\mathbf{f}_{kz}$ are needed. According to Schwaighofer et al. (2005) they are equivalent to the standard predictive mean and covariance for Gaussian process models as displayed in Equations C.8 and C.9. The index kz at $\mathbf{K}_{*,kz}$ and $\mathbf{K}_{kz,kz}$ restricts the covariance matrices to all elements corresponding to the instances for group k and task z. Analogously, $\mathbf{y}_{kz}$ and $\mathbf{g}_{kz}$ are training labels for group k and task z and the corresponding subvector of the group mean.

$$
\begin{aligned}
\mathbf{E}\left[\mathbf{f}_{kz}\right] &= \mathbf{K}_{*,kz} \left(\mathbf{K}_{kz,kz} + \sigma^2 \mathbf{I}\right)^{-1} (\mathbf{y}_{kz} - \mathbf{g}_{kz}) + \mathbf{g}_k && \text{(C.8)} \\
\mathrm{Cov}[\mathbf{f}_{kz}] &= \mathbf{K} - \mathbf{K}_{*,kz} \left(\mathbf{K}_{kz,kz} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}_{*,kz}^{\mathsf{T}} && \text{(C.9)}
\end{aligned}
$$

The computation of Equations C.8 and C.9 constitute the E-step. In the M-step the Q-function is maximized with respect to the prior parameters:

$$
(\mathbf{g}_k^*, \mu^*, \mathbf{K}^*) = \underset{\mathbf{g}_k, \mu, \mathbf{K}}{\mathrm{argmax}}\, Q(\phi | \phi^{(j-1)}) \qquad \text{(C.10)}
$$

In order to find the maximizing parameters we compute the partial derivatives of $Q(\phi|\phi^{(j-1)})$ and set them to zero. The partial derivatives with respect to $\mathbf{g}_k$ are displayed in Equations C.11 and C.12.

$$
\begin{aligned}
\frac{\partial Q(\phi|\phi^{(j-1)})}{\partial \mathbf{g}_k} &= \frac{\partial \left(\sum_{z \in \text{group } k} \log N(\mathbf{E}\left[\mathbf{f}_{kz}\right] | \mathbf{g}_k, \mathbf{K}) + \log N(\mathbf{g}_k | \mu, \mathbf{K}) \right)}{\partial \mathbf{g}_k} && \text{(C.11)} \\
&= \sum_{z \in \text{group } k} \mathbf{K}^{-1} (\mathbf{E}\left[\mathbf{f}_{kz}\right] - \mathbf{g}_k) - \mathbf{K}^{-1} (\mathbf{g}_k - \mu) && \text{(C.12)}
\end{aligned}
$$

Setting Equation C.12 to zero and solving for $\mathbf{g}_k$ gives

$$\mathbf{g}_k = \frac{\sum\limits_{z \in \text{group } k} \mathbf{E}\left[\mathbf{f}_{kz}\right] + \mu}{|z \in \text{group } k| + 1}. \tag{C.13}$$

The partial derivatives with respect to μ are shown in Equations C.14 and C.15.

$$\begin{aligned} \frac{\partial Q(\phi|\phi^{(j-1)})}{\partial \mu} &= \frac{\partial \left(\log N(\mu|0, \frac{1}{\pi}\mathbf{K}) + \sum_k \log N(\mathbf{g}_k|\mu, \mathbf{K})\right)}{\partial \mu} && \text{(C.14)} \\ &= -\pi\mathbf{K}^{-1}\mu + \sum_k \mathbf{K}^{-1}(\mathbf{g}_k - \mu) && \text{(C.15)} \end{aligned}$$

Setting Equation C.15 to zero yields

$$\mu = \frac{\sum_k \mathbf{g}_k}{|\text{groups}| + \pi}. \tag{C.16}$$

The partial derivatives with respect to $\mathbf{K}^{-1}$ are shown in Equations C.17 and C.18.

$$\begin{aligned} &\frac{\partial Q(\phi|\phi^{(j-1)})}{\partial \mathbf{K}^{-1}} = \\ &= \frac{\partial \left(\sum\limits_k \sum\limits_{z \in \text{group } k} \left(-\operatorname{tr}(\operatorname{Cov}[\mathbf{f}_{kz}]K^{-1}) + \log N(\mathbf{E}\left[\mathbf{f}_{kz}\right]|\mathbf{g}_k, \mathbf{K})\right)\right)}{\partial \mathbf{K}^{-1}} && \text{(C.17)} \\ &\quad + \frac{\partial \left(\log IW(\mathbf{K}|\tau, \kappa^{-1}) + \log N(\mu|0, \frac{1}{\pi}\mathbf{K}) + \sum\limits_k \log N(\mathbf{g}_k|\mu, \mathbf{K})\right)}{\partial \mathbf{K}^{-1}} \\ &= \frac{1}{2}\sum_k \sum_{z \in \text{group } k} \left(-\operatorname{Cov}[\mathbf{f}_{kz}] + K - (\mathbf{E}[\mathbf{f}_{kz}] - \mathbf{g}_k)(\mathbf{E}[\mathbf{f}_{kz}] - \mathbf{g}_k)^\mathsf{T}\right) && \text{(C.18)} \\ &\quad + \frac{\tau - 1}{2}\mathbf{K} - \frac{\tau}{2}\kappa + \frac{1}{2}\mathbf{K} + \frac{1}{2}\sum_k \left(\mathbf{K} - (\mathbf{g}_k - \mu)(\mathbf{g}_k - \mu)^\mathsf{T}\right) \end{aligned}$$

Setting Equation C.18 to zero and solving for $\mathbf{K}$ results in Equation C.19.

$$\begin{aligned} \mathbf{K} &= \left(\sum_k \sum_{z \in \text{group } k} \left(\operatorname{Cov}[\mathbf{f}_{kz}] + (\mathbf{E}[\mathbf{f}_{kz}] - \mathbf{g}_k)(\mathbf{E}[\mathbf{f}_{kz}] - \mathbf{g}_k)^\mathsf{T}\right)\right. && \text{(C.19)} \\ &\quad \left. + \tau\kappa + \sum_k (\mathbf{g}_k - \mu)(\mathbf{g}_k - \mu)^\mathsf{T}\right) \frac{1}{\tau + |\text{tasks}| + |\text{groups}|} \end{aligned}$$

Now we can summarize the steps of the EM algorithm.

- In the E-step Equations C.8 and C.9 are computed based on the parameters of the last iteration $\phi^{(j-1)}$.

- In the M-step the Q-function is maximized by computing new parameters with Equations C.13, C.16, and C.19.

■

Bibliography

A. Altmann, N. Beerenwinkel, T. Sing, I. Savenkov, M. Doumer, R. Kaiser, S. Rhee, W. Fessel, W. Shafer, and T. Lengauer. Improved prediction of response to antiretroviral combination therapy using the genetic barrier to drug resistance. *Antiviral Therapy*, 12:169–178, 2007.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 2007.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *The Journal of Machine Learning Research*, 4:83–99, 2003.

J. Bi, T. Xiong, S. Yu, M. Dundar, and B. Rao. An improved multi-task learning approach with applications in medical diagnosis. In *Proceedings of the European Conference on Machine Learning*, 2008.

S. Bickel, editor. *Proceedings of the ECML-PKDD Discovery Challenge Workshop*. 2006.

S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems*, 2007.

S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the International Conference on Machine Learning*, 2007.

S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for HIV therapy screening. In *Proceedings of the International Conference on Machine Learning*, 2008a.

S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning under covariate shift with a single optimization problem. In J. Quinonero Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors, *Dataset Shift in Machine Learning*. MIT Press, Cambridge, 2008b.

S. Bickel, C. Sawade, and T. Scheffer. Transfer learning by distribution matching for targeted advertising. In *Advances in Neural Information Processing Systems*, 2009, to appear.

E. Bonilla, F. Agakov, and C. Williams. Kernel multi-task learning using task-specific features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007.

E. Bonilla, K. Chai, and C. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2008.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2008.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39, 1977.

M. Dudik, R. Schapire, and S. Phillips. Correcting sample selection bias in maximum entropy density estimation. In *Advances in Neural Information Processing Systems*, 2005.

M. Dudik, D. Blei, and R. Schapire. Hierarchical maximum entropy density estimation. In *Proceedings of the International Conference on Machine Learning*, 2007.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the International Joint Conference on Artificial Intellligence*, 2001.

T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2004.

A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2004.

I. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press, 1965.

J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47:153–161, 1979.

M. Hein. Binary classification under sample selection bias. In J. Quinonero Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors, *Dataset Shift in Machine Learning*. MIT Press, Cambridge, 2008.

R. Herbrich. *Learning kernel classifiers*. MIT Press, 2002.

T. Heskes. Empirical Bayes for learning to learn. In *Proceedings of the International Conference on Machine Learning*, 2000.

J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, 2007.

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6:429–449, 2002.

T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the International Conference on Machine Learning*, 1997.

V. Johnson, F. Brun-Vezinet, B. Clotet, H. Günthrad, D. Kuritzkes, D. Pillay, J. Schapiro, A. Telenti, and D. Richman. Update of the drug resistance mutations in HIV-1: 2007. *Top HIV Med.*, 15:119–125, 2007.

S. Kaski and J. Peltonen. Learning from relevant tasks only. In *Proceedings of the European Conference on Machine Learning*, 2007.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

B. Larder, D. Wang, A. Revell, J. Montaner, R. Harrigan, F. De Wolf, J. Lange, S. Wegner, L. Ruiz, MJ. Pérez-Elías, S. Emery, J. Gatell, A. D'Arminio Monforte, C. Torti, M. Zazzi, and C. Lane. The development of artificial neural networks to predict virological response to combination HIV therapy. *Antiviral Therapy*, 12: 15–24, 2007.

R. Lathrop and M. Pazzani. Combinatorial optimization in rapidly mutating drug-resistant viruses. *Journal of Combinatorial Optimization*, 3:301–320, 1999.

X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. *Proceedings of the International Conference on Machine Learning*, 2005.

C. Lin, R. Weng, and S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008.

Q. Liu, X. Liao, and L. Carin. Semi-supervised multitask learning. In *Advances in Neural Information Processing Systems*, 2008.

J. Lunceford and M. Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2004.

C. Manski and S. Lerman. The estimation of choice probabilities from choice based samples. *Econometrica*, 45(8):1977–1988, 1977.

G. Obozinski, B. Taskar, and M Jordan. Multi-task feature selection. Technical report, UC Berkeley, 2007.

R. Prentice and R. Pyke. Logistic disease incidence models and case-control studies. *Biometrika*, 66(3):403–411, 1979.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

M. Rosen-Zvi, A. Altmann, M. Prosperi, E. Aharoni, H. Neuvirth, A. Sönnerborg, E. Schülter, D. Struck, Y. Peres, F. Incardona, R. Kaiser, M. Zazzi, and T. Lengauer. Selecting anti-HIV therapies based on a variety of genomic and clinical factors. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 2008.

P. Rosenbaum and D. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

D. Roy and L. Kaelbling. Efficient Bayesian task-level transfer learning. In *Proceedings of the Joint Conference on Artificial Intelligence*, 2007.

B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. In *Advances in Neural Information Processing Systems*, 2005.

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

B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, 1986.

A. Smith and C. Elkan. A Bayesian network framework for reject inference. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2004.

A. Smith and C. Elkan. Making generative classifiers robust to selection bias. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2007.

N. Sokolovska, O. Cappe, and F. Yvon. The asymptotics of semi-supervised learning in discriminative probabilistic models. In *Proceedings of the International Conference on Machine Learning*, 2008.

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

M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, 2008a.

M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4), 2008b.

Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

J. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. In *Proceedings of the SIAM International Conference on Data Mining*, 2008.

UNAIDS/WHO. AIDS Epidemic Update. 2007.

P. Wu and T.G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. *Proceedings of the International Conference on Machine Learning*, 2004.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. *Proceedings of the International Conference on Machine Learning*, 2005.

S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *Proceedings of the International Conference on Machine Learning*, 2007.

B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the International Conference on Machine Learning*, 2004.

J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. *Advances in Neural Information Processing Systems*, 17, 2005.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, 2002.