Regularization-based Multitask Learning

With Applications in Computational Biology

Regularization-based Multitask Learning

With Applications in Computational Biology

vorgelegt von Dipl.-Inform.

CHRISTIAN WIDMER

geb. in Böblingen

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte

DISSERTATION

PROMOTIONSAUSSCHUSS:

Vorsitzender: Prof. Dr. Olaf Hellwich
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin

Gutachter: Prof. Dr. Klaus-Robert Müller
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin

Prof. Dr. Gunnar Rätsch
Computational Biology Center
Memorial Sloan-Kettering Cancer Center, New York, USA

Prof. Dr. Klaus Obermayer
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin

Tag der mündlichen Aussprache: 29.09.2014

Berlin, 2014

To my family.

Acknowledgements

This work was carried out during the years 2009-2013 at the Friedrich Miescher Laboratory of the Max Planck Society in Tübingen, Germany, the Berlin Institute of Technology (TU Berlin) and the Department of Computational Biology at Memorial Sloan-Kettering Cancer Center in New York City.

These past years were a period of personal growth, going beyond solely scientific, professional or intellectual growth. It is for the constant exchange of ideas, the love of learning, an environment encouraging creativity and insight, but also the experiences of travel, living in vibrant cities abroad making friends along the way and taking far reaching decisions about my own life that I feel have helped me grow as a human being.

I am deeply grateful for having had the opportunity to work with brilliant and creative minds over the last years. First and foremost, I'd like to express my gratitude to my advisors Gunnar Rätsch and Klaus-Robert Müller, who made it possible for me to carry out this work and who contributed to this thesis with their time, ideas and energy. I want to express my warmest gratitude to Yasemin Altun for mentoring me during the first years of my thesis.

If it wasn't for Cheng Soon Ong, who attracted me to the world of research with his love for teaching, wit and passion for machine learning research, I may not have pursued this path. Besides him, it was the early mentorship of Alexander Jung, Petra Philips, Gabrielle Schweikert and Bernhard Schölkopf that helped me decide on a topic, which eventually became my Ph.D. thesis. A very special thanks I'd like to address to Marius Kloft, who supported me through the later parts of this thesis in many ways with his expertise, advice and friendship.

The work done in the last years would have not been possible without the fruitful collaboration with my colleagues Jose Leiva, Sören Sonnenburg, Nora Toussaint, Jonas Behr, Georg Zeller, Nico Görnitz, Vipin Sreedharan, Philipp Drewe, Stephanie Heinrich, Silke Hauf, Raphael Pelossof and Xinghua Lou. I took particular pleasure in working with Sergey Lisitsyn as part of Google Summer of Code and in co-supervising the projects of Shefali Umrania and Behrouz Tajoddin.

I am indebted to my advisers, Marius Kloft, David Kuo, Nico Görnitz, Stefan Laible, Johannes Furch, Timo Sachsenberg and Jose Leiva for valuable comments and corrections to the manuscript.

Much of what I've learned the last years came from white-board discussions, reading groups and chats over coffee. For this, besides many of the people mentioned above, I'd like to thank David Kuo, Christoph Lippert, Gideon Dresdner, Urun Dogan, Fabio DeBona, Oliver Stegle, Regina Bohnert, and everyone who participated in our reading group sessions. I'm grateful to Suvrit Sra and Yevgeny Seldin, who shared their knowledge in our reading group on convex optimization. Finally, I'm deeply grateful to every member of the raetschlab, as it was them who made this journey a memorable experience.

I acknowledge financial support of the Max Planck Society, the Berlin Institute of Technology (TU Berlin), the German National Science Foundation (DFG) under RA 1894/1-1, and the Memorial Sloan-Kettering Cancer Center.

Abstract

In this work, we consider a problem that biologists are very good at: deciphering biological processes by integrating knowledge from experiments in different biological entities, such as organisms, tissues, tumor types or proteins, while respecting their differences and commonalities. We look at this problem from a supervised learning point of view, aiming to solve the same inference task in different biological entities. In this thesis, we investigate two branches of transfer learning: *domain adaptation*, where information is transferred from source tasks with abundant information to target tasks with little information, and *multitask learning*, where information is mutually shared between several tasks.

In the case of domain adaptation, we show simple extensions of prevalent regularized risk minimization frameworks to handle information transfer and derive efficient solvers for classification and structured output learning. We present an algorithm tailored for the setting of hierarchical task relationships. This setting is particularly relevant to computational biology, where different tasks often correspond to different organisms, whose relationship is defined by a phylogeny. We perform experimental analyses on synthetic data sets, problems from sequence biology and prokaryotic gene finding to explore the properties of our algorithms and demonstrate their performance.

Multitask learning, a machine learning technique that has recently received considerable attention, considers the problem of simultaneously learning models for several tasks that are related to each other. Using modern mathematical optimization techniques, we develop a general framework for multitask learning that encompasses a large number of existing multitask learning formulations and carefully explore useful special cases, including several novel formulations. A main feature of our general framework is the ability to learn or refine task similarities using non-sparse multiple kernel learning (MKL). We derive an efficient dual-coordinate descent solver for the special case of the hinge-loss, which brings the performance of state-of-the-art linear SVM solvers for binary classification to multitask learning. We explore the application of our framework to important problems ranging from computational immunotherapy, bioimaging and sequence biology. We further provide run-time experiments on a large range of data sets.

As a whole, this thesis encompasses the design of transfer learning algorithms by means of carefully engineered regularization terms, the effort of creating and making available software that implements these ideas efficiently and the application to a plethora of practical problems, where transfer learning may be regarded as a principled way of obtaining more cost-effective predictors.

Zusammenfassung

Diese Arbeit befasst sich mit einem in der biologischen Forschung alltäglichen Problem: Dem Entschlüsseln von biologischen Prozessen mit Hilfe von Experimenten aus verschiedenen biologischen Einheiten. So kann z.B. das Wissen aus verschiedenen Organismen, Gewebetypen, oder Tumorarten jeweils kombiniert werden, wobei man für deren Ähnlichkeiten und Unterschiede Sorge tragen muss. Wir betrachten diesen Ansatz als Problem des überwachten Lernens, wobei das gleiche Inferenzproblem in verschiedenen biologischen Einheiten gelöst wird.

Hierfür evaluieren wir zwei Teilbereiche des Transferlernens: Zum einen betrachten wir *Domain Adaptation*, bei dem ein gerichteter Informationsaustausch zwischen *Quellen Domänen*, für welche viele Trainingsdaten vorliegen, und den *Ziel Domänen*, für welche kaum Trainingsdaten vorhanden sind, stattfindet. Zum anderen betrachten wir Methoden des *Multitask Learnings*, bei dem Informationen wechselseitig zwischen verschiedenen Domänen geteilt werden.

Für den Fall von *Domain Adaptation* entwickeln wir Erweiterungen von etablierten Algorithmen zur regularisierten Risiko Minimierung, die es erlauben einen Informationstransfer zwischen verschiedenen Domänen zu realisieren. Für diese Erweiterungen präsentieren wir effiziente numerische Algorithmen zur Klassifikation und zum *Structured Output Learning*. Besonderes Augenmerk wird hierbei auf den Fall gelegt, in welchem die Beziehung der verschiedenen Domänen oder Tasks durch eine hierarchische Struktur beschrieben wird. Dieser Fall ist in der Bioinformatik von besonderer Bedeutung, da hier oft Informationen aus verschiedenen Organismen zu kombinieren sind und deren Beziehung durch einen Stammbaum beschrieben werden kann. Wir evaluieren die vorgestellten Methoden sowohl auf synthetischen Daten, als auch in Experimenten mit genomischen Daten.

Der Ansatz des *Mulitask Learnings* ist, Modelle für mehrere verwandte Probleme gemeinsam zu lernen. Unter Verwendung von modernen Methoden aus der mathematischen Optimierung entwickeln wir ein allgemein gehaltenes Rahmenwerk, das sowohl eine Vielzahl von existierenden Multitask Learning Methoden als Spezialfälle abdeckt, sowie die Entwicklung neuer Methoden ermöglicht. Ein besonderes Merkmal unseres Rahmenwerks ist die Möglichkeit, das Ähnlichkeitsmaß zwischen Domänen unter Verwendung des *non-sparse Multiple Kernel Learnings* zu lernen, bzw. zu verfeinern. Zudem leiten wir einen effizienten Algorithmus her, der das resultierende Optimierungsproblem mit einem dualen Koordinatenabstiegsverfahren löst, und damit Neuerungen aus dem Bereich der linearen Support Vector Machine mit Multitask Learning kombiniert. Als Anwendungen unserer Methoden, betrachten wir eine Vielzahl an Problemen aus der Immuntherapie, biologischen Bildverarbeitung und Genomik.

In ihrer Gesamtheit umfasst diese Dissertation die Entwicklung von regularisierungsbasierten Methoden des Transferlernens, das Entwickeln und zur Verfügung stellen von Software, die diese Ideen umsetzt, und die erfolgreiche Anwendung auf ein breites Spektrum an Problemen.

Contents

1 Introduction and Overview **1**
- 1.1 Introduction . . . 1
- 1.2 Related Work . . . 4
- 1.3 Previously Published Work . . . 7
- 1.4 Organization of this Dissertation and Own Contributions . . . 8
- 1.5 Method Overview . . . 9

2 Background **11**
- 2.1 Primer in Sequence Biology . . . 11
- 2.2 Optimization . . . 12
  - 2.2.1 Convexity . . . 13
  - 2.2.2 Duality . . . 15
- 2.3 Supervised Learning . . . 19
  - 2.3.1 Risk Minimization . . . 19
  - 2.3.2 The Kernel Trick . . . 21
  - 2.3.3 Support Vector Machines . . . 22
  - 2.3.4 Structured Output Learning . . . 23
  - 2.3.5 Cross-validation . . . 24
- 2.4 Learning from sequences . . . 24
  - 2.4.1 String Kernels . . . 25
  - 2.4.2 Sequences and the Hashing Trick . . . 26
  - 2.4.3 Example Application: shRNA Prediction . . . 27

3 Domain Adaptation **31**
- 3.1 Source-regularized SVM . . . 31
  - 3.1.1 Dualization . . . 32
  - 3.1.2 Extension of Existing SVM Solvers . . . 35
- 3.2 Extension to Structured Output Learning . . . 38
  - 3.2.1 Structured Output Learning and Extension . . . 38
  - 3.2.2 A Bundle Method for Efficient Optimization . . . 39
- 3.3 Taxonomy-based Transfer Learning . . . 41
- 3.4 Experiments . . . 43
  - 3.4.1 Synthetic Dataset . . . 44
  - 3.4.2 Splice-sites from 15 Organisms . . . 45
  - 3.4.3 Gene Finding in Multiple Prokaryotic Genomes . . . 47
- 3.5 Summary . . . 51

4 Multi-Task Learning **53**
4.1 A Unifying View of Regularized Multi-Task Learning 54
4.1.1 Problem Setting and Notation 55
4.1.2 Dualization 56
4.1.3 Representer Theorem 59
4.1.4 Relation to Multiple Kernel Learning 59
4.1.5 Specific Instantiations of the Framework 60
4.1.6 Proposing Novel Instances of Multitask Learning Machines . . . 65
4.2 Algorithms 68
4.2.1 General Algorithms for Non-linear Kernels 69
4.2.2 A Large-scale Algorithm for Linear or String Kernels and Beyond 69
4.2.3 Convergence Analysis 74
4.3 Applications 77
4.3.1 Graph-regularized 3D Shape Reconstruction 78
4.3.2 Joint $2D/3D$ Model Learning 84
4.3.3 Execution Time Experiments 86
4.3.4 MHC-I Binding Prediction 90
4.3.5 Learning the Similarity 94
4.4 Discussion 98

5 Software **101**
5.1 Computational Considerations 102
5.2 Grid Computing 102
5.3 Availability 103
5.4 User Interaction 104

6 Conclusion **107**

Bibliography **109**

Publication List **125**

Appendix **129**

A Image licenses **131**
A.1 Sources for Figure 1.1 131
A.2 Sources for Figure 2.1 132
A.3 Sources for Figure 2.2 132
A.4 Sources for Figure 4.1 132
A.5 Sources for Figure 4.16 133

B Derivation Details **133**
B.1 Source-regularized SVM 133

C Domain-Adaptation Experimental Details **133**
C.1 Synthetic Dataset 133
C.2 Splice-sites from 15 Organisms 133
C.3 Gene Finding in Multiple Prokaryotic Genomes 134

D Multitask Learning Experimental Details **135**
D.1 GRED: graph-regularized 3D shape reconstruction 135
D.2 Joint 2D/3D Model Learning 135
D.3 Execution Time Experiments 135
D.4 MHC-I binding prediction 137
D.5 Learning the similarity 138
D.5.1 Toy Data 138
D.5.2 Power-set MT-MKL 138

1 Introduction and Overview

1.1 Introduction

In computational biology, we often study the problem of building statistical models from data in order to predict, analyze, and ultimately understand biological systems. Regardless of the specific problem, be it the recognition of splice-site sequence signals, the prediction of protein-protein interactions, or the modeling of metabolic networks, we frequently have access to data sets for multiple organisms, tissues or cell-lines. Thus, our goal is to develop methods that take advantage of the data from different domains in order to improve the performance of the statistical models built for each of these sources. We argue that, when building a predictor for a given domain, data from other domains should be incorporated to the extent of the relation between the domains.

Figure 1.1: Coupling of models between different organisms, tumor types or molecules. See Section A in the Appendix for image licenses.

Over a decade ago, an eight-year lasting collaborative effort resulted in the first completely sequenced genome of a multi-cellular organism, the free-living nematode *Caenorhabditis elegans*. Today, more than 900[1] eukaryotic genomes have been sequenced, and several large genome projects are underway, eventually making available tens of thousands of additional genomes. Genome sequences are the basis for much of the research on the molecular processes in these organisms. Typically, the more closely related the organisms are, the more similar are these processes. For some organisms, certain biochemical experiments for the analysis of particular processes can

[1] `http://www.genomesonline.org/cgi-bin/GOLD/index.cgi`

be performed more readily than for others (i.e. a large part of biological understanding was obtained from experiments based on a few model organisms such as yeast). This understanding can then be transferred to other organisms, for instance by verifying or refining models of the processes—often at a fraction of the original cost. This is but one example where the transfer of knowledge across organisms is very fruitful. Such knowledge transfer may be of particular relevance to massive genome annotation projects, such as the Genome 10k project [Haussler et al., 2008]. While genomes are typically sequenced completely in these projects, genomic annotations (e.g. coordinates of genes) are oftentimes only partially available for newly sequenced genomes. The methods presented in this thesis may be used to complement existing methods (e.g. multiple sequence alignment) to computationally transfer annotations between different organisms by learning joint models. To formalize this idea, we turn to a branch of machine learning commonly known as transfer- or multitask learning.

Thrun [1996] contributed to the emergence of the field of transfer learning in the mid-90's, when he asked the question if "Learning The n-th Thing [is] Any Easier Than Learning The First?". This was motivated by findings in human psychology that state humans are capable of applying more than just the training data for generalization, even going so far as learning based on a simple example [Ahn and Brewer, 1993]. The key insight was that humans build upon previously learned, related concepts when learning new tasks, something Thrun [1996] called *lifelong learning*. Around the same time, another influential line of work coined the term *multitask learning* [Caruana, 1993, 1997]. Rather than conceptualizing the idea of learning a sequence of related tasks, they propose machinery to learn multiple related tasks in parallel using a shared representation in the context of neural networks, as illustrated in Figure 1.2.

The rise of the kernel machines [Müller et al., 2001; Schölkopf and Smola, 2002; Vapnik, 1995], Evgeniou and Pontil [2004] brought the concept of multitask learning to the realm of regularized risk minimization [Vapnik and Chervonenkis, 1971]. In the framework, uniform task relationships were first assumed and later generalized to non-uniform relations [Evgeniou and Pontil, 2004], allowing to couple some tasks more strongly than others [Evgeniou et al., 2005]. It is conceivable that sharing information between closely related domains is more beneficial than sharing between domains that are only distantly related (according to a given criterion). Hence, it is important to take into account the degree of relatedness among the domains when obtaining the set of models. An important special case with respect to task relationships is when these relations between tasks can be described by a *hierarchical structure*. For instance, when utilizing data from different organisms, it is assumed that all of life can be traced back to an ancient common ancestor; therefore, all organisms are ultimately related by a *phylogeny*. Recently, the question of how to learn the relationship between tasks may be inferred along with the classifiers received considerable attention [Argyriou et al., 2006; Blanchard et al., 2011; Widmer et al., 2010a; Zhang and Yeung, 2010a]. When learning multiple tasks simultaneously, this question is of central importance. Often, externally available task relationships (e.g. evolutionary relationship between organisms) do not directly translate into how strongly models of different tasks should

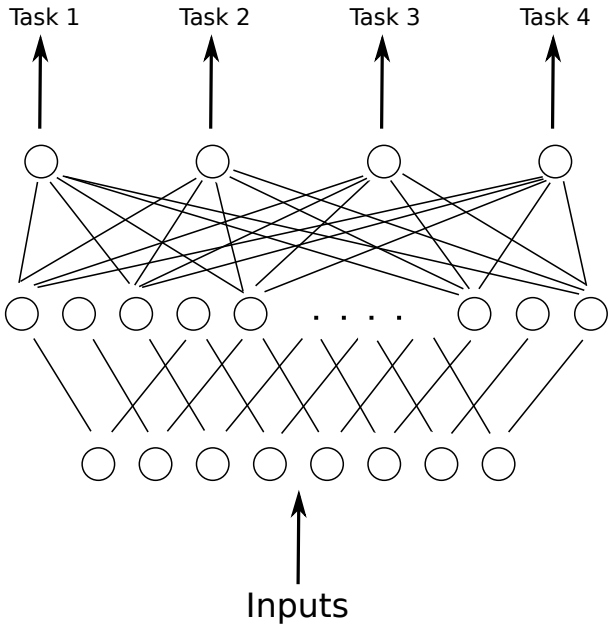


Figure 1.2: Learning a shared representation for multiple tasks with neural networks by a representation that shares some layers between tasks.

be coupled within a multitask learning framework. This also depends on other factors, such as how difficult the problems are or how much data is available.

In order to be able to build state of the art models in sequence biology, certain requirements must be met. First, the training of models must be efficient as oftentimes large numbers of training examples (i.e. $> 10^7$) are available. Second, the model has to be able to readily deal with various types of data, including biological sequences instead of solely vectorial data. Lastly, in order to combine data from multiple tasks, the model at hand needs to be readily extensible to such a scenario. As we will see, a learning machine that meets all of these requirements is the support vector machine. Their ability to incorporate arbitrary objects such as strings using kernels [Leslie et al., 2002; Rätsch et al., 2005; Rieck and Laskov, 2008; Watkins, 1999; Zien et al., 2000], their flexibility in combining kernels [Kloft et al., 2009; Sonnenburg et al., 2006], their speed due to the convexity of the underlying optimization problem and advances in solvers [Sonnenburg and Franc, 2010], and the fact that custom regularization terms can be used for transfer learning makes them perfectly suited for the applications that we have in mind. Their qualities with respect to problems from sequence biology were thoroughly demonstrated by Rätsch et al. [2005, 2007], Schultheiß et al. [2009], Schweikert et al. [2009], and Sonnenburg et al. [2007a]. For these reasons, much of the work presented in this thesis was initially motivated by the support vector machine and later extended to the more general framework of regularized risk minimization.

1.2 Related Work

Some of the earliest contributions to the simultaneous estimation of multiple statistical models came from the statistical and econometrics community [Srivastava and Dwivedi, 1979; Zellner, 1962]. Later, Brown and Zidek [1980] introduced multivariate ridge regression. A related branch of transfer learning (see the excellent overview paper by Pan and Yang [2009] to compare different branches) can be traced back to Heckman [1979] who first proposed methods to account for differences between training and test distributions (i.e. *sample selection bias*). The setting, where distributions of inputs change between training and test samples, but the conditional distributions of outputs stays the same is called *covariate shift*, a special case of non-stationary environments [Sugiyama and Kawanabe, 2012]. Here, it is assumed that no labeled examples are available on the test set, therefore this setting is also referred to as *transductive transfer learning* [Pan and Yang, 2009]. There exists a considerable body of algorithms [Gretton et al., 2009; Sugiyama et al., 2007] and theory [Sugiyama and Müller, 2005] addressing the covariate shift setting. The algorithms explored in this work, however, do not assume identical conditional distributions between tasks, but instead require (at least a small number of) labeled training examples for all tasks. This setting is commonly called *multitask learning* [Pan and Yang, 2009]. In the mid 1990s, learning multiple tasks simultaneously using multi-layered neural networks [Caruana, 1993, 1997; Thrun, 1996] attracted the attention of the machine learning community. In these multi-layered feed-forward neural networks, some of the hidden layers are shared between tasks, giving rise to a common feature representation. Since then, substantial empirical evidence accumulated over the years showing that multitask learning can considerably increase prediction accuracy.

Successful applications were reported in natural language processing [Daumé, 2007], web-document classification [Dai et al., 2007], remote sensing [Leiva-Murillo et al., 2013], computer-vision [Samek et al., 2011] and brain-computer interfaces [Samek et al., 2012]. Particularly relevant to this thesis are applications in computational biology. Pioneering work was presented by Jacob and Vert [2008a] in the context of MHC-I binding prediction. Other applications [Xu and Yang, 2011] include siRNA efficacy prediction [Liu et al., 2010], protein subcellular localization [Mei et al., 2011; Xu et al., 2011], gene expression analysis [Chen and Huang, 2010], gene network reconstruction [Tamada et al., 2005], drug-protein interaction [Bickel et al., 2008; Jacob and Vert, 2008b; Zhang et al., 2010], biomedical text mining [Dahlmeier and Ng, 2010] and bioimage analysis [Bi et al., 2008]. In addition to empirical work, a number of theoretical papers explored the properties of multitask learning algorithms, providing bounds on generalization performance [Baxter, 2000; Ben-David and Schuller, 2003].

The branch of multitask learning that this work is built upon was initiated by Evgeniou and Pontil [2004], who were the first to develop extensions of regularization-based classification methods to allow the simultaneous learning of several tasks. The idea was to learn an offset vector $\boldsymbol{v}_t$ for each task t in addition to a shared parameter vector $\mathbf{w}_0$. Evgeniou and Pontil [2004] show that this is equivalent to learning a pa-

rameter vector $\mathbf{w}_t = \mathbf{w}_0 + \boldsymbol{v}_t$ for each task, whereas the difference of the $\mathbf{w}_t$ to the average parameter vector is minimized by means of an additional regularization term $\sum_{t=1}^{T} \| \mathbf{w}_t - \frac{1}{T} \sum_{s=1}^{T} \mathbf{w}_s \|^2$. This line of work was later extended by Evgeniou et al. [2005], generalizing the multitask regularizer. Instead of uniformly enforcing similarity of all task parameter vectors $\mathbf{w}_t$ to the average parameter vector $\mathbf{w}_0$, the more general regularizer allows to set the degree of relatedness between pairs of tasks (s, t) according to a task similarity matrix $A_{s,t}$. The pairwise multitask regularizer becomes $\sum_{s,t=1}^{T} A_{s,t} \| \mathbf{w}_t - \mathbf{w}_s \|^2$. Based on this idea, Evgeniou et al. [2005] derive a number of interesting special cases, where A is chosen to represent a graph on tasks or a clustering of tasks. Furthermore, they illuminate the relationship of their regularization framework to multitask kernels using ideas from [Micchelli and Pontil, 2005]. In a paper titled *frustratingly easy domain adaptation*, Daumé [2007] presented a special case of this regularizer for the two task setting along with the corresponding augmented kernel. In a related approach, Jacob and Vert [2008a] explore the combination of several multitask kernels, in an application to computational biology. As they do not fully elucidate the relationship to the corresponding regularizer, we fill this gap in later chapters.

A somewhat orthogonal approach to multitask learning is multitask feature selection originating in early work by Jebara [2004], which was later extended by Argyriou et al. [2006]. While the underlying assumption in the aforementioned approaches is similarity of parameter vectors, multitask feature selection assumes that predictors for different tasks share a common set of features (similar approaches were studied in [Obozinski et al., 2010; Obozinski et al., 2007]). The principle approach to this line of multitask learning is $l_{2,1}$ regularization on the matrix of parameter vectors $\|\boldsymbol{W}\|_{2,1} = \sum_{i=1}^{d} \| \mathbf{w}^i \|$, i.e. rows containing the ith feature of parameter vectors $\mathbf{w}_t$ are regularized using the l_2 norm, effectively spreading the weights between tasks. This gives a vector of l_2 norms for each feature on which an additional l_1 norm is applied, effectively penalizing non-zero features. As with Lasso [Tibshirani, 1996] in the single-task case, this can be used to select a small number of features. A equivalent convex formulation of the otherwise non-convex optimization problem (non-convexity can also be avoided by using a fix common set of variables between tasks) is given in [Argyriou et al., 2008] and further advances in solvers are presented in [Liu et al., 2009]. This line of research was further explored in various directions, among them robust feature selection [Nie et al., 2010] (i.e. assuming that irrelevant tasks are present) or accoutning for outlier tasks using a mixture of sparse and group-sparse components [Jalali et al., 2010].

There also exists a number of bayesian approaches to multitask learning, where probability distributions are used to model the relationship between different tasks [Bakker and Heskes, 2003; Daumé III, 2009; Zhang and Yeung, 2010b]. A particularly interesting approach [Bonilla et al., 2007] attempts to model the task relationship as the covariance matrix of a gaussian process, however, this also suffers from the disadvantages of a non-convex optimization problem.

Going beyond bayesian methods, the research question of learning the similarity between tasks (to which we have also contributed) has recently enjoyed considerable attention. One of the first approaches to that end was Alternating Structure Optimiza-

tion (ASO) by Ando and Zhang [2005], which aims at identifying a common subspace between tasks along with learning individual predictor, unfortunately resulting in a non-convex optimization problem. A convex relaxation was presented in [Chen et al., 2009]. ASO was later shown [Zhou et al., 2011] to be equivalent to clustered multitask learning [Jacob et al., 2008]; an approach to identify clusters of related tasks from the data. A more general approach of learning task similarity was presented by Zhang and Yeung [2010a], who learn task similarity using trace-norm regularization to maintain convexity and enforce low-rank structure. Other approaches attempt to learn task similarity prior to model training in a two step procedure [Blanchard et al., 2011]. The methods presented in this thesis constitute a middle ground between assuming a fixed task relationship and learning the task relationship from *ab initio*, as we will see in later chapters.

1.3 Previously Published Work

This thesis is based on the following *selected* publications.
The work on domain adaptation was published in (equal contributions are indicated by *):

> [1] G., Schweikert*, **C. Widmer***., B. Schölkopf, and G. Rätsch. An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis. In *Advances in Neural Information Processing Systems 21*, pages 1433-1440, 2008.
>
> [2] **C. Widmer**, J. Leiva., Y. Altun, and G. Rätsch. Leveraging Sequence Classification by Taxonomy-based Multitask Learning. In *Research in Computational Molecular Biology*, pages 522-534, 2010.
>
> [3] N. Görnitz*, **C. Widmer***., G. Zeller, S. Sonnenburg, and G. Rätsch. Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation. In *Advances in Neural Information Processing Systems 24*, pages 2690–2698, 2011.

An initial empirical survey was conducted in [1]. Based on these findings, we developed algorithms for the case of hierarchical task relationships in [2]. The method was extended to structured output learning in [3].

Extensions to multitask learning methods are based on:

> [2] **C. Widmer**, N. C. Toussaint, Y. Altun, and G. Rätsch. Inferring latent task structure for Multitask Learning by Multiple Kernel Learning. In *BMC bioinformatics*, 11 Suppl 8(Suppl 8), 2010.
>
> [4] **C. Widmer**, M. Kloft, N. Görnitz and G. Rätsch. Efficient Training of Graph-Regularized Multitask SVMs. *European Conference on Machine Learning*, 2012.
>
> [5] **C. Widmer**, M. Kloft, G. Rätsch. A general framework for Multitask Multiple Kernel Learning. *Journal of Machine Learning Research* (JMLR), in preparation.

Applications to computational biology and bioimaging were discussed in:

> [6] **C. Widmer**, N. C. Toussaint, Y. Altun, O. Kohlbacher and G. Rätsch. Novel machine learning methods for MHC Class I binding prediction. In *Pattern Recognition in Bioinformatics*, pages 98-109, 2010.
>
> [7] X. Lou, **C. Widmer**, M. Kang, G. Rätsch, and A.K. Hadjantonakis. Structured Domain Adaptation Across Imaging Modality: How 2D Data Helps 3D Inference. In *NIPS MLCB Workshop*, 2012.
>
> [8] S. Heinrich, E.-M. Geissen, J. Kamenz, S. Trautmann, **C. Widmer**, P. Drewe, . . . , S. Hauf. Determinants of robustness in spindle assembly checkpoint signalling. In *Nature Cell Biology*, 15(11), 1328-39, 2013.

NOTE*: a complete list publications is shown at the end of the bibliography section.*

1.4 Organization of this Dissertation and Own Contributions

The central research question addressed in this thesis is how to bring together information from several related biological entities to obtain better models. Our contribution to that end carefully explores methods based on regularization-based multitask learning to attack that problem and show its application to manifold problems from computational biology and bioimage analysis. Parts of this work were summarized in overview paper [Widmer and Rätsch, 2012] and invited contributions [Widmer et al., 2013b,c].

Background In Chapter 2, we give an overview of the core concepts and techniques that this thesis is built upon, including a primer in sequence biology, an outline of important techniques from mathematical optimization, supervised learning and machinery to learn from biological sequences. We conclude the background chapter with a short excursion to a single-task application of string-kernels to an important problem from cancer biology [Pelossof* et al., 2014, in prep.].

Domain Adaptation In Chapter 3, we develop regularization-based techniques to adapt a given classifier to a new domain. To the best of our knowledge, we were the first to explore such techniques to transfer information between several organisms in Schweikert* et al. [2009]. We build upon this work and derived an algorithm to transfer knowledge between organisms, while respecting their evolutionary relationships in Widmer et al. [2010b]. These methods were extensions of single-task sequence-based classification algorithms; therefore, we extend the underlying solvers to handle information transfer. Our contributions are available as part of the SHOGUN Machine Learning Toolbox [Sonnenburg et al., 2010]. We then generalize the idea of regularization based domain adaptation from binary classification to structured output learning in [Görnitz* et al., 2011]. We evaluate our methods on problems from sequence biology and prokaryotic gene finding.

Multitask Learning In Chapter 4, we explore regularization-based methods to jointly learn models for a set of related tasks. Based on our experience with different multitask learning methods over the years [Widmer and Rätsch, 2012], we derive a general framework and show its relation to many established multitask learning methods as special cases [Widmer et al., 2014, in prep.]. Using this general formulation, we make several contributions. We propose an efficient dual-coordinate descent solver for the graph-regularized SVM formulation, which achieves considerable speed-ups over existing solvers [Widmer et al., 2012]. An interesting property of our general framework is its ability to learn or refine task similarities using non-sparse multiple kernel learning along with the multitask classifiers, a prototype of which we propose in [Widmer et al., 2010a] and which we further investigate in [Widmer et al., 2013b, 2014, in prep.]. To address the central research question of how to transfer information between related biological entities, we perform a series of experiments in different application domains. In close collaboration with biologists, we combine the ideas of multitask learning with

techniques from computer vision and software engineering to develop GRED [Widmer et al., 2013a], a software tool to rapidly analyze images from fluorescense microscopy, resulting in a high-impact publication [Heinrich et al., 2013]. In a second application to bioimaging, we ask the question if multitask learning may be used as a means of reducing labeling cost by combining existing $2d$ annotations with newly curated $3d$ annotations to improve model quality [Lou et al., 2012b]. In MHC-I binding prediction, a problem relevant to immunotherapy [Widmer et al., 2010c], we improve upon existing methods by combining an improved multitask learning approach with a novel string kernel [Toussaint et al., 2010]. Lastly, we explore novel instances derived from our general framework on synthetic data as well as on data from sequence biology.

Software In Chapter 5, we discuss an aspect of this thesis that is often overlooked: the software developed as part of this thesis. We outline the computational considerations, strategies for cluster computing and how it is made available. In the process of this work, we published a python tool to facilitate parallel computations in a cluster computing environment [Widmer and Ong, 2013]. Finally, a major contribution of this work is the availability of efficient implementations as open-source software as part of the SHOGUN Machine Learning Toolbox [Sonnenburg et al., 2010].

1.5 Method Overview

From a bird's eye view, the methods presented in this thesis are embedded into the general framework of regularization-based supervised learning methods, where we minimize a functional

$$\mathfrak{R}(\mathbf{w}) + C\,\mathfrak{L}(\mathbf{w})\,,$$

which consists of a loss-term $\mathfrak{L}(\mathbf{w})$ measuring the training error and a regularizer $\mathfrak{R}(\mathbf{w})$ penalizing the complexity of the model $\mathbf{w}$. The positive constant $C > 0$ controls the trade-off between the two terms. Prominent supervised learning methods such as the support vector machine (SVM) [Boser et al., 1992; Cortes and Vapnik, 1995; Müller et al., 2001; Schölkopf and Smola, 2002; Vapnik, 1995], logistic regression, ridge regression or Lasso [Tibshirani, 1996], are subsumed by this framework, depending on the chosen regularizer and loss functions. The methods explored in this work augment the above formulation by introducing additional regularization terms that penalize differences between model parameters of different tasks.

Domain Adaptation In the *domain adaptation* setting, we adapt a previously trained classifier to a new problem [Thrun, 1996]. We approach this problem by introducing an additional regularization term, that penalizes the deviation of the current parameter vector $\mathbf{w}$ from a given source parameter vector $\mathbf{w}_s$. The functional to be minimized now becomes

$$J(\mathbf{w}) = \mathfrak{R}(\mathbf{w}) + B\,\mathfrak{R}_{DA}(\mathbf{w}, \mathbf{w}_s) + C\,\mathfrak{L}(\mathbf{w}),$$

where positive constant B controls the influence of the newly introduced term.

Multitask Learning In case of *Multitask Learning* [Caruana, 1997] we are interested in obtaining several models simultaneously. These models are parameterized by $\mathbf{w}_1, \ldots, \mathbf{w}_T$, where T is the number of tasks. The single-task formulation can be extended by introducing a joint regularization term $\mathfrak{R}_{MTL}(\mathbf{w}_1, \ldots, \mathbf{w}_T)$ that penalizes the discrepancy between the individual models [Agarwal et al., 2010; Evgeniou et al., 2005]. The objective function becomes

$$J(\mathbf{w}_1, \ldots, \mathbf{w}_M) = \mathfrak{R}_{MTL}(\mathbf{w}_1, \ldots, \mathbf{w}_M \,|Q) + \sum_{i=1}^{T} \mathfrak{R}(\mathbf{w}_t) + \sum_{t=1}^{T} C\, \mathfrak{L}(\mathbf{w}_t).$$

A common approach is, for example, to set $\mathfrak{R}_{MTL}(\mathbf{w}_1, \ldots, \mathbf{w}_T) = \frac{1}{2} \sum_{s,t=1}^{T} q_{st} \left\|\mathbf{w}_s - \mathbf{w}_t\right\|^2$, where $Q = (q_{st})_{a \leq s,t \leq T}$ is a task similarity matrix.

Multitask Multiple Kernel Learning Finally, we develop a novel, general framework for multi-task learning of the form

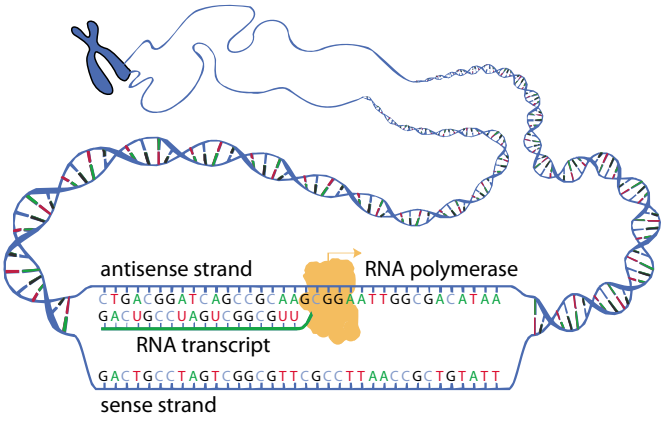
$$J(\mathbf{w}_1, \ldots, \mathbf{w}_M, \boldsymbol{\theta}) = \sum_{m=1}^{M} \frac{\mathfrak{R}_{MTL}(\mathbf{w}_1, \ldots, \mathbf{w}_M \,|Q_m)}{\theta_m} + \sum_{i=1}^{T} \mathfrak{R}(\mathbf{w}_t) + \sum_{t=1}^{T} C\, \mathfrak{L}(\mathbf{w}_t),$$

where Q_m are different task similarity matrices. This approach has the additional flexibility of incorporating *multiple task similarity measures* into the learning problem, each equipped with a weighting factor Θ_m [Widmer et al., 2010a, 2013b, 2014, in prep.]. In this framework, we are able to derive existing MTL methods as special cases, but also develop novel instantiations that exploit the additional flexibility.

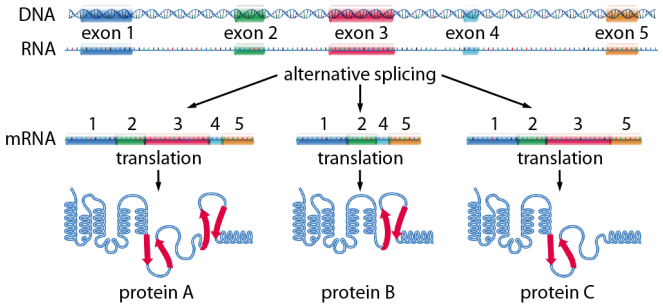
2 Background

2.1 Primer in Sequence Biology

As most experiments in this thesis were be carried out on biological data and most algorithms were designed for biological challenges, we provide an elementary background in sequence biology, including necessary pointers to acquire the understanding required for the experiments performed in later sections.



(a) Transcription.



(b) RNA splicing.

Figure 2.1: Illustrations are taken from Wikipedia [2011] and Institute [2011]. See Section A in the Appendix for image licenses.

The *Central Dogma* of molecular biology was proposed by Crick [1970] and describes the *flow of genetic information* as illustrated in Figure 2.1. The blueprint of all life is

encoded in the genome, a collection of long *DNA* strings over the alphabet {A,G,T,C}, corresponding the bases *adenine*, *guanine*, *thymine*, and *cytosine*. The genome is organized in distinct *chromosomes* located in the cell's *nucleus*, which also contains the cellular apparatus necessary to interpret the information encoded in it. Certain functional segments on the genome, referred to as *genes*, are interpreted by this apparatus. Molecular machines known as *enzymes* recognize certain patterns on the DNA indicating the start of a gene, the *promoter sequence*, and *transcribe* the information encoded in DNA into an intermediate form called *RNA*. The encoding of RNA is very similar to that of DNA, as it is also encoded by 4 bases, however they differ in one base as RNA replaces *thymine* with *uracil*.

In some cases, a string of RNA is the final gene product, for example, *shRNAs* are short RNA molecules that are complementary to other RNAs and may be used to *silence* certain genes. In other cases, the RNA of so-called *protein-coding genes* is further processed by the cell to become a *protein*, which typically makes up the molecular machines performing various tasks in the cell in combination with other proteins or RNAs. The RNA molecules of protein-coding genes undergo another processing step called *splicing*. Certain segments of the RNA molecule, the *introns*, are cut out, while the *exons* are combined into *messenger RNA*. The positions where the molecule is cut are marked by *splice-sites*, certain patterns on the RNA molecule, which are recognized by the *spliceosome*, the apparatus performing the splicing operation. Depending on the state of the cell, the RNA molecule is spliced in different ways, giving rise to *alternative splicing*. This mechanism enables the cell to produce a more diverse set of proteins coded by combining common building blocks. To get from mRNA to protein, another transcoding operation is necessary. The *ribosome*, another molecular machine, interprets the mRNA and *translates* triplets of bases into amino acids. While a triplet over an alphabet of size 4 has $4^3 = 64$ possible combinations, there are only 20 encoded amino acids. Because of this, the genetic code is said to be *redundant* in the sense that several triplets encode for the same amino acid, which also happens to be the reason why mutations changing the encoded amino acid may have a more severe effect than mutations that change a nucleotide, while still coding for the same amino acid. Again, this transcoding operation is guided by certain signatures on the RNA molecule that are recognized by the molecular machinery, telling it where to start and stop the translation into protein. As we will see in later chapters, using methods from machine learning, we will be able to learn to recognize these signatures from labeled data.

Each of the steps described above is subject to *regulation*, meaning each step depends on a number of factors that may be influenced by internal or externals cellular signals, providing the cell with powerful mechanisms to change *gene expression* as part of *development* or in response to external factors, such as *heat-stress*.

2.2 Optimization

At the turn of the millennium, the widely recognized article on the *two cultures* outlined the differences between the statistical community and the emerging machine learning

community [Breiman, 2001]. The authors criticize that classical statistics departments produced a large body of theory derived from questionable probabilistic modeling assumptions, while the machine learning community with neural networks, random forests and support vector machines focused on improving prediction accuracy and the development of efficient learning algorithms. While it may be argued that the two communities have converged in recent years, it is unquestionable that a remarkable co-evolution occurred between the field of machine learning and efficient algorithms from mathematical optimization as nicely illuminated by Bennett and Parrado-Hernández [2006]. For many methods developed in this thesis, we make heavy use of ideas and concepts from mathematical optimization. Thus, we will review basic concepts based on [Boyd, S.P. and Vandenberghe, 2004], which is one of the standard works on the subject. We begin by stating the most general form of an optimization problem

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{aligned} \tag{2.1}$$

where vector $x \in \mathbb{R}^d$ is the *optimization variable* of the problem, the function $f_0 : \mathbb{R}^d \to \mathbb{R}$ is the *objective* function, $f_i : \mathbb{R}^d \to \mathbb{R}, i = 1, \ldots, m$ are the *inequality constraints* and the $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ are the bounds for the constraints.

2.2.1 Convexity

In a review paper from the 1990s, Rockafellar [1993] stated that *"In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"*. As we will see, with one exception, we are interested in *convex* optimization problems in which the objective and the constraint functions are convex, which means they satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y),$$

for all $x, y \in \mathbb{R}^d$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$ and $\alpha \geq 0, \beta \geq 0$. Intuitively, a convex function is one where a straight line between any two points on the graph of a function must entirely lie above (or on) the graph of the function. Convex optimization problems have the highly desirable property that any local optimum is also globally optimal, giving us a guarantee that we have found the optimal solution. Furthermore, from a practical point of view, most problems that can be formulated as a convex optimization problem can be solved efficiently.

The following types of convex optimization problems will play a role in this thesis, therefore we will briefly outline their properties.

Linear Program (LP) A *Linear Program* consists of an affine objective function and constraints. Clearly, an LP is a convex program as linear functions are both, concave

and convex. A LP in standard form is written as

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \succeq 0, \end{aligned}$$

where $A \in \mathbb{R}^{p \times n}$ [Boyd, S.P. and Vandenberghe, 2004]. Note that any LP may be rewritten in standard form by using techniques for manipulating optimization problems. For example, introducing additional variables called *slack variables* with the goal of eliminating inequality constraints other than $x \geq 0$. From a practical point of view, efficient LP solvers (based on simplex or interior-point algorithms) exist that handle millions of variables and are implemented in freely available software libraries such as GLPK [Makhorin, 2006].

Quadratic Program (QP) Quadratic programs (QPs) are among the most common optimization problems encountered in machine learning. They consist of a quadratic objective and affine constraint functions.

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x + b^T x + r \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b, \end{aligned}$$

where $Q \in S_+^n$ (i.e. in the cone of positive-semidefinite matrices), $G \in \mathbb{R}^{m \times n}$, and $A \in \mathbb{R}^{p \times n}$ [Boyd, S.P. and Vandenberghe, 2004]. Since Q coincides with the Hessian of the objective, the objective is clearly only convex if Q is positive-semidefinite and non-convex otherwise (which is precisely why we restrict ourselves to $Q \in S_+^n$). As we will see in Section 2.3.1, QPs play a central role in *regularized risk minimization*, an important branch of machine learning. A large body of research in machine learning is concerned with deriving efficient solvers for particular QPs, exploiting special problem structure to make solvers efficient in terms of time and memory usage.

Semi-Infinite Linear Program (SILP) Semi-infinite linear programs have a finite number of variables, a linear objective and an infinite number of linear constraints [Bennett and Parrado-Hernández, 2006].

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subject to} \quad & ax = 0 \quad a \in \mathcal{A} \\ & b^T x \succeq 0 \quad b \in \mathcal{B}, \end{aligned}$$

where $a \in \mathcal{A}$ and $b \in \mathcal{B}$ are possibly infinite sets of vectors [Bennett and Parrado-Hernández, 2006].

2.2.2 Duality

Duality is a concept that is heavily used in Machine Learning for developing efficient solvers and gaining new insights and interpretations for learning machines. The basic idea of *Lagrangian duality* is to extend the objective function by a linear combination of constraint functions f_i, yielding the *Lagrangian* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ (with respect to Problem 2.2.2)

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x), \tag{2.2}$$

where λ_i is the Lagrange multiplier associated with the ith inequality constraint and ν_i associated with the ith equality constraint, respectively. The *Lagrange dual function* g is attained by considering the Lagrangian at minimal value of x:

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left\{ f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right\}, \tag{2.3}$$

The optimal value of the dual function (λ^*, ν^*) gives us a lower bound on the optimal value of the optimization problem,

$$f_0(x^*) \leq g(\lambda, \nu), \tag{2.4}$$

for any (λ, ν). We now define the *Lagrange dual problem* as

$$\begin{aligned} \max_{\lambda, \nu} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \succeq 0. \end{aligned}$$

Note that the Lagrange dual problem is a convex optimization problem even if the primal problem is not convex. We denote the objective at the optimum of the primal and dual problems as p^* and d^*, respectively. Clearly, the greater g, the tighter the bound $f_0(x^*) \leq g(\lambda, \nu)$, thus the best we can do is

$$p^* \leq d^*, \tag{2.5}$$

a property called *weak duality*. The difference $p^* - d^*$ is called the optimal *duality gap*. Under certain conditions, equality holds (i.e. $p^* = d^*$), which is called *strong duality*. *Slater's theorem* states that if the primal problem is convex and the so-called *Slater's conditions* hold, then strong duality holds (note that there are other ways to imply strong duality). *Slater's conditions* state that there has to be an x, which is feasible (i.e. the inequality and equality constraints are fulfilled) and for which the inequality constraints are not *tight*:

$$\exists x \in D : \quad f_i(x) < 0, \quad i = 1, \ldots, m, \quad Ax = b. \tag{2.6}$$

We can now use strong duality to ensure optimality, as a duality gap of zero tells us that x is primal optimal and (λ, ν) is dual optimal. This is often used in numerical algorithms, using stopping criteria based on the duality gap (e.g. $p - d < \epsilon$).

For convex optimization problems (assuming differentiable objective and constraint functions), the *Karush-Kuhn-Tucker* (KKT) conditions are necessary and sufficient to ensure optimality (primal and dual).

$$f_i(\tilde{x}) \leq 0, \quad i = 1, \ldots, m \tag{2.7}$$

$$h_i(\tilde{x}) = 0, \quad i = 1, \ldots, p \tag{2.8}$$

$$\tilde{\lambda} \geq 0, \quad i = 1, \ldots, m \tag{2.9}$$

$$\tilde{\lambda} f_i(\tilde{x}) = 0, \quad i = 1, \ldots, m \tag{2.10}$$

$$\nabla f_0(\tilde{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{x}) + \sum_{i=1}^{p} \tilde{\nu}_i \nabla h_i(\tilde{x}) = 0. \tag{2.11}$$

The first two conditions in Equation 2.7, 2.8 ensure primal feasibility, 2.9 ensures the non-negativity of the Lagrange multiplier, 2.10 enforces *complementary slackness* (either the Lagrange multiplier $\tilde{\lambda}_i$ is zero, or the corresponding inequality constraint is tight $f_i(\tilde{x}) = 0$) and finally the condition in Equation 2.11 can be derived from the observation that the gradient of the Lagrangian L must vanish at x^*.

Duality and conjugate functions There is a striking relationship between duality and conjugate functions, which is formalized in Fenchel's duality theorem. The *conjugate* f^* of a function $f : \mathbb{R} \to \mathbb{R}$ is defined as

$$f^*(y) = \sup_x (y^T x - f(x)). \tag{2.12}$$

Let us first consider a useful example, starting with an optimization problem with linear equality and inequality constraints:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & Ax \leq b \\ & Cx = d \end{aligned}$$

Using the definition of a conjugate function given in Equation 2.12, we show the relationship to the Lagrange dual function [Boyd, S.P. and Vandenberghe, 2004]:

$$\begin{aligned} g(\lambda, \nu) &= \inf_{x \in D} \left\{ f_0(x) + \lambda^T (Ax - b) + \nu (Cx - d) \right\} \\ &= -b^T \lambda - d^T \nu + \inf_x \left\{ f_0(x) + (A^T \lambda + C^T \nu)^T x \right\} \\ &= -b^T \lambda - d^T \nu - f_0^*(-A^T \lambda - C^T \nu). \end{aligned}$$

Note that in the last step, we are able to replace the $\inf_x$ term with the definition of the conjugate function as given in Equation 2.12.

Fenchel's duality theorem The relationship between conjugate functions and duality is captured by Fenchel's duality theorem [Rockafellar, 1970], which states that for any two proper (i.e. $\text{dom} f = \{x \in X | f(x) \in \mathbb{R}\} \neq \emptyset$) convex functions f and g, we have

$$\inf_x (f(Ax) + g(x)) = \sup_\lambda (-f^*(-\lambda) - g^*(A^T\lambda)), \tag{2.13}$$

where f^* and g^* are the conjugate functions of f and g. Due to its importance, it is worthwhile to review the proof of Fenchel's theorem, which is based on [Tomioka, 2010]. We start by first introducing an auxiliary variable z:

$$\begin{aligned} \min_{x,z} \quad & f(z) + g(x) \\ \text{subject to} \quad & Ax = z \end{aligned}$$

We prove Fenchel's theorem using the same reasoning as in the previous example.

$$\begin{aligned} g(\lambda, \nu) &= \inf_{x,z} \left\{ f(z) + g(x) + \lambda^T (z - Ax) \right\} \\ &= \inf_z (f(z) + \lambda^T z) + \inf_x (g(x) - \lambda^T Ax) \\ &= -\sup_z (-\lambda^T z - f(z)) - \sup_x (\lambda^T Ax - g(x)) \\ &= -f^*(-\lambda) - g^*(A^T\lambda). \end{aligned}$$

If both f and g are convex and *Slater's condition* holds, then we have strong duality and Fenchel's theorem given in Equation 2.13 holds.

As we will see in Section 2.3.1, optimization problems involving two convex components f and g are ubiquitous in regularized-risk minimization. An insightful work elaborating the application of Fenchel's theorem to problems from Machine Learning was presented by Rifkin and Lippert [2007], who derived conjugate functions for a whole family of functions (loss-functions and regularizers) relevant to Machine Learning.

Fenchel Duality in Hilbert Spaces A more general variant of Fenchel's theorem is given in Lewis [1996], assuming general linear operators instead of Euclidean spaces. In later chapters of this thesis, we require an even more general version of Fenchel Duality. We now briefly review Fenchel duality theory for convex functions over real Hilbert spaces. This section is based on Chapters 15 and 19 in Bauschke and Combettes [2011]. For complementary reading, we recommend the excellent introduction of Bauschke and Lucet [2012]. We start with the definition of the convex conjugate function.

Definition 2.2.1 (Convex conjugate)**.** *Let $\mathcal{H}$ be a real Hilbert space and let $g : \mathcal{H} \to \mathbb{R} \cup \{\infty\}$ be a convex function. We assume in the whole section that g is proper, that is, $\{\mathbf{w} \in \mathcal{H} \,|\, g(\mathbf{w}) \in \mathbb{R}\} \neq \emptyset$. Then the convex conjugate $g^* : \mathcal{H} \to \mathbb{R} \cup \{\infty\}$ is defined by $g^*(\mathbf{w}) = \sup_{\boldsymbol{v} \in \mathcal{H}} \langle \boldsymbol{v}, \mathbf{w} \rangle - g^*(\boldsymbol{v})$.*

As the convex conjugate is a supremum over affine functions, it is convex and lower semi-continuous. We have the duality

$$g = g^{**} \Leftrightarrow \begin{cases} g \text{ is convex and} \\ \text{lower semi-continuous}\,. \end{cases}$$

This indicates that the "right domain" to study conjugate functions is the set of convex, lower semi-continuous, and proper functions. In order to present the main result of this section, we need the following standard result from operator theory.

Proposition 2.2.2 (Definition and uniqueness of the adjoint map)**.** *Let $\mathcal{H}$ be a real Hilbert space and let $A : \mathcal{H} \to \widetilde{\mathcal{H}}$ be a continuous linear map. Then there exists a unique continuous linear map $A^* : \widetilde{\mathcal{H}} \to \mathcal{H}$ with $\langle A(\mathbf{w}), \boldsymbol{\alpha}\rangle = \langle \mathbf{w}, A^*\boldsymbol{\alpha}\rangle$, which is called* adjoint map *of A.*

For example, in the Euclidean case, we have $\mathcal{H} = \mathbb{R}^m$, $\widetilde{\mathcal{H}} = \mathbb{R}^n$, and $A \in \mathbb{R}^{m\times n}$ so A^* becomes the transpose $A^* = A^\top \in \mathbb{R}^{n\times m}$. We now present the main result of this section, which is the general variant of *Fenchel's duality theorem*:

Theorem 2.2.3 (Fenchel's duality theorem)**.** *Let $\mathcal{H}, \widetilde{\mathcal{H}}$ be real Hilbert spaces and let $g : \mathcal{H} \to \mathbb{R}\cup\{\infty\}$ and $h : \widetilde{\mathcal{H}} \to \mathbb{R}\cup\{\infty\}$ be ccp. Let $A : \mathcal{H} \to \widetilde{\mathcal{H}}$ be a continuous linear map. Then the primal and dual problems,*

$$\begin{aligned} p^* &= \inf_{\mathbf{w}\in\mathcal{H}} \; g(\mathbf{w}) + h(A(\mathbf{w})) \\ d^* &= \sup_{\boldsymbol{\alpha}\in\widetilde{\mathcal{H}}} \; -g^*(A^*(\boldsymbol{\alpha})) - h^*(-\boldsymbol{\alpha})\,, \end{aligned}$$

satisfy weak duality *(i.e., $d^* \le p^*$). Assume, furthermore, that $A(dom(g)) \cap cont(h) \neq \emptyset$, where* $\mathrm{dom}(f) := \{\mathbf{w} \in \mathcal{H} : g(\mathbf{w}) < \infty\}$ *and* $\mathrm{cont(h)} := \{\boldsymbol{\alpha} \in \widetilde{\mathcal{H}} : h$ *continuous in* $\boldsymbol{\alpha}\}$*. Then we even have* strong duality *(i.e., $d^* = p^*$) and any optimal solution $(\mathbf{w}^\star, \boldsymbol{\alpha}^\star)$ satisfies*

$$\mathbf{w}^\star = \nabla g^*(A^*(\boldsymbol{\alpha}^\star))\,,$$

if $g^* \circ A^*$ is (Gâteaux) differentiable in $\boldsymbol{\alpha}^\star$.

When applying Fenchel duality theory, we frequently need to compute the convex conjugates of certain functions. To this end, the following computational rules are helpful.

Proposition 2.2.4. *The following rules hold for the convex conjugate:*

1. *Let $g : \mathcal{H} \to \mathbb{R}\cup\{\infty\}$ be a proper convex function on a real Hilbert space $\mathcal{H}$. Then, for any $\lambda > 0$ and $\mathbf{w} \in \mathcal{H}$, we have $(\lambda g)^*(\mathbf{w}) = \lambda h^*(\mathbf{w}/\lambda)$.*
2. *Furthermore, assume that $\mathcal{H} = \mathcal{H}_1 \bigoplus \mathcal{H}_2$ and $g(\mathbf{w}) = g_1(\mathbf{w}_1) + g_2(\mathbf{w}_2)$, where $g_1 : \mathcal{H}_1 \to \mathbb{R}\cup\{\infty\}$ and $g_2 : \mathcal{H}_2 \to \mathbb{R}\cup\{\infty\}$, are proper convex functions on Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. Then, for any $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2) \in \mathcal{H}_1 \bigoplus \mathcal{H}_2$, we have $g^*(\mathbf{w}) = g_1^*(\mathbf{w}_1) + g_2^*(\mathbf{w}_2)$.*

2.3 Supervised Learning

We now move on to introducing the idea of learning from data. In supervised learning, we are interested in learning to predict a target variable $y \in \mathcal{Y}$, which may be binary (or categorical) in the case of classification, real-valued in the case of regression or structured (e.g. a graph) in the case of structured output learning. We are given a sample $D = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ of observations, which are pairs $(\boldsymbol{x}, y)$ of examples $\boldsymbol{x} \in \mathcal{X}$ and target variables y. These pairs are assumed to be drawn independently and identically distributed (i.i.d.) from a fixed, but unknown distribution $P(\boldsymbol{x}, y)$. Given a finite sample D *training a model* equates to choosing a function f from a function class $\mathcal{F}$, such that a certain utility function is minimized [Guyon et al., 2010].

2.3.1 Risk Minimization

In the *frequentist* approach (which is the route taken in this thesis), function f is chosen such that the *expected risk* or *generalization error* is minimized:

$$R(f) = E[L(f(\boldsymbol{x}), y)] = \int L(f(\boldsymbol{x}), y) dP(\boldsymbol{x}, y),$$

where $L : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is a *loss function* that quantifies the discrepancy between $f(\boldsymbol{x})$ and y. In other words, the expected risk is the expectation of the loss function with respect to an infinite number of example-label pairs drawn from distribution $P(\boldsymbol{x}, y)$.

Empirical Risk Minimization As $P(\boldsymbol{x}, y)$ is typically unknown, we resort to approximating $P(\boldsymbol{x}, y)$ using the given sample D. Assuming independent and identically distributed (i.i.d.) random variables, we get the *empirical risk*

$$R_{\text{emp}}(f) = \sum_{(\boldsymbol{x}, y) \in D} L(f(\boldsymbol{x}), y).$$

When minimizing empirical risk on a sample D, an immediate problem arises from the fact that for a sufficiently large function class $\mathcal{F}$, the empirical risk may be very small. Clearly, this does not mean that our chosen function f will then perform well on unseen data. This is the effect of *overfitting*. Generally, it holds that $R_{\text{emp}}(f) \leq R(f)$, i.e. the empirical risk underestimates actual risk. Luckily, as we will see in the next paragraph, theory derives an *upper bound* on the expected risk $R(f)$ depending on the *capacity* of function class $\mathcal{F}$.

Regularized risk minimization What we are ultimately interested in is the ability to perform well on previously unseen data. For this, it is not sufficient to simply minimize empirical risk on our training data set but it is imperative to also restrict the function class $\mathcal{F}$ from where we choose f. A theory that laid much of the foundation of this line of machine learning [Schölkopf et al., 2013] is statistical learning theory [Vapnik, 1995]. It provides the tools to derive theoretical performance bounds on the generalization

error based on the *capacity* of a function class. For instance, in case of classification, Chervonenkis and Vapnik [1971] define the *VC-dimension* to capture the capacity of a function class as the number of points that can be shattered (separated in all possible ways). A widely used strategy to control capacity is *regularization*, where a term is added in addition to the term quantifying empirical risk. The idea of regularization was first used in the context of ill-posed equation systems [Tikhonov, 1943], and can be formalized as

$$\min_f \lambda\Omega(f) + R_{\text{emp}}(f), \tag{2.14}$$

where λ is a regularization parameter for controlling the trade-off between the two terms. In statistics, a related concept is the *bias-variance trade-off* [Geman et al., 1992]. It is related in the sense that a regularizer Ω leads to a biased predictor, while an unbiased predictor might suffer from high variance and thus poor generalization performance.

Definition 2.3.1 (Bias-variance trade-off)**.** *Predictor f can be understood as an estimator of the expectation $E(y|\boldsymbol{x})$. Let $f(\boldsymbol{x}|D)$ denote a model f trained on data set D of a given size t and let $E_D[.]$ be the expected value taken over all data sets of size t drawn according to the distribution $P(\boldsymbol{x}, y)$. It holds that:*

$$E_D[(f(\boldsymbol{x}|D) - E[y|\boldsymbol{x}])^2] = \underbrace{(E_D[f(\boldsymbol{x}|D)] - E[y|\boldsymbol{x}])^2}_{bias^2} + \underbrace{E_D[(f(\boldsymbol{x}|D) - E_D[f(\boldsymbol{x}|D)])^2]}_{variance}.$$

Often, there is an optimal compromise between bias and variance [Guyon, 2009]. In practice, the regularization parameter λ is tuned in *model selection*, as we will see in Section 2.3.5. The connection between regularization and kernels was explored by Smola et al. [1998].

An intriguing connection to Bayesian methods is that the regularizer may be interpreted as *prior* distribution on f, and inference of f using the above optimization problem may be viewed as *Maximum a-posteriori* estimate, where one obtains a point estimate maximizing the posterior. It is remarkable that a great number of existing machine learning methods can be expressed by choosing different combinations of regularization term and loss-function, helping to develop a "big picture" [Mika et al., 2000]. Different loss functions have different properties with respect to their robustness to outliers or computational tractability. Oftentimes, so-called *surrogate loss functions* are chosen for computational considerations [Nguyen et al., 2005]. In classification, for instance, we want to optimize the *zero-one loss* - a penalty of one for a misclassification and no penalty for correct classification. However, this leads to an NP-hard combinatorial problem. One may therefore resort to optimizing a convex upper bound, a reasonably close approximation of the zero-one loss, maintaining *convexity*, which in most cases opens the door to efficiently solving the resulting optimization problem. Similarly, different regularizers have different effects. For instance, choosing the L_1-norm (i.e. $\| \mathbf{w} \|_1 = \sum_i | \mathbf{w}_i |$) that is used in Lasso [Tibshirani, 1996], 1-norm SVM [Zhu et al., 2004] or Boosting [Rosset et al., 2004] induces sparsity in the parameter vector $\mathbf{w}$

of the linear model and may therefore be used for *feature selection*. Other regularizers, such as the L_2-norm used, for example, in support vector machine [Boser et al., 1992], ridge regression [Hoerl, 1962], regularized RBF networks [Poggio and Girosi, 1990] and gaussian processes [MacKay, 1992] have the interesting property of giving rise to *kernels* as we will see in the next paragraph. A table of common loss functions is given in

	Loss $l(f, y)$
Hinge [Gentile and Warmuth, 1998]	$\max(0, -yf)$
Squared Hinge [Keerthi and DeCoste, 2005]	$\frac{1}{2}\max(0, -yf)^2$
Soft Margin [Bennett and Mangasarian, 1992]	$\max(0, 1 - yf)$
Squared Soft Margin [Chapelle, 2007]	$\frac{1}{2}\max(0, 1 - yf)^2$
Exponential [Spiegelhalter et al., 1993]	$\exp(-yf)$
Logistic [Collins et al., 2002]	$\log(1 + \exp(-yf))$
Novelty [Schölkopf et al., 2001b]	$\max(0, 1 - f)$
Least mean squares [Williams, 1998]	$\frac{1}{2}(f - y)^2$
Least absolute deviation	$\lvert f - y\rvert$
Quantile regression [Koenker, 2005]	$\max(\tau(f - y), (1 - \tau)(y - f))$
ϵ-insensitive [Vapnik et al., 1997]	$\max(0, \lvert f - y\rvert - \epsilon)$
Huber's robust loss [Müller et al., 1997]	$\frac{1}{2}(f - y)^2$ if$\lvert f - y\rvert < 1$, else $\lvert f - y\rvert - \frac{1}{2}$
Poisson regression [Cressie, 1992]	$\exp(f) - yf$

Table 2.1: Table of common scalar loss functions, adapted from [Teo et al., 2007].

Table 2.1.

2.3.2 The Kernel Trick

As we have seen in Equation 2.13, optimization problems of the form given in Equation 2.14 can be expressed in a dual form. Intuitively, the kernel trick can be applied if feature vectors $\boldsymbol{x}_i$ only occur within an inner product with another feature vector $\boldsymbol{x}_j$ (i.e. $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$). Then the inner product may be replaced with any positive definite kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \leftarrow \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ [Schölkopf et al., 1997]. Alternatively, input data may be pre-processed by kernel principle kernel analysis [Schölkopf et al., 1997]. The algorithm now operates in a Reproducing Kernel Hilbert Space [Aronszajn, 1950]. From a practical point of view, the use of kernels allows the incorporation of arbitrary objects such as strings or graphs into a learning algorithm (as long as we can define a positive semi-definite kernel function). Furthermore, computations are often more efficient as the dimensionality of the dual form does not exceed the number of data points. Many prominent machine learning methods are known to work with kernels and have solvers specialized to solve the dual form.

In this context, it is worthwhile to ask the question which conditions need apply to regularizer and loss in Equation 2.14 to give rise to a kernelizable algorithm. It has long been known [Kimeldorf and Wahba, 1971] that a combination of Euclidean norm $\|.\|_2$ and convex loss allows the use of kernels, while the sparsity-inducing L_1-norm as used

in the Lasso [Tibshirani, 1996] prohibits the use of kernels. Providing a cornerstone for the understanding of kernel methods, [Nashed and Wahba, 1974; Schölkopf et al., 2001a] gave the following result.

Theorem 2.3.2 (Representer theorem)**.** *If $\Omega = h(\|f\|)$ for some increasing function $f : \mathbb{R}_+ \to \bar{\mathbb{R}}$, then some minimizer of 2.14 must admin the form $f(.) = \sum_{i=1} \alpha_i \kappa(., \boldsymbol{x}_i)$ for some $\alpha \in \mathbb{R}^n$. If h is strictly increasing, all minimizers admit this form.*

The representer theorem stated sufficient conditions, while the question which conditions are necessary were addressed in a recent line of work. Building upon the work of Argyriou et al. [2009], Yu et al. [2013] prove that the representer theorem holds *if and only if* the regularizer Ω is a *weakly* increasing function of the Hilbert norm, i.e.

$$\forall f, g \in H, \|g\| > \|f\| \to \Omega(g) \leq \Omega(f),$$

giving necessary and sufficient conditions.

2.3.3 Support Vector Machines

A learning machine of central importance to this thesis is the support vector machine (SVM), which has enjoyed great popularity during the last two decades. While the subject began in the 1970s with the work of Vapnik [1979], it has taken the machine learning community's center of attention from neural networks in the mid 90's [Boser et al., 1992; Cortes and Vapnik, 1995; Müller et al., 2001; Schölkopf and Smola, 2002; Schölkopf et al., 1997; Vapnik, 1995] (although neural networks have enjoyed a comeback [Hinton et al., 2006] in the form of *deep learning* [Montavon et al., 2012]). It brings together many of the concepts introduced above: The SVM is a convex regularized risk minimization problem, featuring L_2 regularization and the hinge-loss (see Table 2.1). As we have seen in Section 2.3.2, with this kind of regularizer, the representer theorem holds and thus allows the use of kernels. Duality plays an important role in SVMs, as a large body of research was concerned with the development of efficient SVM solvers that work in the dual (see Section 2.2.2). We briefly state the primal and dual form of the SVM, but omit the detailed derivation as we discuss a closely related learning machine in depth in Section 3.1. Furthermore, we refer the reader to Section 3.1.2 for a discussion of common solvers, where we review common strategies as a precursor to deriving a number of extensions.

Definition 2.3.3 (Primal SVM)**.**

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \\ \text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) + \xi_i - 1 \geq 0 \ \forall i \in [1, n] \\ \xi_i \geq 0 \ \forall i \in [1, n] \end{aligned}$$

Definition 2.3.4 (Dual SVM)**.**

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$
$$\text{s.t. } \alpha^T\mathbf{y} = 0, \qquad 0 \leq \alpha_i \leq C \;\forall i \in \{1,n\}.$$

Note that, in the above formulations, parameter C controls the trade-off between the regularizer and loss term, as explained in Section 2.3.1. The combination of these properties was responsible for the SVM's great success in practical applications, as it allowed efficient training due to its convexity, good generalization performance due to its *built-in* capacity control (i.e. regularization and large-margin approach) and flexibility by being able to incorporate various types of data due to its use of kernels.

2.3.4 Structured Output Learning

Structured output learning is a generalization of classification and regression, where the target domain $\mathcal{Y}$ is a structure, such as a graph, a sequence or an image. For instance, consider the target domain when trying to predict a sequence of length n over alphabet Σ. If we, as in the case of classification, considered each possible sequence to be a class, then we would be facing $|\Sigma|^n$ classes. Due to the exponential nature of the output space, the idea of structured output learning is to exploit the *structure* of the output space.

Formally, structured prediction is given by

$$Y = f(X) = \operatorname*{argmax}_{\hat{Y}} g(X,\hat{Y},\mathbf{w}),$$

where $g : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is a *compatibility* function between inputs and outputs which is parameterized by weight vector $\mathbf{w} \in \mathbb{R}^d$. The prediction is the $\hat{Y}$ that maximizes compatibility with X. An interesting observation in this context is that this general definition of g is an increase in flexibility compared to the probabilistic counterpart, Conditional Random Fields [Lafferty et al., 2001], which restrict g to a conditional probability $g(x,y,w) = \log p(y|x,w)$ [Nowozin and Lampert, 2011]. To formulate a structured output problem, the following components need to be considered:

- compatibility function g that exploits the structure of $\mathcal{Y}$
- computation of $\operatorname{argmax}_{\hat{Y}}$
- inference of $\mathbf{w}$ that parametrizes g

We refer the reader to the excellent monograph by Nowozin and Lampert [2011] for an in-depth discussion of each of the above points. For the discussion of our transfer learning extensions in later sections it is, however, necessary to elaborate on the inference

aspect. The inference task in structured output learning consists of learning the *best compatibility* function by tuning its parameter vector $\mathbf{w}$. The structured support vector machine extends the *large-margin* approach of the standard SVM to the structured output case [Altun et al., 2003; Tsochantaridis et al., 2005].

Definition 2.3.5 (Structured Support Vector Machine)**.** *Let $g(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \langle \mathbf{w}, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$ be a compatibility function parameterized by $\mathbf{w} \in \mathbb{R}^D$ and C the regularization parameter. Further, let $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be a loss function, where $\Delta(\boldsymbol{y}, \boldsymbol{y}')$ quantifies the cost associated with predicting $\boldsymbol{y}'$ when the true label is $\boldsymbol{y}$.*

$$\max_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} l(\boldsymbol{x}_i, \boldsymbol{y}_i, \mathbf{w}),$$

where

$$l(\boldsymbol{x}_i, \boldsymbol{y}_i, \mathbf{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - g(\boldsymbol{x}_i, \boldsymbol{y}_i, \mathbf{w}) + g(\boldsymbol{x}_i, \boldsymbol{y}, \mathbf{w}).$$

The loss defined above is a convex upper bound of the Δ-loss (as it is the maximum over many functions affine in $\mathbf{w}$ [Nowozin and Lampert, 2011]). We review existing solvers for the S-SVM in Section 3.2, where we also develop an extension for transfer learning with S-SVMs.

2.3.5 Cross-validation

Cross-validation [Golub et al., 1979; Stone, 1974] is often the method of choice for out-of-sample performance evaluation and model selection. In k-fold cross-validation, data sets are first partitioned in k disjoint subsets of roughly equal size. In the ith cross-validation fold, the ith subset is used to evaluate generalization performance, while the remaining $k-1$ are used for training. The averaged performance of the k folds is used to estimate generalization performance, with slightly pessimistic bias [Cawley and Talbot, 2010]. As pointed out by Cawley and Talbot [2010], there is a risk of *overfitting in model selection* when choosing hyper-parameters via cross-validation. For this reason, a rigorous evaluation procedure such as nested cross-validation [Stone, 1974] is needed to obtain an unbiased estimate of generalization performance, when performing model selection.

2.4 Learning from sequences

As we have seen in the section on SVMs, the use of kernels opens the door to dealing with arbitrary objects within the learning machine as long as we can define, or rather engineer, a similarity measure or kernel function that fulfills certain properties [Zien et al., 2000]. Dealing with string data is of central importance in the context of computational biology, as biological sequences such as DNA, RNA, protein, bodies of medical text (e.g. doctor's notes or patient medical records) are ubiquitous. String kernels

provide a convenient way of achieving this, in particular we will focus on two types of string kernels: 1) the spectrum kernel [Leslie et al., 2002] which defines similarity based on how many substrings or k-mers two strings have in common irrespective of their position and 2) the weighted degree string kernel [Rätsch et al., 2007; Sonnenburg et al., 2007a], which additionally takes position into account.

2.4.1 String Kernels

Spectrum Kernel Given two strings x, z over an alphabet Σ, the spectrum kernel [Leslie et al., 2002] is defined as:

$$k_l(x, z) = \langle \phi_l(x), \phi_l(z) \rangle, \tag{2.15}$$

where $\phi_l(x) = \sum_{i=1}^{|x|-l+1} I(x_{[i:i+l]} = s)$ is a feature map that maps a sequence to a $|\Sigma|^l$-dimensional feature space, whereas each dimension corresponds to the count of the associated substring s or length l in sequence x. A key insight is that this kernel may also be written without depending on the explicit definition of feature map ϕ, allowing for efficient computation, as:

$$k_l(x, z) = \sum_{i=1}^{|x|-l+1} \sum_{j=1}^{|z|-l+1} I(x_{[i:i+l]} = z_{[i:i+l]}) \tag{2.16}$$

Weighted Degree Kernel (WDK) Oftentimes, not only the presence of certain substrings is relevant, but also their exact position. Rätsch et al. [2007] have extended the idea of the Spectrum kernel to *localized signals* by counting the number of substrings that two strings of fixed length L have in common at the same (or similar) position. Formally, it is defined as

$$k_l(x, z) = \sum_{d=1}^{l} \beta_d \sum_{i=1}^{L-d+1} I(x_{[i:i+d]} = z_{[i:i+d]}), \tag{2.17}$$

where $\beta_d = 2\frac{l-d+1}{l^2+l}$ defines the weight for each substring length d. We can express the WD-Kernel in terms of feature map ϕ_d^{spec} as

$$\phi_d^{\text{wd}}(x) = (\phi_d^{\text{spec}}(z_{[1:d]}, \ldots, \phi_d^{\text{spec}}(z_{[L-d+1:L]}). \tag{2.18}$$

In [Toussaint et al., 2010], we extended the Spectrum kernel and the Weighted Degree kernel to be better suited for amino acid sequences, where it is important to account for the fact that some amino acids are more similar to each other than others due to their physio-chemical properties.

2.4.2 Sequences and the Hashing Trick

Recently, hashing was proposed as a means of efficiently dealing with very high-dimensional feature spaces [Shi et al., 2009]. Based on these ideas, Sonnenburg and Franc [2010] propose a hashing based strategy that allows the use of high dimensional feature spaces associated with the above string kernels efficiently in linear SVM solvers. They exploit the fact that linear SVM algorithms may be expressed in terms of two operations on feature and parameter vectors:

- Dot product between feature vector and vector: $r = \langle w, x \rangle$
- Addition of vector to feature vector: $w = w + \alpha x$

An efficient implementation requires the following two ingredients: On-demand feature computation and efficient representation of parameter vector $\mathbf{w}$.

On-demand features Sonnenburg and Franc [2010] proposed to compute the *non-zero elements* $\phi_{\neq 0}$ of features $\boldsymbol{x} = \phi(\boldsymbol{z})$ *on-demand* rather than in a preprocessing step. We assume that $\phi_{\neq 0}$ is indexed by index set $\mathcal{J}$ and require that the non-zero elements can be efficiently looked up. Further, for this to be efficient, the individual features have to be quickly computable and the number of non-zero elements must be small. For example, we can efficiently compute the non-zero elements of ϕ_d^{spec}, by sliding a window over the input sequence and keeping track of the observed k-mers.

Efficient representation of $\mathbf{w}$ Depending on the dimensionality of $\mathbf{w}$, Sonnenburg and Franc [2010] propose to either store $\mathbf{w}$ in a dense representation or use hashing. For the latter, the number of bits is fixed to γ and a hashing function $h : \mathcal{J} \to 1, \ldots, 2^\gamma$ is used to map from the index set $\mathcal{J}$ to the hash table. Clearly, this may come at the cost of information loss if several indices map to the same bin in the hash table. The implications of this have been theoretically and empirically analyzed by Shi et al. [2009].

For the *Spectrum kernel*, Sonnenburg and Franc [2010] recommend a dense representation of $\mathbf{w}$ (for reasonable degree) and we have already seen that an efficient on-demand feature computation ϕ_d^{spec} is easily implemented. For the *Weighted Degree kernel*, things look a bit different due to the sheer dimensionality of the associated feature space as defined in Equation 2.18. Thus, Sonnenburg and Franc [2010] propose to use multiple hash tables, one for each degree and position that are implemented efficiently using incremental or rolling hashes,

$$h(x_{[1:k+1]}, \sigma) = h(x_{[1:k+1]}, h(x_{[1:k]}, \ldots, h(x_{[1]}, \sigma))),$$

where σ is an initial seed. Putting it all together enables the use of the Spectrum- and the Weighted Degree kernel in linear SVMs.

2.4.3 Example Application: shRNA Prediction

We give an example of a successful application of string kernels for a pressing biological problem from cancer biology. In cancer cells, the expression of certain genes is oftentimes substantially up-regulated. These genes, also referred to as oncogenes, are typically associated with functions that enable tumor growth, such as cell growth or angiogenesis, the growth of blood vessels. It is the hope of recent medical research to come up with treatments that specifically target genes that discriminate tumors from normal cells to hamper or halt tumor growth. One possible strategy is to target the mRNA of these genes with short hairpin RNAs, called shRNAs, that are complementary to the sequence of the target. Complementary RNA molecules will form a double strand, which is in turn degraded by the cell. However, not all shRNA bind their targets equally well. Binding efficacy depends on the many molecular processing steps involved, which (among other factors) are dependent on the respective sequence composition. In this project, we propose splashRNA, a string-kernel based machine learning system that learns to predict the binding efficacy of shRNAs, a tool that could be of great value for designing targeted cancer therapies. For training, we use data from massively parallel sensor arrays that quantify whether a given short RNA molecule is effective in binding its target.

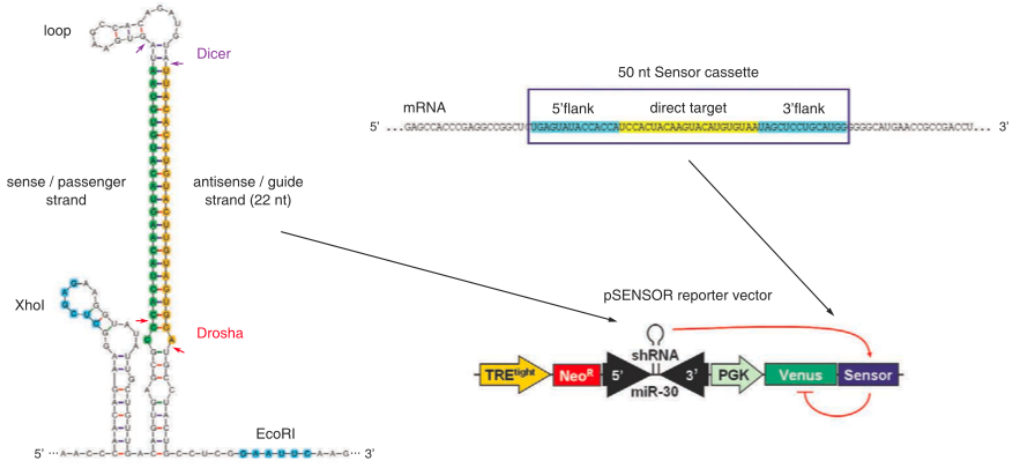


Figure 2.2: Depiction of the reporter assay used to generate training data. Figure taken from [Fellmann et al., 2011], Copyright (2011), with permission from Elsevier. See Section A in the Appendix for image licenses.

Data Generation The sensor assay screens almost 20,000 shRNAs against a set of 9 target genes in a massively parallel fashion. A population of cells are transduced with a pool of GFP-reporter vectors, each containing an shRNA as well as its cognate target sequence from the target mRNA, at a single copy per cell. By screening cells at progressively more stringent thresholds the assay eventually retains only potent

shRNAs by the final screen.

Method We learned a combination of two support vector machine (SVM) classifiers trained on an early and later screens. The central piece of our method was a carefully engineered string kernel. For this, we combined two spectrum kernels with the weighted degree string kernel. From biological experiments [Fellmann et al., 2011] we were aware that GC-composition at the head and the tail of the shRNA molecules play a role in the efficacy of the processing machinery. Therefore, we used two spectrum kernels of degree $k = 8$ to capture the GC-composition (among other things) at the beginning and end of the molecule. Using two spectrum kernels instead of one allows to capture the fact that the GC composition of highly effective shRNAs is different between the head and tail. Furthermore, it is well established [Fellmann et al., 2011] that certain patterns at particular positions are required for the shRNA to be effective. This kind of position-dependent feature may be perfectly captured using the weighted degree string kernel of degree $k = 4$, which we therefore use on the central segment of the shRNA molecule.

Results We compared our predictions against DSIR [Vert et al., 2006], a popular siRNA classifier and achieved significant improvement in accuracy on held-out assay and knockdown experiments. Our method achieved an area under the ROC curve

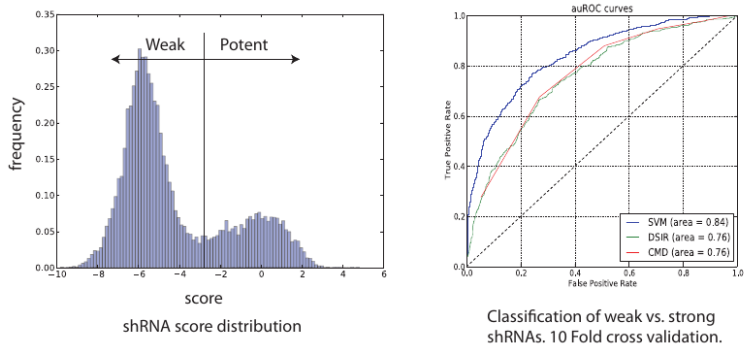


Figure 2.3: Classification of weak vs. strong shRNAs. 10 Fold cross validation.

(AUC) of 0.84, compared to 0.76 for DSIR, and the "rules of thumb" (CMD) from Fellmann et al. [2011] that are commonly used by biologists in practice, in 10- fold cross-validation. Furthermore, we gained detailed insight into the features that make a potent shRNA by using POIMs [Sonnenburg et al., 2008], an algorithm for visualizing sequence information contained in k-mer based SVM classifiers. Fellmann's features [Fellmann et al., 2011] define important position and k-mer combinations (1st positions at 1) 1U/A, 10A/U, 13-14AU, and 20G. We observed and extended them to include new combinations such as 1-2UU/UA, 19-20 no AA/AU. We further analyzed the rules that govern shRNAs that passed the biological processing steps, Drosha and Dicer enzymes,

and identified features that are specific to these processing steps. Taken together, these results demonstrate a greatly successful application of string kernels in conjunction with SVMs, not only pushing the limits in an application highly relevant to cancer research, but also gathering novel insight and deeper understanding of the biological mechanisms.

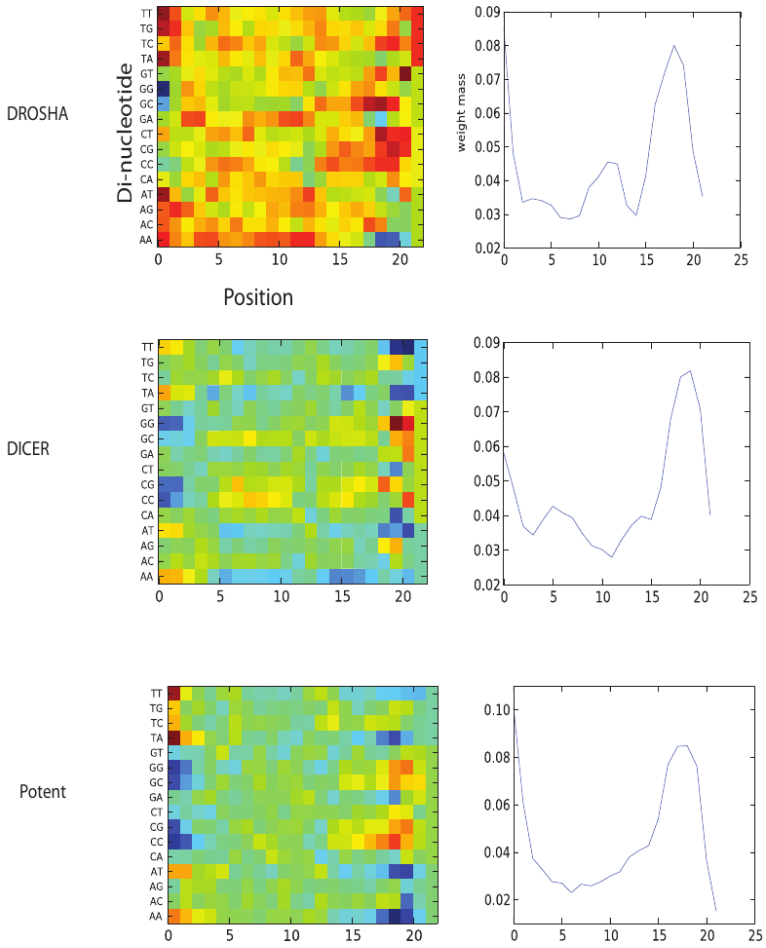


Figure 2.4: Interpretation of SVM classifier using POIMs [Sonnenburg et al., 2008]. Here, we use POIMs to visualize which positions (and di-nucleotides at these positions) in the RNA molecule of length 22 are important for classification. The different rows corresponds to different processing steps in the expression of the shRNA. *Drosha* and *Dicer* are necessary processing steps for creating a potent shRNA. Here, we generated labels indicating whether or not a shRNA passed each of these processing steps. The labels based on *Potent* are the ones we are ultimately interested in. One can observe a peak in importance at positions 1-2 and around position 19, which is consistent with what is known from literature, but provides a substantially more fine-grained set of rules.

3 Domain Adaptation

In this chapter, we propose an answer to Thrun [1996]'s question, if "learning the n-th thing [is] any easier than learning the first", which was posed in the mid-1990s in the context of neural networks. This setting can be understood as a special case of *transfer learning*, which generally refers to learning methods that transfer information from one or multiple source tasks to a target task with the aim of improving the prediction accuracy on the target task.

The main **contributions** in this chapter are the following:

- We propose a regularization-based strategy for adapting SVM to new domains
- Based on this, we derive an algorithm for hierarchical task relationships
- We derive efficient solvers for binary classification and structured output learning

Parts of this chapter are based on the following (equal contributions are indicated by *).

> G., Schweikert*, **C. Widmer***., B. Schölkopf, and G. Rätsch. An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis. In *Advances in Neural Information Processing Systems 21*, pages 1433-1440, 2008.
>
> **C. Widmer**, J. Leiva., Y. Altun, and G. Rätsch. Leveraging Sequence Classification by Taxonomy-based Multitask Learning. In *Research in Computational Molecular Biology*, pages 522-534, 2010.
>
> N. Görnitz*, **C. Widmer***., G. Zeller, S. Sonnenburg, and G. Rätsch. Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation. In *Advances in Neural Information Processing Systems 24*, pages 2690–2698, 2011.
>
> **C. Widmer**, G. Rätsch. Multitask Learning in Computational Biology. *Journal of Machine Learning Research Workshop and Conference Proceedings 27* Best paper award at *ICML 2011 Unsupervised and Transfer Learning Workshop*. pages 207-216., 2012.

We complement the work done by [Thrun, 1996] by exploring their research question in the context of regularization-based learning machines instead of neural networks. We propose extensions to generalize existing SVM solvers and derive a bundle methods based solver for the structured output learning case. Finally, we present an algorithm to handle hierarchical structures building upon the above regularization-based strategy and show its successful application to computational biology.

3.1 Source-regularized SVM

In this section, we will describe a SVM-based method for the *domain adaptation* setting, which is also known as *asymmetric transfer learning*. This asymmetry typically stems

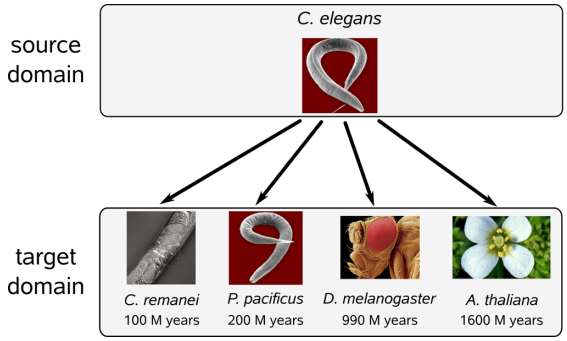


Figure 3.1: Example for a domain adaptation setting, where organisms correspond to tasks.

from the fact that training data is abundant for the *source* task while it is scarce for the *target* task. We approach this setting by augmenting the classic SVM formulation by an additional regularization term (see Section 1.5) that penalizes the deviation of the target parameter vector $\mathbf{w}$ from a constant parameter vector $\mathbf{w}_s$ that was previously obtained by training a model on the available source data.

$$\min_{\mathbf{w},b} \frac{1-B}{2} ||\mathbf{w}||^2 + \frac{B}{2} ||\mathbf{w} - \mathbf{w}_s||^2 + C \sum_{(\boldsymbol{x},y)\in S} \ell\left(\langle \boldsymbol{x}, \mathbf{w}\rangle + b, y\right), \tag{3.1}$$

where ℓ is the hinge loss $\ell(z, y) = \max\{1 - yz, 0\}$, $\mathbf{w}_s$ is the parameter vector from the source model and $B \in [0, 1]$ is a hyper-parameter that controls the trade-off between the two regularization terms. In practice, the optimal value for parameter B depends on the similarity between tasks, the number of available training examples for source and target tasks and should be determined by cross-validation. If we choose B to be zero, the source parameter $\mathbf{w}_s$ has no effect and the resulting solution will be identical to that of the standard SVM. Analogously, if $B = 1$ (and the influence of the loss term is down-weighted by a very small C) the solution will be forced to be very similar to $\mathbf{w}_s$. For most problems, the optimal B parameter is expected in between these two corner cases, as illustrated in Figure 3.2.

3.1.1 Dualization

One of the main contributions of this chapter is the extension of several established SVM solvers to handle the source-regularization term. As many prominent SVM solvers work in the dual (SVMlight, LibSVM) or depend on the primal-dual relationship (LibLinear), we first state the dual formulation of the optimization problem given in Equation 3.1.

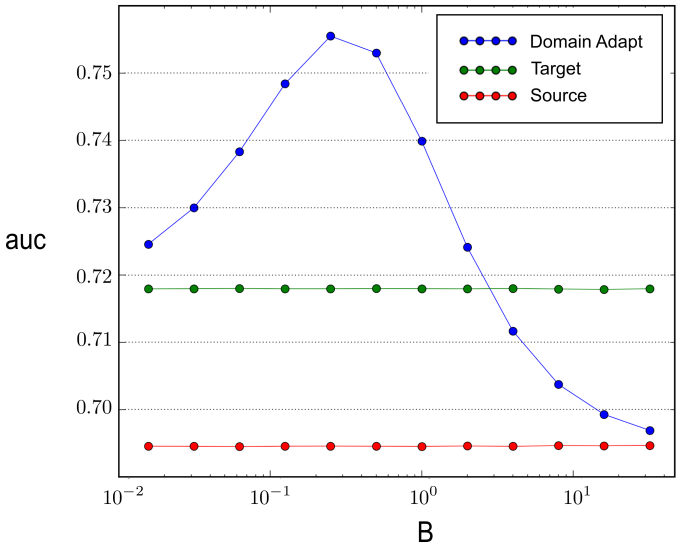


Figure 3.2: A illustration of the effect of regularization parameter B. Three curves are shown: For *Target*, a classifier is trained on examples from the target domain only. Similarly, for *Source* a classifier is trained on examples from the source domain only. In *Domain Adaptation*, the classifier trained on the source domain is adapted to the target domain for different values of B. All methods are evaluated on a separate test set from the target domain.

The dual of the above formulation is given by [Widmer et al., 2010b]:

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i\left(\underbrace{B\left(\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i,\mathbf{x}'_j)\right)}_{p_i} - 1\right)$$

$$\text{s.t. } \alpha^T\mathbf{y} = 0, \qquad 0 \le \alpha_i \le C \;\forall i \in \{1,n\},$$

where n and m are the number of training examples of the current task and the source task, respectively.

Proof Using the techniques introduced in Section 2.2, we derive the dual of the above *primal* problem using Lagrange multipliers. We start with a primal formulation constituting a minor reformulation (see Section B.1 in the appendix) of Equation 3.1.

$$\min_{\mathbf{w}_T,\xi} \frac{1}{2}\,\mathbf{w}_T^T\,\mathbf{w}_T + C\sum_{i=1}^{n}\xi_i - B\,\mathbf{w}_T^T\,\mathbf{w}_s$$

$$\text{s.t. } y_i(\mathbf{w}_T^T\,\mathbf{x}_i + b) + \xi_i - 1 \ge 0 \;\forall i \in [1,n] \tag{3.2}$$

$$\xi_i \ge 0 \;\forall i \in [1,n] \tag{3.3}$$

The Lagrangian of this is given by:

$$L(\mathbf{w}_T, \xi, \alpha) = \frac{1}{2}\,\mathbf{w}_T^T\,\mathbf{w}_T + C\sum_{i=1}^{n}\xi_i - B\,\mathbf{w}_T^T\,\mathbf{w}_s - \underbrace{\sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}_T^T\,\mathbf{x}_i + b) + \xi_i - 1)}_{(3.2)} - \underbrace{\sum_{i=1}^{n}\mu_i\xi_i}_{(3.3)}$$

We find the stationary points by setting the partial derivatives to zero:

$$\frac{\partial L}{\partial\,\mathbf{w}_T} = \mathbf{w}_T - B\,\mathbf{w}_s - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0 \tag{3.4}$$

$$\frac{\partial L}{\partial \xi} = \mathbf{C} - \alpha - \mu = 0 \tag{3.5}$$

$$\frac{\partial L}{\partial b} = -\alpha^T\mathbf{y} = 0 \tag{3.6}$$

Substituting Equation 3.4 into the Lagrangian leads to the following after some algebra:

$$\begin{aligned}
L =& -\frac{1}{2}(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i^T)(\sum_{j=1}^{n}\alpha_j y_j \mathbf{x}_j) - B(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i^T\,\mathbf{w}_s) + \sum_{i=1}^{n}\alpha_i + \xi^T(\mathbf{C} - \alpha - \mu) - \frac{1}{2}B^2\,\mathbf{w}_s^T\,\mathbf{w}_s \\
=& -\frac{1}{2}(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i^T)(\sum_{j=1}^{n}\alpha_j y_j \mathbf{x}_j) - B(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i^T\,\mathbf{w}_s) + \sum_{i=1}^{n}\alpha_i \\
=& -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{n}\alpha_i(B y_i \mathbf{x}_i^T\,\mathbf{w}_s - 1)
\end{aligned}$$

We arrive at the following dual optimization problem:

$$\begin{aligned}
\max_{\alpha} & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{n}\alpha_i(B y_i \mathbf{x}_i^T\,\mathbf{w}_s - 1) \\
& \text{s.t. } 0 \le \alpha_i \le C\ \forall i \in [1, n] \\
& \alpha^T\mathbf{y} = 0
\end{aligned}$$

Note that this formulation explicitly relies on the parameter vector $\mathbf{w}_s$ of the previously trained SVM. However, it is often desirable to obtain a kernelized formulation of a learning machine (see Section 2.3.2). The remaining steps address this issue. We use the representer theorem $\mathbf{w}_s = \sum_{j=1}^{m}\alpha'_j y'_j \mathbf{x}'_j$ to substitute $\mathbf{w}_s$, where $\boldsymbol{x}'_j$ denotes an example from the source dataset, y'_j the corresponding label, and α'_j the weight that

was assigned to example $\boldsymbol{x}'_j$ with respect to the solution of the previously trained SVM:

$$\begin{aligned}
L =& -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{n}\alpha_i(B(y_i\mathbf{x}_i^T(\sum_{j=1}^{m}\alpha'_j y'_j\mathbf{x}'_j)) - 1) \\
=& -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{n}\alpha_i(B(\sum_{j=1}^{m}\alpha'_j y_i y'_j\mathbf{x}_i^T\mathbf{x}'_j) - 1) \\
=& -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j \underbrace{y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)}_{Q_{ij}} - \sum_{i=1}^{n}\alpha_i\underbrace{((B(\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i,\mathbf{x}'_j)) - 1)}_{p_i}
\end{aligned}$$

We obtain the final optimization problem:

$$\begin{aligned}
\max_{\alpha} & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i((B(\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i,\mathbf{x}'_j)) - 1) \\
& \text{s.t. } 0 \le \alpha_i \le C\ \forall i \in [1,n] \\
& \alpha^T\mathbf{y} = 0
\end{aligned}$$

■

This directly corresponds to a QP in standard form (see Equation 2.3.4):

$$\begin{aligned}
\min_{\alpha}\ & \alpha^T\mathbf{Q}\alpha + \alpha^T\mathbf{p} \\
& \text{s.t. } 0 \le \alpha_i \le C\ \forall i \in [1,n] \\
& \alpha^T\mathbf{y} = 0
\end{aligned} \tag{3.7}$$

where $p_i = B(\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i,\mathbf{x}'_j)) - 1$. We have shown that the source-regularized SVM corresponds to a QP, which closely resembles the standard SVM. We exploit this insight in the derivation of efficient solvers for the source-regularized SVM by appropriately extending existing SVM solvers.

3.1.2 Extension of Existing SVM Solvers

As we have seen in the last section (e.g. Equation 3.7), the dual formulation of the source-regularized SVM closely resembles the standard dual SVM formulation. Therefore, it should be possible to re-use existing SVM solvers by adding minor extensions to handle a custom linear term $\boldsymbol{p}$.

Recall from Section 2.3.3 that SVM training involves solving a large quadratic program (QP), where one of the main challenges is that Q is a dense matrix that may be too large to be stored in memory. The dual SVM optimization problem is typically tackled by using custom solvers that exploit its special structure, due to this difficulty. The source of this special structure first and foremost comes from the linear inequality

constraints of the SVM problem (see Definition 2.3.4). Inequality constraints are considered to be *active* if they fulfill equality, *inactive* otherwise. As inactive constraints have no effect on the final solution, a common strategy for solving the SVM problem is to identify the active constraints, or active set, and subsequently solve a sub-problem based on the subset of variables within the active set. At each iteration, a working set selection step takes place in which variables for the active set are selected. The steps of working set selection and the solving of sub-problems are alternated until convergence [Bennett and Parrado-Hernández, 2006]. A possible downside of this strategy is that restricted active set methods often show slow convergence when close to the optimal solution [Bennett and Parrado-Hernández, 2006]. In the following, we will discuss the extension of three established SVM solvers, all of which are part of the SHOGUN Machine Learning Toolbox [Sonnenburg et al., 2010], to allow source-regularization.

SVMLight

SVMLight is a classic restricted active set method, where only a few variables are allowed to change at each iteration [Joachims, 1999]. In each iteration, the dual variables α_i are split into two sets; the set of fixed variables N and the set of free variables B, referred to as *working set*. Assume we have selected q variables for the working set while the remaining ones are fixed at their current value. Then, a sub-problem based on the free variables in the active set is solved. The algorithm will proceed by alternating active set updates with the solving of sub-problems until optimality conditions are met. Starting from Equation 3.7, sub-problems are decomposed assuming variables Q, α, $\boldsymbol{y}$ and p are properly arranged:

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_B \\ \boldsymbol{y}_N \end{bmatrix} \qquad \boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_B \\ \boldsymbol{p}_N \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BN} \\ \boldsymbol{Q}_{NB} & \boldsymbol{Q}_{NN} \end{bmatrix}$$

We have extended the standard SVMLight decomposition by carrying through the precomputed custom linear term $\boldsymbol{p}$, as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}_B} \quad & \frac{1}{2}\,\boldsymbol{\alpha}_B^T\,\boldsymbol{Q}_{BB}\,\boldsymbol{\alpha}_B + \frac{1}{2}\,\boldsymbol{\alpha}_N^T\,\boldsymbol{Q}_{NN}\,\boldsymbol{\alpha}_N + \boldsymbol{\alpha}_B^T(\boldsymbol{Q}_{BN}\,\boldsymbol{\alpha}_N - \boldsymbol{p}_B) - \boldsymbol{\alpha}_N^T\,\boldsymbol{p}_N \\ \text{s.t. } & 0 \leq \alpha_i \leq C \qquad \forall i \in [1, n] \\ & \boldsymbol{\alpha}_B^T\,\boldsymbol{y}_B + \boldsymbol{\alpha}_N^T\,\boldsymbol{y}_N = 0 \end{aligned} \tag{3.8}$$

Note that the original formulation of SVMLight is easily recovered by setting $\boldsymbol{p} = \mathbf{1}$. As the fixed variables in set N are constant, the terms $\frac{1}{2}\,\boldsymbol{\alpha}_N^T\,\boldsymbol{Q}_{NN}\,\boldsymbol{\alpha}_N$ and $\boldsymbol{\alpha}_N^T\,\boldsymbol{p}_N$ can be omitted without changing the solution of the optimization problem. To solve the sub-problem on the active set, SVMLight employs a LOQO-based QP solver [Vanderbei, 1994], which is capable of handling a custom linear term $\boldsymbol{p}$ and is therefore well suited for our purposes.

Relation to chunking A side-by-side comparison of Equation 3.7 and Equation 3.8 reveals an interesting insight about the relationship between the source-regularized

SVM and chunking. It becomes clear that these problems exhibit close resemblance, where the cross-term between active and inactive examples in Equation 3.8 $\boldsymbol{Q}_{BN}\,\boldsymbol{\alpha}_N$ corresponds to the cross-term between source and target examples $\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i, \mathbf{x}'_j)$ in Equation 3.7. The only difference are the constraints, which are only with respect to target variables in Equation 3.7, while both active and inactive variables are constrained in Equation 3.8, in the term $\boldsymbol{\alpha}_B^T\,\boldsymbol{y}_B + \boldsymbol{\alpha}_N^T\,\boldsymbol{y}_N = 0$. However, this equality constraint will always be fulfilled in the domain adaptation setting as the values themselves come from a SVM training procedure.

LibSVM

LibSVM [Chang and Lin, 2011] employs sequential minimal optimization (SMO) [Platt, 1999] - the extreme case of active set methods, where only two variables change per iteration. The update for these two variables can be efficiently computed using a simple update rule. As two variables have to be selected at each step, the selection method becomes a critical part of the algorithm in terms of convergence speed [Bennett and Parrado-Hernández, 2006]. Similar to SVMLight, the SVM problem given in Definition 2.3.4 is decomposed into an active set consisting of two variables (α_i and α_j) and an inactive set of the remaining variables, where constant terms are omitted.

$$
\begin{aligned}
&\min_{\alpha_i,\alpha_j} \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}\boldsymbol{Q}_{ii}\boldsymbol{Q}_{ij}\\ \boldsymbol{Q}_{ji}\boldsymbol{Q}_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + \begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\left(\boldsymbol{Q}_{BN}\,\boldsymbol{\alpha}_N - \boldsymbol{p}_B\right)\\
&\text{s.t.} \quad 0 \le \alpha_i, \alpha_j \le C\\
&\qquad y_i\alpha_i + y_j\alpha_j + \boldsymbol{\alpha}_N^T\,\boldsymbol{y}_N = 0
\end{aligned}
$$

Using the above problem decomposition, Chang and Lin [2011] derive a simple update rule to solve the two variable problem. We extend LibSVM to handle our source-regularizer by replacing the constant vector of ones $\boldsymbol{p} = \mathbf{1}$ of the standard SVM with the precomputed custom linear term $\boldsymbol{p}$.

LibLinear

Finally, we discuss how to extend LibLinear [Hsieh et al., 2008], a fast solver based on a similar idea as LibSVM. In contrast to LibSVM, LibLinear does not include a bias term b, (however a regularized pseudo-bias is realized by appending an additional constant dimension to data points x), which leads to the omission of the equality constraints in the dual problem [Steinwart et al., 2011]. To fulfill the equality constraint $\boldsymbol{\alpha}^T\,\boldsymbol{y} = 0$, after each update, at least two dual variables need to change. As we have seen in the previous section, in SMO the change δ_i of one variable α_i has to be complemented by the change δ_j in another variable α_j such that $\delta_j y_i + \delta_j y_j = 0$ and the equality constraint holds. The omission of this equality constraint therefore opens the door to a dual-coordinate descent strategy, where only a single dual variable α_i is updated at

each step. Starting from Equation 3.7 only one dual variable α_i is updated in one step, leading to:

$$\begin{aligned} \min_{\alpha_i} \quad & \frac{1}{2}\alpha_i \boldsymbol{Q}_{ii}\alpha_i + \alpha_i(\boldsymbol{Q}_{BN}\,\boldsymbol{\alpha}_N - \boldsymbol{p}_B) + const \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \end{aligned}$$

As $\boldsymbol{Q}$ may be too large to be stored in memory, LibLinear exploits the fact that in linear SVMs, one can explicitly represent the primal parameter vector $\mathbf{w} = \sum_{j=1}^{l} y_j \alpha_j \boldsymbol{x}_j$. Thus, we can reformulate the above in terms of $\mathbf{w}$, leading to:

$$\alpha_i(\boldsymbol{Q}_{BN}\,\boldsymbol{\alpha}_N - \boldsymbol{p}_B) = y_i\,\mathbf{w}^T\,\boldsymbol{x}_i - p_i + D_{ii}\alpha_i,$$

greatly reducing the time complexity of a dual single-variable update [Hsieh et al., 2008]. Again, we employ a precomputed custom linear term $\boldsymbol{p}$, which means the standard LibLinear formulation is recovered by setting $\boldsymbol{p} = \mathbf{1}$.

3.2 Extension to Structured Output Learning

In this section, we extend the principal idea of the source-regularized SVM (see Equation 3.1) to structured output (SO) learning. While the SVM case was solved by extending existing SVM solvers, we present a novel bundle methods-based solver to tackle the source-regularized SO-SVM. SO learning has been successfully applied in the analysis of images, natural language and sequences. Of these, sequences are of particular interest in computational biology for the analysis of DNA, RNA or proteins.

3.2.1 Structured Output Learning and Extension

Recall from Section 2.3.4 that, in contrast to binary classification, elements from the output space Υ (e.g., sequences, trees, or graphs) of structured output problems have an inherent structure which makes more sophisticated, problem-specific loss functions desirable. The discrepancy between the true label $\boldsymbol{y} \in \Upsilon$ and the predicted label $\hat{\boldsymbol{y}} \in \Upsilon$ is captured by a loss function $\Delta : \Upsilon \times \Upsilon \to \Re^+$. A widely used approach to predict $\hat{\boldsymbol{y}} \in \Upsilon$ is the use of a linearly parametrized model given an input vector $\boldsymbol{x} \in \mathcal{X}$ and a joint feature map $\Psi : \mathcal{X} \times \Upsilon \to \mathcal{H}$ that captures the dependencies between input and output [Tsochantaridis et al., 2005]:

$$\hat{\boldsymbol{y}}_{\mathbf{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\bar{\boldsymbol{y}} \in \Upsilon} \quad \langle \mathbf{w}, \Psi(\boldsymbol{x}, \bar{\boldsymbol{y}}) \rangle.$$

The most common approaches to estimate the model parameters $\mathbf{w}$ are based on structured output SVMs (e.g., Altun et al. [2003] and Tsochantaridis et al. [2005]) and conditional random fields (e.g., Lafferty et al. [2001]; see also Hazan and Urtasun [2010]). Here we follow the approach taken by Tsochantaridis et al. [2005] and Rätsch

and Sonnenburg [2006], where estimating the parameter vector $\mathbf{w}$ amounts to solving the following optimization problem

$$\min_{\mathbf{w}\in\mathcal{H}} \left\{ R(\mathbf{w}) + C\sum_{i=1}^{n} \ell(\max_{\bar{\boldsymbol{y}}\in\Upsilon} \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \bar{\boldsymbol{y}})\rangle + \Delta(\boldsymbol{y}_i, \bar{\boldsymbol{y}}) - \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i)\rangle) \right\}, \tag{3.9}$$

where $R(\mathbf{w})$ is a regularizer and ℓ is a loss function. For $\ell(a) = \max(0, a)$ and $R(\mathbf{w}) = \| \mathbf{w} \|_2^2$ we obtain the structured output support vector machine [Altun et al., 2003; Tsochantaridis et al., 2005] with margin rescaling and hinge-loss.

It turns out that we can combine the structured output formulation to incorporate a source-regularizer analogous to the binary SVM (as discussed in Section 3.1). We extend the regularizer $R(\mathbf{w})$ in (3.9) with a γ-parametrized convex combination of a multitask regularizer $\frac{1}{2}\,||\mathbf{w} - \mathbf{w}_s||_2^2$ with the original term (the binary SVM case is given in Equation 3.1). When $R(\mathbf{w}) = \frac{1}{2}\| \mathbf{w} \|_2^2$ and omitting constant terms, we arrive at $R_{p,\gamma}(\mathbf{w}) = \frac{1}{2}\| \mathbf{w} \|_2^2 - \gamma\langle \mathbf{w}, \mathbf{w}_s\rangle$, giving rise to the following optimization problem:

$$\min_{\mathbf{w}\in\mathcal{H}} \left\{ R_{p,\gamma}(\mathbf{w}) + C\sum_{i=1}^{n} \ell(\max_{\bar{\boldsymbol{y}}\in\Upsilon} \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \bar{\boldsymbol{y}})\rangle + \Delta(\boldsymbol{y}_i, \bar{\boldsymbol{y}}) - \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i)\rangle) \right\} \tag{3.10}$$

A major difficulty remains: solving the resulting optimization problems that now can become considerably larger than for the single-task case.

3.2.2 A Bundle Method for Efficient Optimization

A common approach to obtain a solution to (3.9) is to use so-called cutting-plane or column-generation methods. Here, one considers growing subsets of all possible structures and solves restricted optimization problems. An algorithm implementing a variant of this strategy based on primal optimization is given in Algorithm 1 (similar in Tsochantaridis et al. [2005]).

Cutting-plane and column generation techniques often converge slowly. Moreover, the size of the restricted optimization problems grows steadily and solving them becomes more expensive in each iteration. Simple gradient descent or second order methods can not be directly applied as alternatives because (3.10) is continuous but non-smooth. Our approach is instead based on bundle methods for regularized risk minimization as proposed in Smola et al. [2008] and Teo et al. [2010] and Do [2010]. In the case of SVMs, this further relates to the OCAS method introduced in Franc and Sonnenburg [2009]. In order to achieve fast convergence, we use a variant of these methods adapted to structured output learning that is suitable for source-regularized transfer learning.

We consider the objective function $J(\mathbf{w}) = R_{p,\gamma}(\mathbf{w}) + L(\mathbf{w})$, where

$$L(\mathbf{w}) := C\sum_{i=1}^{n} \ell(\max_{\bar{\boldsymbol{y}}\in\Upsilon} \{\langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \bar{\boldsymbol{y}})\rangle + \Delta(\boldsymbol{y}_i, \bar{\boldsymbol{y}})\} - \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i)\rangle)$$

and $R_{p,\gamma}(\mathbf{w})$ is as defined in Section 3.2.1. Direct optimization of J is very expensive as computing L involves computing the maximum over the output space. Hence, we

Algorithm 1 Column Generation Method for Structured Output Learning

$\mathbf{w}^{(1)} = \mathbf{w}_s$
$k = 1$ and $\Gamma_i = \emptyset \quad \forall i$
repeat
  for $i = 1, \ldots, n$ **do**
    $\boldsymbol{y}^* = \text{argmax}_{\boldsymbol{y} \in \Upsilon} \{\langle \mathbf{w}^{(k)}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}) \rangle + \Delta(\boldsymbol{y}_i, \boldsymbol{y})\}$
    if $\langle \mathbf{w}^{(k)}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}^*) \rangle + \Delta(\boldsymbol{y}_i, \boldsymbol{y}^*) > \max_{(\Psi,\Delta) \in \Gamma_i} \{\langle \mathbf{w}^{(k)}, \Psi \rangle + \Delta\}$ **then**
      $\Gamma_i = \Gamma_i \cup (\Psi(\boldsymbol{x}_i, \boldsymbol{y}^*), \Delta(\boldsymbol{y}_i, \boldsymbol{y}^*))$
    end if
  end for
  $\mathbf{w}^{(k)} = \underset{\mathbf{w} \in \mathcal{H}}{\text{argmin}} \quad \left\{ R_{p,\gamma}(\mathbf{w}) + C \sum_{i=1}^{n} \ell \left(\max_{(\Psi,\Delta) \in \Gamma_i} \{\langle \mathbf{w}, \Psi \rangle + \Delta\} - \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, y_i) \rangle \right) \right\}$
  $k = k + 1$
until no changes in $(\Psi_i, \Delta_i)\ \forall i$

propose to optimize an estimate of the empirical loss $\hat{L}\ (\mathbf{w})$, which can be computed efficiently. We define the estimated empirical loss $\hat{L}\ (\mathbf{w})$ as

$$\hat{L}(\mathbf{w}) := C \sum_{i=1}^{N} \ell \left(\max_{(\Psi,\Delta) \in \Gamma_i} \{\langle \mathbf{w}, \Psi \rangle + \Delta\} - \langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i) \rangle \right).$$

Accordingly, we define the estimated objective function as $\hat{J}(\mathbf{w}) = R_{p,\gamma}(\mathbf{w}) + \hat{L}(\mathbf{w})$. It is easy to verify that $J(\mathbf{w}) \geq \hat{J}(\mathbf{w})$. Γ_i is a set of pairs $(\Psi(\boldsymbol{x}_i, \boldsymbol{y}), \Delta(\boldsymbol{y}_i, \boldsymbol{y}))$ defined by a suitably chosen, growing subset of Υ, such that $\hat{L}(\mathbf{w}) \to L(\mathbf{w})$ (cf. Algorithm 2).

In general, bundle methods are extensions of cutting plane methods that use a prox-function to stabilize the solution of the approximated function. In the framework of regularized risk minimization, a natural prox-function is given by the regularizer. We apply this approach to the objective $\hat{J}(\mathbf{w})$ and solve

$$\min_{\mathbf{w}} R_{p,\gamma}(\mathbf{w}) + \max_{i \in I} \{\langle \boldsymbol{a}_i, \mathbf{w} \rangle + b_i\} \tag{3.11}$$

where the set of cutting planes $\boldsymbol{a}_i$, b_i lower bound $\hat{L}$. As proposed in Do [2010] and Teo et al. [2010], we use a set I of limited size. Moreover, we calculate an aggregation cutting plane $\bar{\boldsymbol{a}}$, $\bar{b}$ that lower bounds the estimated empirical loss $\hat{L}$. To be able to solve the primal optimization problem in (3.11) in the dual space as proposed by Do [2010] and Teo et al. [2010], we adopt an elegant strategy described in Do [2010] to obtain the aggregated cutting plane $(\bar{\boldsymbol{a}}', \bar{b}')$ using the dual solution $\boldsymbol{\alpha}$ of (3.11):

$$\bar{\boldsymbol{a}}' = \sum_{i \in I} \alpha_j \boldsymbol{a}_i \qquad \text{and} \qquad \bar{b}' = \sum_{i \in I} \alpha_i b_i. \tag{3.12}$$

The following two formulations reach the same minimum when optimized with respect to $\mathbf{w}$:

$$\min_{\mathbf{w} \in \mathcal{H}} \left\{ R_p(\mathbf{w}) + \max_{i \in I} \langle \boldsymbol{a}_i, \mathbf{w} \rangle + b_i \right\} = \min_{\mathbf{w} \in \mathcal{H}} \{R_p(\mathbf{w}) + \langle \bar{\boldsymbol{a}}', \mathbf{w} \rangle + \bar{b}'\}.$$

This new aggregated plane can be used as an additional cutting plane in the next iteration step. We therefore have a monotonically increasing lower bound on the estimated empirical loss and can remove previously generated cutting planes without compromising convergence (see Do [2010] for details).

Algorithm 2 Bundle Methods for Structured Output Algorithm [Görnitz* et al., 2011]

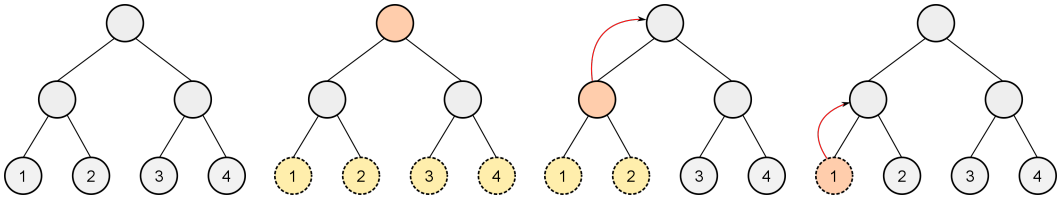
$S \geq 1$: maximal size of the bundle set
$\theta > 0$: linesearch trade-off (cf. Franc and Sonnenburg [2009] for details)
$\mathbf{w}^{(1)} = \mathbf{w}_p$
$k = 1$ and $\bar{\boldsymbol{a}} = \mathbf{0}, \bar{b} = 0, \Gamma_i = \emptyset \quad \forall i$
repeat
  for $i = 1, \ldots, n$ **do**
    $\boldsymbol{y}^* = \operatorname{argmax}_{\boldsymbol{y} \in \Upsilon} \{\langle \mathbf{w}^{(k)}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}) \rangle + \Delta(\boldsymbol{y}_i, \boldsymbol{y})\}$
    if $\ell \left(\max_{\boldsymbol{y} \in \Upsilon} \{\langle \mathbf{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}) \rangle + \Delta(\boldsymbol{y}_i, \boldsymbol{y})\} \right) > \ell \left(\max_{(\Psi, \Delta) \in \Gamma_i} \langle \mathbf{w}, \Psi \rangle + \Delta \right)$ **then**
      $\Gamma_i = \Gamma_i \cup (\Psi(\boldsymbol{x}_i, \boldsymbol{y}^*), \Delta(\boldsymbol{y}_i, \boldsymbol{y}^*))$
    end if
    Compute $\boldsymbol{a}_k \in \partial_{\mathbf{w}} \hat{L}(\mathbf{w}^{(k)})$
    Compute $b_k = \hat{L}(\mathbf{w}^{(k)}) - \langle \mathbf{w}^{(k)}, \boldsymbol{a}_k \rangle$
    $\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{H}} \left\{ R_{p,\gamma}(\mathbf{w}) + \max \left(\max_{(k-S)_+ < i \leq k} \{\langle \boldsymbol{a}_i, \mathbf{w} \rangle + b_i\}, \langle \bar{\boldsymbol{a}}, \mathbf{w} \rangle + \bar{b} \right) \right\}$
    Update $\bar{\boldsymbol{a}}$, $\bar{b}$ according to (3.12)
    $\eta^* = \operatorname{argmin}_{\eta \in \Re} \hat{J}(\mathbf{w}^* + \eta(\mathbf{w}^* - \mathbf{w}^{(k)}))$
    $\mathbf{w}^{(k+1)} = (1 - \theta)\, \mathbf{w}^* + \theta \eta^* (\mathbf{w}^* - \mathbf{w}^{(k)})$
    $k = k + 1$
  end for
until $L(\mathbf{w}^{(k)}) - \hat{L}(\mathbf{w}^{(k)}) \leq \epsilon$ and $J(\mathbf{w}^{(k)}) - J_k(\mathbf{w}^{(k)}) \leq \epsilon$

The algorithm is able to handle any (non-)smooth convex loss function ℓ, since only the subgradient needs to be computed. This can be done efficiently for the hinge-loss, squared hinge-loss, Huber-loss, and logistic-loss. The resulting optimization algorithm is outlined in Algorithm 2.

3.3 Taxonomy-based Transfer Learning

In the previous two sections of this chapter, we have developed a regularization-based strategy to adapt a trained learning machine to a new target task and showed how to solve it efficiently for binary classification and structured output learning. Based on this idea, we now develop an algorithm that is hand-tailored to the case of *hierarchical task structures*.

Assuming a tree structure of tasks is particularly relevant to computational biology where different tasks often correspond to different organisms, which are in turn related by a common evolutionary history that traces back to a single common ancestor. In this



(a) Given taxonomy (b) Root training (c) Inner training (d) Taxon training

Figure 3.3: Illustration of the Top-down training procedure, based on [Widmer et al., 2010b]. In this example, we consider four tasks related via the tree structure in 3.3(a). Each leaf is associated with a task t and its training sample $S_t = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. In 3.3(b), training begins by obtaining the predictor at the root node. Next, we move down one level and train a classifier at an inner node, as shown in 3.3(c). Here, the training error is computed with respect to $S_1 \cup S_2$, while the classifier *pulled* towards to the parent model $\mathbf{w}_p$ using the newly introduced regularization term, as indicated by the red arrow. Lastly, in 3.3(d), we obtain the final classifier of task 1 by only taking into account S_1 to compute the loss, while again regularizing against the parent predictor. The procedure is applied in a top-down manner to the remaining nodes until we have obtained a predictor for each leaf.

context, a hierarchical structure is referred to as *taxonomy*. We expect that the longer the common evolutionary history between two organisms, the more beneficial it is to share information between these organisms. In the given taxonomy, tasks correspond to leaves (i.e. taxa) and are related by its inner nodes. The basic idea of the training procedure is to mimic biological evolution by obtaining a more specialized classifier with each step from root to leaf.

This specialization is achieved by minimizing the training error with respect to training examples from the current subtree (i.e. the tasks below the current node), while similarity to the parent classifier (we assume every node except the root node has access to a parent parameter vector $\mathbf{w}_p$) is enforced through regularization. In this context, the *parent model* $\mathbf{w}_p$ corresponds to the *source model* from Section 3.1. An outline of the algorithm in pseudocode is given in Algorithm 3. The algorithm is executed in a top-down manner such that a model $\mathbf{w}$ is trained at each node n, whereas the loss L is measured with respect to the union of training data at the leaves below the current node, while the current parameter vector $\mathbf{w}$ is regularized with respect to its parent $\mathbf{w}_p$. When interpreting the organisms at the leaves in the taxonomy as current species, then interior nodes correspond to extinct species, meaning that there is no training data available at the inner nodes. To further clarify the algorithm, an illustration of the training procedure is given in Figure 3.3.

Taxonomically Constrained Model Selection

Model selection for transfer learning involving multiple tasks is particularly difficult; it requires hyper-parameter selection for several different, but related tasks in a dependent manner. For the described hierarchical learning procedure, each node n in the given taxonomy corresponds to solving an optimization problem that is subject to hyper-

Algorithm 3 Top-down training procedure

T: number of tasks
τ: taxonomy relating all T tasks, containing nodes $n = [1, \ldots, N]$
descendants(n) : function that returns set of descending nodes of n according to τ
children(n) function that returns set of children of n according to τ
parent(n) function that return parent of n according to τ
$\mathbf{X}_t \, \forall t$: training examples for task t
$\boldsymbol{y}_t \, \forall t$: training labels for task t
$B_n > 0$: regularization constant at each node n in τ
stack = *root*
while stack $\neq \emptyset$ **do**
  n = stack.pop()
  $S = \{t : n_t \in \text{descendants}(n) \wedge \text{children}(n_t) = \emptyset\}$
  $p = \text{parent}(n)$
  $\mathbf{w}_n, b_n = \text{argmin}_{\mathbf{w},b} \; \frac{1}{2} ||\mathbf{w}||^2 + \frac{B_n}{2} ||\mathbf{w} - \mathbf{w}_p||^2 + C \sum_{t \in S} \sum_{i=1}^{N_t} \ell \left(\langle \boldsymbol{x}_i, \mathbf{w} \rangle + b, y_i \right)$
  stack = stack $\bigcup$ children(n)
end while

parameters γ_n and C_n (except for the root node, where only C_n is relevant). Hence, the direct optimization of all combinations of dependent hyper-parameters in model selection is not feasible in many cases. We therefore propose to perform a *local* model selection and optimize the current C_n and γ_n at each node n independently from top to bottom. This can be interpreted as using the taxonomy for *reducing the parameter search space*. To clarify this point, assume a perfect binary tree for n tasks. The length of the path from root to leaf is $log_2(n)$. The parameters along one path are dependent, e.g. the values chosen at the root will influence the optimal choice further down the tree. Given k candidate values for parameter γ_n, jointly optimizing all dependent parameters along one path corresponds to optimizing over a grid of $k^{log_2(n)}$ in contrast to $k \cdot log_2(n)$ when using our proposed *local* strategy.

3.4 Experiments

In the following, we report results from three experiments. First, we consider synthetic sequence data, which were created from a simple generative model (0-th order Markov chain, also known as Position Specific Scoring Matrix). Models for different tasks were *evolved* by applying subsequent mutations to model parameters according to a pre-defined binary tree.

In a second experiment, we consider the problem of splice-site recognition for multiple organisms. We labeled sequences of acceptor splice sites from 15 different eukaryotic genomes, which were similarly used in Rätsch et al. [2005] and Schweikert* et al. [2009] (see Figure 3.6 for the taxonomic relation between the organisms).

Third, we show an application of the structured output extension of our method to

prokaryotic gene finding. Here, the task is to identify genomic regions encoding genes (see Section 2.1), which we cast as a labeled sequence learning problem.

Experiments were performed for the presented MTL method and the following two baseline methods. In *Union*, all data are combined into one data set $S = \cup_{t=1}^{T} S_t$, and a single global model is obtained, which is used to predict on all domains. Furthermore, we consider the baseline method *Plain*, in which an individual SVM is trained on the data S_i of each domain separately, not taking into account any information from the other domains. For each method, the regularization parameter C was chosen by cross-validation. The performance was measured on a separate test sets, which are large enough to obtain reliable estimates of the predictors' performances.

3.4.1 Synthetic Dataset

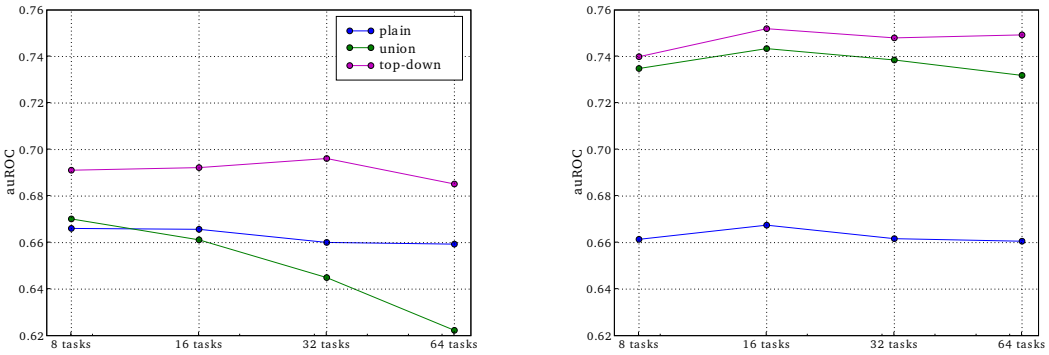
For this experiment, we start from a binary tree structure and emulate biological evolution by applying mutations to the parameter vector of a generative model along the edges of the given tree. To simulate biological sequences, we choose a simple 0th-order Markov chain parametrized by Θ. For now, we are interested in binary classification, so we assume a generative model with a distribution $P(\Theta_{\text{pos}})$ for the positive class and a distribution for the negative class $P(\Theta_{\text{neg}})$. Starting at the root node, we apply *mutations* of controlled magnitude to the parameters of the generative model such that the Bayes error remains constant. To control the distance between distributions, we use a symmetrized *Kullbach-Leibler* divergence, which which we define as $\text{KLS}(P_1, P_2) = \frac{\text{KL}(P_1,P_2)+\text{KL}(P_2,P_1)}{2}$, based on the standard KL divergence. We generate a pair of parameters $(\Theta_{\text{pos}}, \Theta_{\text{neg}})$ for a node, given the parameters of its parent node $(\tilde{\Theta}_{\text{pos}}, \tilde{\Theta}_{\text{neg}})$ such that

$$\text{KLS}(P(\Theta_{\text{pos}}), P(\Theta_{\text{neg}})) \approx b,$$

where b is a constant controlling the Bayes error and

$$\text{KLS}(P(\Theta_{\text{pos}}), P(\tilde{\Theta}_{\text{pos}})) \approx \text{KLS}(P(\Theta_{\text{neg}}), P(\tilde{\Theta}_{\text{neg}})) \approx m,$$

where m is the *mutation rate*. The models at the leaves of the binary tree are the ones that were used to sample sequences for positive and negative classes for each task. Balanced, equally sized training data sets, 100 examples each, were sampled for each of the leaves. As test set, an additional $5,000$ examples were sampled for each task. More information, including the source code of the outlined generation procedure, is provided in Section C.1 in the appendix. We used the area under the ROC curve to evaluate the prediction performance (an evaluation using the area under the precision recall curve yields similar results). We consider two different settings of generating the toy data sets: one with *high mutation rate* (larger differences between parent and child tasks in the hierarchy) and one with *low mutation rate* (i.e. tasks are more closely related). For this study, we analyze how the different methods perform in these two settings with respect to the number of tasks (8, 16, 32 and 64 tasks). Results are shown in Figure 3.4. We can observe that the two baseline methods perform very differently: While the *Plain* method performs essentially indifferent for different numbers of tasks and mutation



(a) Toy data: High mutation rate

(b) Toy data: Low mutation rate

Figure 3.4: Results for the five considered methods on the toy data sets with high and low mutation rate. Shown are the average areas under the ROC curves for different numbers of tasks that have been generated from full binary trees [Widmer et al., 2010b].

rates, the *Union* method performs very well when the mutation rate is low, but quite poorly for high mutation rates. Moreover, the performance of *Union* degrades for an increasing number and, hence, diversity of tasks. This is particularly pronounced for high mutation rates. The proposed hierarchical method clearly outperforms the two baseline methods, indicating that it pays off to take the additional data and the relation between the tasks into account.

3.4.2 Splice-sites from 15 Organisms

Background The recognition of splice-sites is a highly relevant problem in genome biology as the knowledge about the positions of splice-sites in the genome are of central importance for tasks such as RNA-seq read alignment [Jean et al., 2010], gene-finding [Schweikert et al., 2009; Stanke and Morgenstern, 2005] and quantification of expressed transcripts [Behr et al., 2013; Bohnert et al., 2009]. The relevant biological background is summarized in Section 2.1. Recall that all splice-sites in the genome exhibit a so-called *consensus sequence*, a sequence of two nucleotides (GU or AG, for donor or acceptor, respectively) that is always present. However, not all positions

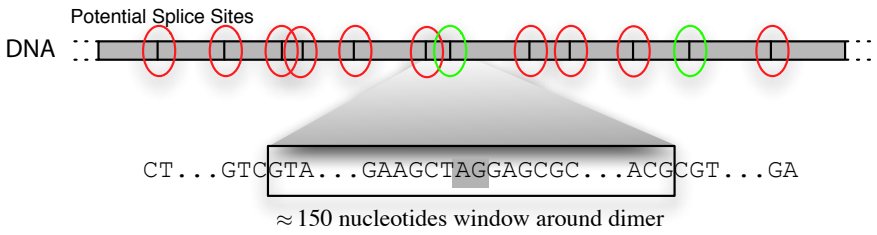


Figure 3.5: Illustration of splice-site recognition. Taken from presentation slides of [Rätsch et al., 2007].

in the genome where this consensus sequence occurs are splice-sites. These so-called *decoy-sites* will serve as negative examples in a binary classification problem. When predicting splice-sites from a genomic sequence, we typically cut out windows around the consensus sequence and incorporate these sequences into the learning framework using string kernels (see Section 2.4.1).

Experimental Setup In this experiment, we consider splice-site data from 15 organisms, which we treat as different tasks. We assume that we are given the evolutionary history of these 15 organisms in the form of a tree structure as shown in Figure 3.6. For each task, we obtained $10,000$ training examples and an additional test set of $5,000$ examples. We normalized the data sets to have 100 negative examples per positive example. We report the area under the precision recall curve (auPRC), which is an appropriate measure for unbalanced classification problems (i.e. detection problems) [Rätsch et al., 2007]. Please see Section C.2 for further information on data generation and implementations of the compared methods.

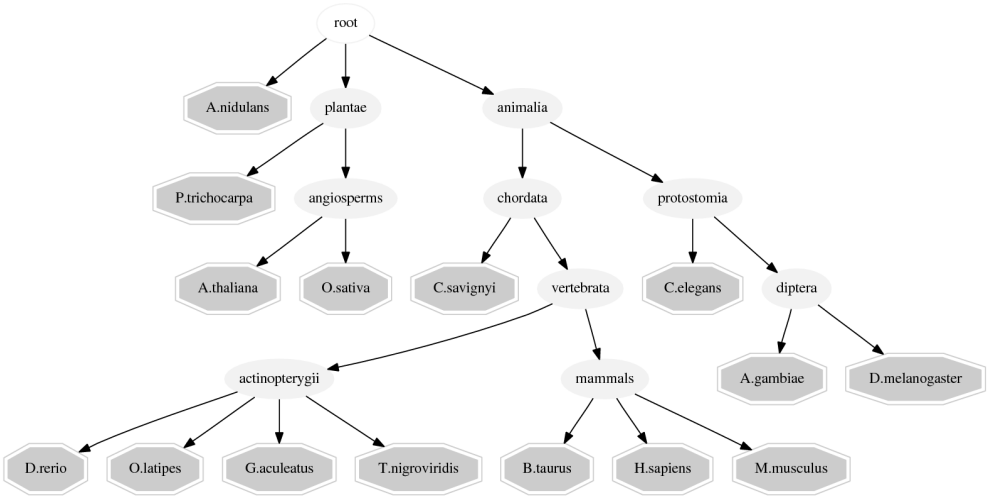


Figure 3.6: Taxonomy relating organisms in splicing experiment [Widmer et al., 2010b].

Results For the splice-site data set, examining the average performance across all organisms (the last column in Figure 3.7), we observe superior performance of the Top-Down method over the baseline methods. The *Plain* method is outperformed by the other methods, which emphasizes the importance of using data from related organisms.

Zooming in on the results for individual organisms, we observe the same trend: the *Top-Down* method outperforms the baselines in most cases and the *Plain* method yields the worst performance. The *Top-Down* method always performs better than the baseline methods, except for *A. nidulans*, which is the only fungal organism in our set. The gain from hierarchy is more pronounced in the lower levels of the hierarchy, e.g.,

for *A. thaliana*, *O. sativa*, *O. latipes*, and *D. rerio*, where data from closely related organisms are used to leverage the learning process. An exception to this behavior is seen on the mammals branch, where the performance gain from transfer learning gets smaller for *H. sapiens* and essentially diminishes for *M. musculus*. Our conjecture is that the hierarchy for this branch is not deep enough in order to represent closely related organisms. Including more similar organisms, therefore, extends the hierarchy to capture more evolutionary steps and can improve the performance gain for these organisms.

It is worthwhile to investigate the performance of the methods on *A. nidulans*, which is the child of the root node and the most distantly related to the other organisms. We observe that the *Union* method performs worst on this organism and the *Top-Down* method performs on the same level as the *Plain* method. Hence, the *Top-Down* method manages to improve the performance for closely related organisms, while causing (essentially) no performance loss on the distantly related ones.

Moreover, it is interesting to observe that *A. nidulans* and *C. elegans* are the only organisms/tasks, for which the *Union* method performs considerably worse than the *Plain* method. This hints at major differences between the recognition of splice-sites in these two organisms. This is less surprising for *A. nidulans* as it is for *C. elegans*. It appears worth investigating this property also in other nematode genomes to better understand this observation.

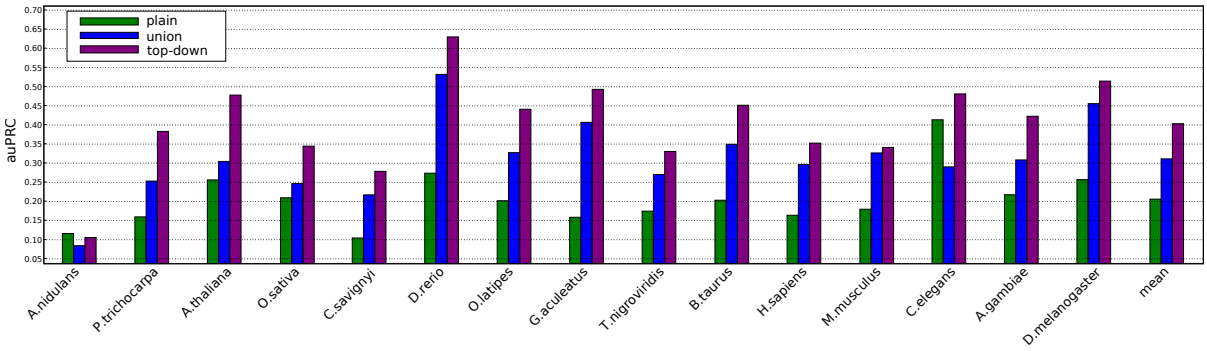


Figure 3.7: Results for the splice-site data sets from 15 eukaryotic genomes: Shown are auPRC performances of the 3 considered methods for each organism (two baseline methods: *Plain*, *Union*; and the proposed method: *Top-Down*). We can observe that the two *Top-Down* consistently outperform the baseline methods. In 14/15 cases, the hierarchy information lead to improved prediction results [Widmer et al., 2010b].

3.4.3 Gene Finding in Multiple Prokaryotic Genomes

Background Recall from Section 2.1 that genomic sequences can be represented by long strings of the four letters A, C, G, and T (genome sizes range from a few megabases to several gigabases). In prokaryotes (mostly bacteria and archaea), gene structures are comparably simple (cf. Figure 3.8A): the protein coding region starts by a start

codon (one out of three specific 3-mers in many prokaryotes) followed by a number of codon triplets (of three nucleotides each) and is terminated by a stop codon (one out of five specific 3-mers in many prokaryotes). Genic regions are first transcribed to RNA, subsequently the contained coding region is translated into a protein. Parts of the RNA that are not translated are called untranslated region (UTR). Genes are separated from one another by intergenic regions. The protein coding segment is depleted of stop codons making the computational problem of identifying coding regions relatively straight forward. Here, we consider the key problem in computational gene finding of predicting the coding regions for prokaryotes.

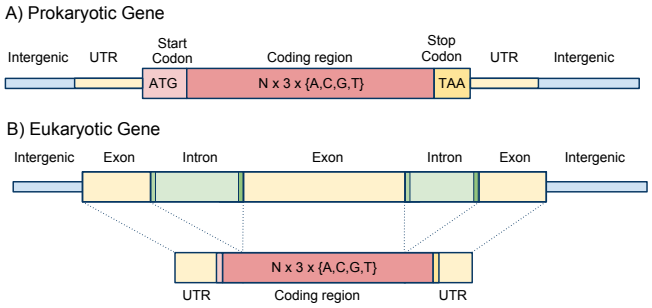


Figure 3.8: This figure Panel A shows the structure of a prokaryotic gene. The protein coding region is flanked by a start and a stop codon and contains a multiple of three nucleotides. UTR denotes the untranslated region. Panel B shows the structure of an eukaryotic gene. The transcribed region contains introns and exons. Introns are flanked by splice-sites and are removed from the mRNA. The remaining sequence contains the UTRs and coding region. Based on [Görnitz* et al., 2011].

The problem of identifying genes can be posed as a label sequence learning task, where one assigns a label (out of *intergenic, transcript start, untranslated region, coding start, coding exon, intron, coding stop, transcript stop*) to each position in the genome. The labels have to follow a grammar dictated by the biological processes of transcription and translation (see Figure 3.8) making it suitable to apply structured output learning techniques to identify genes. Because the biological processes and cellular machineries that recognize genes have evolved over time, genes of closely related species tend to exhibit similar sequence characteristics. Therefore, these problems are very well suited for the application of transfer learning: sharing information among species is expected to lead to more accurate gene predictions compared to approaching the problem for each species in isolation. Currently, the genomes of many prokaryotic and eukaryotic species are being sequenced. In spite of these efforts, often very little is known about the genes encoded and standard methods are typically used to infer them without systematically exploiting reliable information on related species.

In the following we will describe the problem of prokaryotic gene finding in several species and demonstrate that the *Top-Down* hierarchical structured output learning significantly improves gene prediction accuracy.

SO prediction method We modeled prokaryotic genes as a Markov chain on the nucleotide level. To account for the biological fact that genetic information is encoded in triplets, the model contains a 3-cycle of exon states; details are given in Figure 3.9:

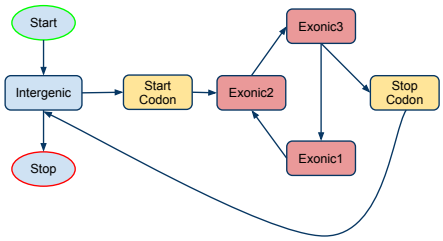


Figure 3.9: Simple state model for prokaryotic gene finding. A suitable model for prokaryotic gene prediction needs to consider 1) that a gene starts with a start codon (i.e. a certain triplet of nucleotides) 2) ends with a stop codon and 3) has a length divisible by 3. Properties 1) and 2) are enforced by allowing only transitions into and out of the exonic states on start and stop codons, respectively. Property 3) is enforced by only allowing transitions from exon state *Exonic3* to the stop codon state. Figure based on [Görnitz* et al., 2011].

Data generation We selected a subset of organisms with publicly available genomes to broadly cover the spectrum of prokaryotic organisms. In order to investigate whether transfer learning is beneficial even for relatively distant species, we selected representatives from two different domains: bacteria and archaea. The relationship between these organisms is captured by the taxonomy shown in Figure 3.10, which was created based on the information available on the NCBI website. More details are provided in in Section C.3 of the appendix. For each organism, we generated one training exam-

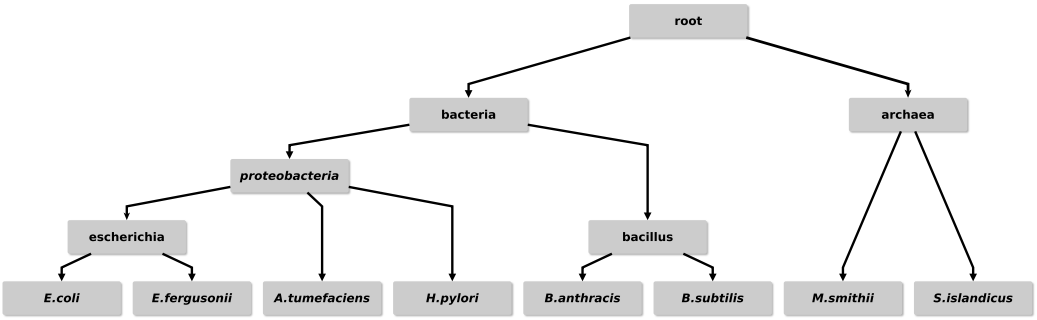


Figure 3.10: Taxonomy used for prokaryotic gene finding [Görnitz* et al., 2011].

ple per annotated gene. The genomic sequences were cut between neighboring genes (splitting intergenic regions equally), such that a minimum distance of 6 nucleotides between genes was maintained. Features for SO learning were derived from the nucleotide sequence by transcoding it to a numerical representation of triplets. This resulted in binary vectors of size $4^3 = 64$ with exactly one non-zero entry. We sub-sampled from the complete dataset of N_i examples for each organism i and created new data sets with 20 training examples, 40 evaluation examples and 200 test examples.

Experimental setup For model selection we used a grid over the following two parameter ranges $C = [100, 250], \gamma = [0, 0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 1.0]$ for each node in the taxonomy (cf. Figure 3.10). Sub-sampling of the dataset was performed 3 times and results were subsequently averaged. We compared our *Top-Down* algorithm to two baseline methods, one where predictors for all tasks where trained without information transfer (*independent*) and the other extreme case, where one global model was fitted for all tasks based on the union of all data sets (*union*). Performance was measured by the F-score, the harmonic mean of precision and recall, where precision and recall were determined on nucleotide level (e.g. whether or not an exonic nucleotide was correctly predicted) in single-gene regions. (Note that due to its per-nucleotide Markov restriction, however, our method is not able to exploit that there is only one gene per example sequence.)

Results Figure 3.11 shows the results for our proposed *Top-Down* method and the two baseline methods described above. We observe that it is generally beneficial to combine information from different organisms, as *union* always performs better than *independent*. Indeed *Top-Down* improves over the naïve combination method *union* with F-score increases of up to 4.05 percentage points in *A. tumefaciens*. On average, we observe an improvement of 13.99 percentage points for *Top-Down* over *independent* and 1.13 percentage points for *Top-Down* over *union*, confirming the value of *Top-Down* in transferring information across tasks. In addition, the new bundle method converges at least twice as fast as the originally proposed cutting plane method.

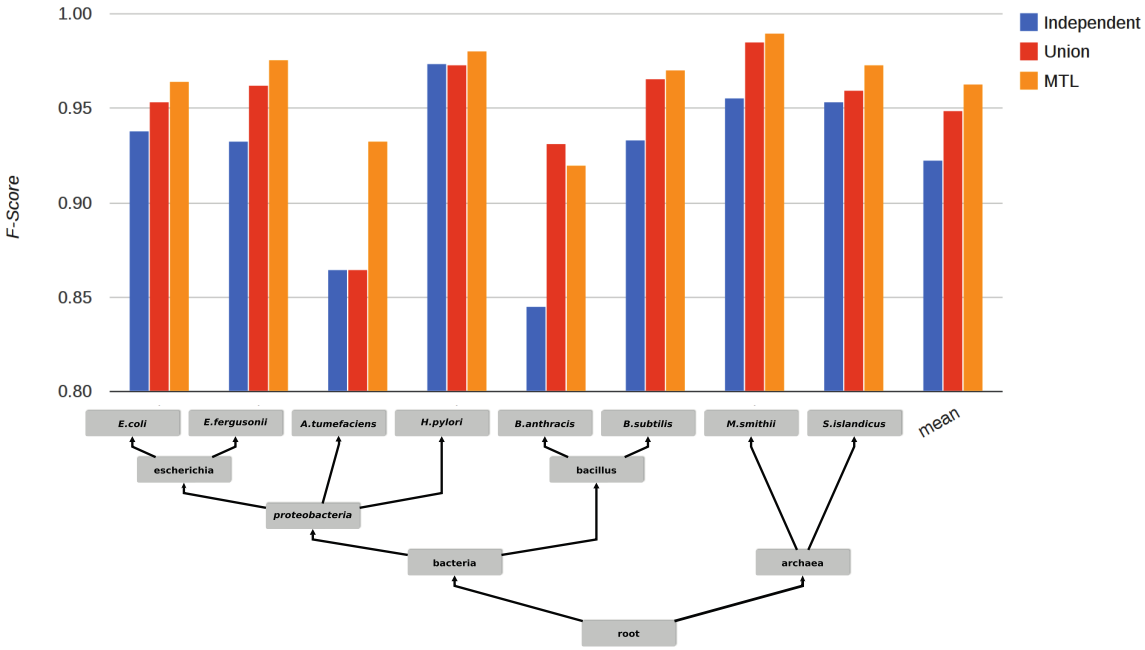


Figure 3.11: Evaluation of *Top-Down* and baselines *independent* and *union* [Görnitz* et al., 2011].

3.5 Summary

We have proposed a *principled way of adapting a trained SVM* to a new domain by introducing an additional regularization term. We have provided a *source-regularized SVM* formulation for the binary classification and the structured output learning case. To cope with the increased problem size when integrating data from multiple tasks, we have developed *extensions of established SVM solvers* (SVMLight, LibSVM and LibLinear) for the binary case, and also give *insight* between source-regularization and chunking strategies. For structured output learning, we have presented an efficient solver based on *bundle-methods* for to solve the source-regularized SO-SVM problem efficiently. Implementations of all algorithms were made available as part of Shogun [Sonnenburg et al., 2010]. Based on the idea of source-regularization, we derived an algorithm for transfer learning according to a given *tree structure*. We have demonstrated that our methods outperform baseline methods on synthetic data, splice-site prediction and Prokaryotic gene finding. The poor performance of *Plain* relative to *Top-Down* shows that exploiting information from other tasks is in fact beneficial. Oftentimes, the other baseline method *Union* achieves improvements upon *Plain*, the better performance of *Top-Down-SVM* indicates that there is no single model that fits all tasks equally. Clearly, methods that carefully combine the data from different tasks according to their relatedness perform best.

The proposed hierarchical algorithm has certain shortcomings, which we will discuss at this point. First, no globally optimal solution can be guaranteed, as we solve a sequence of optimization problems rather than a joint one. However, solving a sequence of learning problems also enables us to model select regularization parameters at each step, which in practice turned out to be of value (see Section 3.3). Second, training is expensive since a number of optimization problems have to be solved sequentially. However, the algorithm can easily be parallelized according to the underlying graph structure. On top of that, one can pursue a hot-start strategy when moving from one node to the next to speed up training.

An advantage of our method is that is allows the re-use of already trained models. For instance, ideas similar to the ones presented here, have been successfully been applied to brain-computer interfaces, where existing models of other subjects where *adapted* to new subjects using a similar approach [Alamgir et al., 2010]. Another potential advantage of our method is that we obtained a trained predictor for each inner node of the graph structure. This could be particularly useful in case we are planning to predict on yet *unseen tasks*. As a first step, one would first assign the new task to the closest node in the tree – this could be a leaf node or an *inner node* – and subsequently perform predictions using the respective classifier. For instance, this property could be of particular interest in the case of metagenomics, where new strains of bacteria are sequenced every day. Thus, if one was to model biological processes in different bacteria, one could therefore to build a family of trained models using the strategy that we have proposed and sequentially update this *tree of predictors* as more species become available.

4 Multi-Task Learning

The methods in this chapter differ from the approach taken in Chapter 3 in several respects. First, rather than adapting an existing classifier to a new task, we consider

The main **contributions** in this chapter are the following:

- a general framework of multitask learning that includes common formulations
- extensions of graph-regularized MTL that employs MKL to learn task similarities
- a fast dual-coordinate descent solver for our formulation
- the application of MTL and MT-MKL to various problems from computational biology and computer vision

Parts of this chapter are based on:

C. Widmer, N. C. Toussaint, Y. Altun, and G. Rätsch. Inferring latent task structure for Multitask Learning by Multiple Kernel Learning. In *BMC bioinformatics*, 11 Suppl 8(Suppl 8), 2010.

C. Widmer, M. Kloft, N. Görnitz and G. Rätsch. Efficient Training of Graph-Regularized Multitask SVMs. *European Conference on Machine Learning*, 2012.

C. Widmer, M. Kloft, G. Rätsch. A general framework for Multitask Multiple Kernel Learning. *Journal of Machine Learning Research* (JMLR), in preparation.

C. Widmer, N. C. Toussaint, Y. Altun, O. Kohlbacher and G. Rätsch. Novel machine learning methods for MHC Class I binding prediction. In *Pattern Recognition in Bioinformatics*, pages 98-109, 2010.

X. Lou, **C. Widmer**, M. Kang, G. Rätsch, and A.K. Hadjantonakis. Structured Domain Adaptation Across Imaging Modality: How 2D Data Helps 3D Inference. In *NIPS MLCB Workshop*, 2012.

S. Heinrich, E.-M. Geissen, J. Kamenz, S. Trautmann, **C. Widmer**, P. Drewe, ..., S. Hauf. Determinants of robustness in spindle assembly checkpoint signalling. In *Nature Cell Biology*, 15(11), 1328-39, 2013.

C. Widmer, G. Rätsch. Multitask Learning in Computational Biology. *Journal of Machine Learning Research Workshop and Conference Proceedings 27* Best paper award at *ICML 2011 Unsupervised and Transfer Learning Workshop*. pages 207-216., 2012.

C. Widmer, M. Kloft and G. Rätsch. Multi-task Learning for Computational Biology: Overview and Outlook. In *Vapnik Festschrift*, pages 117–127 , 2013.

C. Widmer, P. Drewe, X. Lou, S. Umrania, S. Heinrich and G. Rätsch. GRED: Graph-Regularized 3D Shape Reconstruction from Highly Anisotropic and Noisy Images. In *arXiv preprint*, arXiv:1309.4426, 2013.

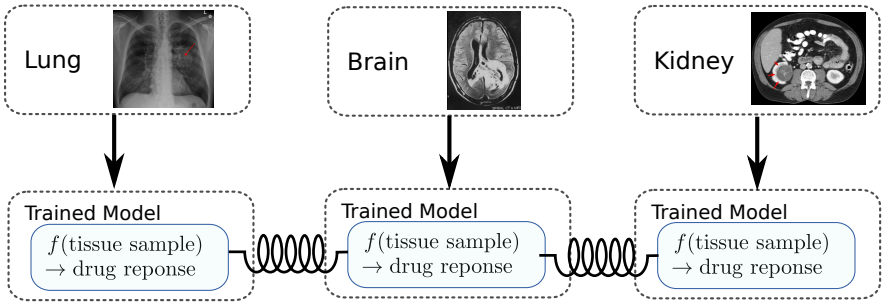


Figure 4.1: Example for jointly learning models for several tumor types. Figure is based on pictures from Wikipedia, see Section A in the Appendix for image licenses.

the problem of learning models for several related problems simultaneously, similar to what Caruana [1997] have explored in early work on multitask learning. Second, we now consider task relationships that may be described by a general graph structure rather than restricting them to directed acyclic graphs as before. Lastly, instead of assuming a fixed task relationship, we derive a framework for multitask learning methods that allows the learning or refinement of task similarities from data.

4.1 A Unifying View of Regularized Multi-Task Learning

In Section 1.5, we have given an overview of the family of methods considered in this thesis. We now develop a novel multi-task framework that subsumes these and many other formulations, allowing us to view prevalent approaches from a unifying perspective, yielding new insights. Furthermore, we derive new learning machines as special instantiations of this general model. Our approach is embedded into the framework of regularization-based supervised learning methods. Recall from Section 2.3.1 that in regularized risk minimization, we minimize a functional

$$\mathfrak{R}(\mathbf{w}) + C\,\mathfrak{L}(\mathbf{w})\,,$$

which consists of a loss-term $\mathfrak{L}(\mathbf{w})$ measuring the training error and a regularizer $\mathfrak{R}(\mathbf{w})$ penalizing the complexity of the model $\mathbf{w}$. The positive constant $C > 0$ controls the trade-off of the criterion. The formulation can easily be generalized to the multi-task setting, where we are interested in obtaining several models parametrized by $\mathbf{w}_1, \ldots, \mathbf{w}_T$, where T is the number of tasks.

In the past, this has been achieved by employing a joint regularization term $\mathfrak{R}(\mathbf{w}_1, \ldots, \mathbf{w}_T)$ that penalizes the discrepancy between the individual models [Agarwal et al., 2010; Evgeniou et al., 2005],

$$\mathfrak{R}(\mathbf{w}_1, \ldots, \mathbf{w}_T) + C\,\mathfrak{L}(\mathbf{w}_1, \ldots, \mathbf{w}_t)\,.$$

A common approach is, for example, to set $\mathfrak{R}(\mathbf{w}_1, \ldots, \mathbf{w}_T) = \frac{1}{2}\sum_{s,t=1}^{T} q_{st}\,\|\mathbf{w}_s - \mathbf{w}_t\|^2$, where $Q = (q_{st})_{a \le s,t \le T}$ is a task similarity matrix. In this thesis, we develop a novel,

general framework for multi-task learning of the form

$$\min_{\boldsymbol{W},\theta} \; \mathfrak{R}(\boldsymbol{W},\boldsymbol{\theta}) + C\mathfrak{L}(\boldsymbol{W})\,,$$

where $\boldsymbol{W} = (W_m)_{1\le m\le M}$, $W_m = (\mathbf{w}_{m1},\ldots,\mathbf{w}_{mT})$. This approach has the additional flexibility of allowing us to incorporate *multiple task similarity matrices* into the learning problem, each equipped with a weighting factor. Instead of specifying the weighting factor a priori, we will automatically determine optimal weights from the data as part of the learning problem. We show that the above formulation comprises many existing lines of research in the area; this not only includes very recent lines but also seemingly different ones. The unifying framework allows us to analyze a large variety of MTL methods jointly, as exemplified by deriving a general dual representation of the criterion, without making assumptions on the employed norms and losses, besides the latter being convex. This delivers insights into connections between existing MTL formulations and, even more importantly, can be used to derive *novel* MTL formulations as special cases of our framework, as done in Section 4.1.6.

4.1.1 Problem Setting and Notation

Let $D = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ be a set of training pattern/label pairs. In multitask learning, each training example (x_i,y_i) is associated with a task $\tau(i) \in \{1,\ldots,T\}$. Furthermore, we assume that for each $t \in \{1,\ldots,T\}$ the instances associated with task t are independently drawn from a probability distribution P_t over a measurable space $\mathcal{X}_t \times \mathcal{Y}_t$. We denote the set of indices of training points of the tth task by $I_t := \{i \in \{1,\ldots,n\} : \tau(i) = t\}$. The goal is to find, for each task $t \in \{1,\ldots,T\}$, a prediction function $f_t : \mathcal{X} \to \mathbb{R}$. In this thesis, we consider composite functions of the form $f_t : x \mapsto \sum_{m=1}^{M} \langle \mathbf{w}_{mt}, \varphi_m(x)\rangle$, $1 \le t \le T$, where $\varphi_m : \mathcal{X} \to \mathcal{H}_m$, $1 \le m \le M$, are mappings into reproducing Hilbert spaces $\mathcal{H}_1,\ldots,\mathcal{H}_M$, encoding multiple views of the multi-task learning problem via kernels $k_m(x,\tilde{x}) = \langle \varphi_m(x), \varphi_m(\tilde{x})\rangle$, and $\boldsymbol{W} := (\mathbf{w}_{mt})_{1\le m\le M, 1\le t\le T}$, $w_{mt} \in \mathcal{H}_m$ are parameter vectors of the prediction function.

For simplicity of notation, we concentrate on binary prediction, i.e., $\mathcal{Y} = \{-1,1\}$, and encode the loss of the prediction problem as a loss term $\mathfrak{L}(\boldsymbol{W}) := \sum_{i=1}^{n} l(y_i f_{\tau(i)}(x_i))$, where $l : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ is a loss function, assumed to be closed convex, lower bounded and finite at 0. To consider sophisticated couplings between the tasks, we introduce so-called *task-similarity matrices* $Q_1,\ldots,Q_M \in \mathrm{GL}_n(\mathbb{R})$ with $Q_m = (q_{mst})_{1\le s,t\le T}$, $Q_m^{-1} = \left(q_{mst}^{(-1)}\right)_{1\le s,t\le T}$ and consider the regularizer $\mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) = \frac{1}{2}\sum_{m=1}^{M} \|W_m\|_{Q_m}^2/\theta_m$ (setting $1/0 := \infty$, $0/0 := 0$) with $\|W_m\|_{Q_m} := \mathrm{tr}(W_m Q_m W_m^*) = \sqrt{\sum_{s,t=1}^{T} q_{mst}\,\langle \mathbf{w}_{ms}, \mathbf{w}_{mt}\rangle}$, where $W_m = (\mathbf{w}_{m1},\ldots,\mathbf{w}_{mT}) \in \bigoplus_{t=1}^{T} \mathcal{H}_m =: \mathcal{H}_m^T$ with adjoint W_m^* and $\mathrm{tr}(\cdot)$ denotes the trace class operator of the tensor Hilbert space $\mathcal{H}_m \otimes \mathcal{H}_m$. Note that also the direct sum $\mathcal{H} := \bigoplus_{m=1}^{M} \mathcal{H}_m^T$ is a Hilbert space, which will allow us to view $\boldsymbol{W} \in \mathcal{H}$ as an element in a Hilbert space. The parameters $\boldsymbol{\theta} = (\theta_m)_{1\le m\le M} \in \Theta_p$, $\Theta_p := \{\boldsymbol{\theta} \in \mathbb{R}^M : \theta_m \ge 0, 1 \le m \le M, \|\boldsymbol{\theta}\|_p \le 1\}$, are adaptive weights of the views, where $\|\boldsymbol{\theta}\|_p = \sqrt[p]{\sum_{m=1}^{M} |\theta_m|^p}$ denotes the ℓ_p-norm. Here $\boldsymbol{\theta} \succeq \mathbf{0}$ denotes $\theta_m \ge 0$, $m = 1,\ldots,M$.

Using the above specification of the regularizer and the loss term, we study the following unifying primal optimization problem [Widmer et al., 2013b, 2014, in prep.].

Problem 4.1.1 (Primal problem)**.** *Solve*

$$\inf_{\boldsymbol{\theta}\in\Theta_p,\boldsymbol{W}\in\mathcal{H}} \quad \mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) \;+\; C\,\mathfrak{L}\big(A(\boldsymbol{W})\big)\,,$$

where

$$\mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) := \frac{1}{2}\sum_{m=1}^{M}\frac{\|W_m\|_{Q_m}^2}{\theta_m}\,, \quad \|W_m\|_{Q_m}^2 := \operatorname{tr}(W_m Q_m W_m^*)$$

$$\mathfrak{L}(A(\boldsymbol{W})) := \sum_{i=1}^{n} l\big(A_i(\boldsymbol{W})\big)\,, \quad A(\boldsymbol{W}) := (A_i(\boldsymbol{W}))_{1\le i\le n}\,, \quad A_i(\boldsymbol{W}) := y_i\sum_{m=1}^{M}\big\langle \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\big\rangle\,.$$

In the above formulation, we express the multitask learning problem with respect to M positive definite task-similarity matrices Q_m, each of which carry their own weight Θ_m. These task-similarity matrices Q_m are the principal way of introducing prior knowledge about task relationships into the learning problem and are assumed to be *given*. More detail on how these Q_m are constructed given varying degree of prior knowledge (e.g. grouping of tasks, hierarchical task relationship, task graph) is discussed in Section 4.1.5. A key aspect of this formulation is that these weights Θ_m can be learned from data by solving the above *convex* optimization problem. It is important to note that we do not yet commit to any specific loss function, which makes our framework very flexible in deriving novel instantiations. To facilitate understanding, $\boldsymbol{W}$ can be thought of as a tensor from $\mathbb{R}^{n,T,M}$ (for finite dimensional feature spaces), where the corresponding adjoint map can be thought of as a generalization of a matrix transpose. However, in the formulation stated above, all operators are expressed with respect to Hilbert spaces, making it even more general (e.g. allowing the use of RBF kernels).

4.1.2 Dualization

Dual representations of optimization problems deliver insight into the problem, which can be used in practice to develop efficient optimization algorithms (so done in Section 4.2). In this section, we derive a dual representation of our unifying primal optimization problem, i.e., Problem 4.1.1. Our dualization approach is based on Fenchel-Rockafellar duality theory. The basic results of Fenchel-Rockafellar duality theory for Hilbert spaces are reviewed in Section 2.2.2. We present two dual optimization problems: one that is dualized with respect to $\boldsymbol{W}$ only (i.e., considering $\boldsymbol{\theta}$ as being fixed) and one that completely removes the dependency on $\boldsymbol{\theta}$.

Computation of Conjugates and Adjoint Map

To apply Fenchel's duality theorem, we need to compute the adjoint map A^* of the linear map $A : \mathcal{H} \to \mathbb{R}^n$, $A(\boldsymbol{W}) = \big(A_i(\boldsymbol{W})\big)_{1\le i\le n}$, as well as the convex conjugates of

$\mathfrak{R}$ and $\mathfrak{L}$. See Section 2.2.2 for a review of the definitions of the convex conjugate and the adjoint map. First, we notice that, by the basic identities for convex conjugates of Prop. 2.2.4 in Section 2.2.2, we have that

$$\big(C\mathfrak{L}(\boldsymbol{\alpha})\big)^* \;=\; C\mathfrak{L}^*(\boldsymbol{\alpha}/C) \;=\; C\Big(\sum\nolimits_{i=1}^n l(\alpha_i/C)\Big)^* \;=\; C\sum\nolimits_{i=1}^n l^*(\alpha_i/C)\,.$$

Next, we define $A^* : \mathbb{R}^n \to \mathcal{H}$ by $A^*(\boldsymbol{\alpha}) = \big(\sum_{i\in I_t}\alpha_i y_i \varphi_m(x_i)\big)_{1\le m\le M, 1\le t\le T}$. Recall that the mapping between tasks and examples may be expressed in one of two ways. We may use index set I_t to retrieve the indices of training examples associated with task t. Alternatively, we may use task indicator $\tau(i) \in \{1,\ldots,T\}$ to obtain the task index $\tau(i)$ associated with ith training example. Using this notation, we verify that, for any $\boldsymbol{W} \in \mathcal{H}$ and $\boldsymbol{\alpha} \in \mathbb{R}^n$, it holds

$$\begin{aligned}
\langle \boldsymbol{W}, A^*(\boldsymbol{\alpha})\rangle &= \Big\langle \big(w_{mt}\big)_{1\le m\le M,1\le t\le T}\,,\Big(\sum\nolimits_{i\in I_t}\alpha_i y_i\varphi_m(x_i)\Big)_{1\le m\le M,1\le t\le T}\Big\rangle \\
&= \sum_{m=1}^M\sum_{t=1}^T\sum_{i\in I_t}\alpha_i y_i\,\langle w_{mt},\varphi_m(x_i)\rangle \\
&= \sum_{i=1}^n\sum_{m=1}^M \alpha_i y_i\,\big\langle w_{m\tau(i)},\varphi_m(x_i)\big\rangle \\
&= \langle A(\boldsymbol{W}),\boldsymbol{\alpha}\rangle\;.
\end{aligned}$$

Thus, A^* as defined above is indeed the adjoint map. Finally, we compute the conjugate of $\mathfrak{R}$ with respect to $\boldsymbol{W}$, where we consider $\boldsymbol{\theta}$ as a constant (be reminded that Q_m are given). We write $r_m(W_m) := \frac{1}{2}\|W_m\|^2_{Q_m}$ and note that, by Prop. 2.2.4,

$$\mathfrak{R}^*_{\boldsymbol{\theta}}(\boldsymbol{W}) \;=\; \left(\sum_{m=1}^M \theta_m^{-1} r_m(W_m)\right)^* \;=\; \sum_{m=1}^M \theta_m^{-1} r_m^*(\theta_m W_m)\,.$$

Furthermore,

$$r_m^*(W_m) = \sup_{V_m\in\mathcal{H}_m^T} \underbrace{\langle V_m, W_m\rangle - \frac{1}{2}\operatorname{tr}(V_m Q_m V_m)}_{=:\psi(V_m)}\;. \tag{4.1}$$

The supremum is attained when $\nabla_{V_m}\psi(V_m) = 0$ so that in the optimum $V_m = Q_m^{-1}W_m$. Resubstitution into (4.1) gives $r_m^*(W_m) = \frac{1}{2}\operatorname{tr}(W_m Q_m^{-1} W_m) = \frac{1}{2}\|W_m\|^2_{Q_m^{-1}}$, so that we have

$$\mathfrak{R}^*_{\boldsymbol{\theta}}(\boldsymbol{W}) = \frac{1}{2}\sum_{m=1}^M \theta_m \|W_m\|^2_{Q_m^{-1}}\;.$$

Dual Optimization Problems

We may now apply Fenchel's duality theorem (cf. Theorem 2.2.3 in Section 2.2.2), which gives the following dual MTL problem:

Problem 4.1.2 (Dual problem—partially dualized minimax formulation)**.** *Solve*

$$\inf_{\boldsymbol{\theta}\in\Theta_p} \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha})) \;-\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C)\,, \tag{4.2}$$

where

$$\begin{aligned} &\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha})) = \frac{1}{2}\sum_{m=1}^{M}\theta_m \left\|A^*_m(\boldsymbol{\alpha})\right\|^2_{Q_m^{-1}}\,, \quad \mathfrak{L}^*(\boldsymbol{\alpha}) = \sum_{i=1}^{n} l^*(\alpha_i)\,, \\ &A^*(\boldsymbol{\alpha}) := (A^*_m(\boldsymbol{\alpha}))_{1\le m\le M}\,, \quad A^*_m(\boldsymbol{\alpha}) = \Big(\textstyle\sum_{i\in I_t}\alpha_i y_i \varphi_m(x_i)\Big)_{1\le t\le T}\,. \end{aligned} \tag{4.3}$$

The above problem involves minimization with respect to (the primal variable) $\boldsymbol{\theta}$ and maximization with respect to (the dual variable) $\boldsymbol{\alpha}$. The optimization algorithm presented in later sections is based on this minimax formulation. However, we may completely remove the dependency on $\boldsymbol{\theta}$, which sheds further insights into the problem, which will later be exploited for optimization, i.e., to control the duality gap of the computed solutions.

To remove the dependency on $\boldsymbol{\theta}$, we first note that Problem 4.1.2 is convex (even affine) in $\boldsymbol{\theta}$ and concave in $\boldsymbol{\alpha}$ and thus, by Sion's minimax theorem, we may exchange the order of minimization and maximization:

$$\begin{aligned} \text{Eq. (4.2)} \;&=\; \inf_{\boldsymbol{\theta}\in\Theta_p} \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\frac{1}{2}\sum_{m=1}^{M}\theta_m \left\|A^*_m(\boldsymbol{\alpha})\right\|^2_{Q_m^{-1}} \;-\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C) \\ &=\; \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\sup_{\boldsymbol{\theta}\in\Theta_p} \frac{1}{2}\sum_{m=1}^{M}\theta_m \left\|A^*_m(\boldsymbol{\alpha})\right\|^2_{Q_m^{-1}} \;+\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C) \\ &=\; \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\frac{1}{2}\left\| \left(\left\|A^*_m(\boldsymbol{\alpha})\right\|^2_{Q_m^{-1}}\right)_{1\le m\le M} \right\|_{p^*} \;+\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C) \end{aligned}$$

where the last step is by the definition of the dual norm, i.e., $\sup_{\boldsymbol{\theta}\in\Theta_p}\langle\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}\rangle = \|\widetilde{\boldsymbol{\theta}}\|_{p^*}$ and $p^* := p/(p-1)$ denotes the conjugated exponent. We thus have the following alternative dual problem.

Problem 4.1.3 (Dual problem—completely dualized formulation)**.** *Solve*

$$\sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\frac{1}{2}\left\| \left(\left\|A^*_m(\boldsymbol{\alpha})\right\|^2_{Q_m^{-1}}\right)_{1\le m\le M} \right\|_{p^*} \;+\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C)$$

where

$$\mathfrak{L}^*(\boldsymbol{\alpha}) = \sum_{i=1}^{n} l^*(\alpha_i)\,, \quad A^*_m(\boldsymbol{\alpha}) = \Big(\textstyle\sum_{i\in I_t}\alpha_i y_i \varphi_m(x_i)\Big)_{1\le t\le T}\,.$$

4.1.3 Representer Theorem

Fenchel's duality theorem (Theorem 2.2.3 in Section 2.2.2) yields a useful optimality condition, that is,

$$(\boldsymbol{W}^\star, \boldsymbol{\alpha}^\star) \text{ optimal } \quad \Leftrightarrow \quad \boldsymbol{W}^\star = \nabla g^*(A^*(\boldsymbol{\alpha}^\star))\,,$$

under the minimal assumption that $g \circ A^*$ is differentiable in $\boldsymbol{\alpha}^\star$. The above requirement can be thought of as an analog to the KKT condition *stationarity* in Lagrangian duality. Note that we can rewrite the above equation by inserting the definitions of g and A from the previous subsection; this gives, for any $m = 1, \ldots, M$,

$$\forall m = 1, \ldots, M: \quad W_m^\star = \theta_m Q_m^{-1} \left(\sum\nolimits_{i \in I_t} \alpha_i^\star y_i \varphi_m(x_i) \right)_{1 \leq t \leq T}\,,$$

which we may rewrite as

$$\forall\, m = 1, \ldots, M,\, t = 1, \ldots, T: \quad \mathbf{w}_{mt}^\star = \theta_m \sum_{i=1}^{n} q_{m\tau(i)t}^{(-1)} \alpha_i^\star y_i \varphi_m(x_i)\,. \tag{4.4}$$

The above equation gives us a representer theorem [Argyriou et al., 2009] for the optimal $\boldsymbol{W}^\star$, which we will exploit later in this thesis for deriving an efficient optimization algorithm to solve Problem 4.1.1.

4.1.4 Relation to Multiple Kernel Learning

Evgeniou et al. [2005] introduce the notion of a *multi-task kernel*. We can generalize this framework by defining multiple multi-task kernels

$$\tilde{k}_m(x_i, x_j) := q_{m\tau(i)\tau(j)}^{(-1)} k_m(x_i, x_j)\,, \quad m = 1, \ldots, M\,. \tag{4.5}$$

To see this, first note that the term $\|A_m^*(\boldsymbol{\alpha})\|_{Q^{-1}}^2$ can alternatively be written as

$$\begin{aligned}
\|A_m^*(\boldsymbol{\alpha})\|_{Q_m^{-1}}^2 &= \operatorname{tr}\left(A_m^*(\boldsymbol{\alpha})\, Q_m^{-1}\, A_m^*(\boldsymbol{\alpha})^* \right) \\
&= \operatorname{tr}\left(\left(\sum\nolimits_{i \in I_s} \alpha_i y_i \varphi_m(x_i) \right)_{1 \leq s \leq T} Q_m^{-1} \left(\sum\nolimits_{i \in I_t} \alpha_i y_i \varphi_m(x_i) \right)_{1 \leq t \leq T}^* \right) \\
&= \sum_{s,t=1}^{T} q_{mst}^{(-1)} \left\langle \sum\nolimits_{i \in I_s} \alpha_i y_i \varphi_m(x_i), \sum\nolimits_{i \in I_t} \alpha_i y_i \varphi_m(x_i) \right\rangle \\
&= \sum_{s,t=1}^{T} q_{mst}^{(-1)} \sum\nolimits_{i \in I_s, j \in I_t} \alpha_i \alpha_j y_i y_j \underbrace{\varphi_m(x_i) \varphi_m(x_j)}_{= k_m(x_i, x_j)} \\
&= \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{q_{m\tau(i)\tau(j)}^{(-1)} k_m(x_i, x_j)}_{\tilde{k}_m(x_i, x_j)}\,.
\end{aligned} \tag{4.6}$$

	loss $l(a), a \in \mathbb{R}$	dual loss $l^*(a)$
hinge loss	$\max(0, 1-a)$	$\begin{cases} a, \text{ if } -1 \leq a \leq 0 \\ \infty, \text{ elsewise} \end{cases}$
logistic loss	$\log(1+\exp(-a))$	$-a\log(-a) + (1+a)\log(1+a)$

Table 4.1: Some loss functions used in this thesis and corresponding conjugate functions.

so it follows

$$\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha})) \;=\; \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} \theta_m \tilde{k}_m(x_i, x_j)$$

and thus Problem 4.1.2 becomes

$$\inf_{\boldsymbol{\theta}\in\Theta_p} \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} \; -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} \theta_m \tilde{k}_m(x_i, x_j) \;-\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C)\,, \tag{4.7}$$

which is an ℓ_p-regularized multiple-kernel-learning problem over the kernels $\tilde{k}_1, \ldots, \tilde{k}_M$ [Kloft et al., 2011].

4.1.5 Specific Instantiations of the Framework

In this section, we show that several regularization-based multi-task learning machines are subsumed by the generalized primal and dual formulations of Problems 4.1.1–4.1.2. As a first step, we specialize our general framework to the hinge-loss, and show its primal and dual form. Based on this, we then instantiate our framework further to known methods in increasing complexity, starting with single-task learning (standard SVM) and working towards graph-regularized multitask learning and its relation to multitask kernels. Finally, we derive several novel methods from our general framework.

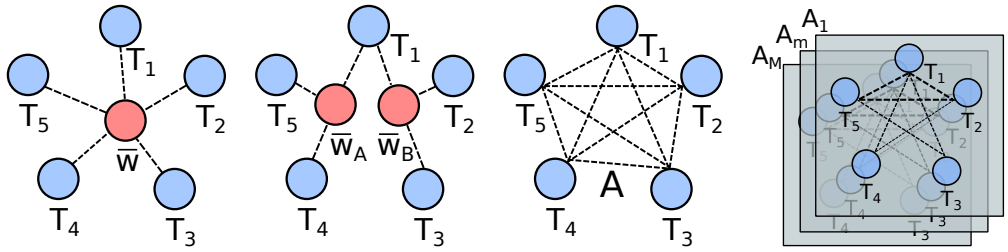
Hinge Loss

Many existing multi-task learning machines utilize the hinge loss $l(a) = \max(0, 1-a)$. Employing the hinge loss in Problem 4.1.1, yields the loss term

$$\mathfrak{L}(A(\boldsymbol{W})) = \sum_{i=1}^{n} \max\left(0, 1 - y_i \sum\nolimits_{m=1}^{M} \left\langle \mathbf{w}_{m\tau(i)}, \varphi_m(x_i) \right\rangle\right).$$

As shown in Table 4.1, the conjugate of the hinge loss is $l^*(a) = a$, if $-1 \leq a \leq 0$ and ∞ otherwise, which is easily verified by elementary calculus. Thus, we have

$$-\,C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C) = -C\sum_{i=1}^{n} l^*(-\alpha_i/C) = \sum_{i=1}^{n} \alpha_i, \tag{4.8}$$



(a) MTL (b) Clustered MTL (c) Graph MTL (d) Multi-Graph MTL

Figure 4.2: Graphical Illustration of Common Multi-Task formulations Figure 4.2(a) shows the case of uniform task relationships (see Section 4.1.5), Figure 4.2(b) extends this to the case of several task clusters (see Section 4.1.5), Figure 4.2(c) shows MTL where task relationship are given by a graph (see Section 4.1.5), and finally Figure 4.2(d) shows an extension of the latter to multiple graphs as discussed in Section 4.1.6.

provided that $\forall i = 1, \ldots, n : 0 \leq \alpha_i \leq C$; otherwise we get $-C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C) = -\infty$. Hence, for the hinge-loss, we obtain the following pair of primal and dual problem.

Primal:

$$\inf_{\substack{\boldsymbol{\theta}\in\Theta_p \\ \boldsymbol{W}\in\mathcal{H}}} \quad \frac{1}{2}\sum_{m=1}^{M}\frac{\|W_m\|_{Q_m}^2}{\theta_m} \;+\; C\sum_{i=1}^{n}\max\left(0, 1 - y_i\sum\nolimits_{m=1}^{M}\left\langle \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\right\rangle\right) \tag{4.9}$$

Dual:

$$\inf_{\boldsymbol{\theta}\in\Theta_p} \quad \sup_{\mathbf{0}\preceq\boldsymbol{\alpha}\preceq\mathbf{C}} \quad -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \sum_{m=1}^{M}\theta_m\tilde{k}_m(x_i,x_j) \;+\; \sum_{i=1}^{n}\alpha_i, \tag{4.10}$$

where $\tilde{k}$ is the multitask kernel as defined in Equation 4.5.

Single Task Learning

Starting from the simplest special case, we briefly show how single-task learning methods may be recovered from our general framework. By mapping well understood single-task methods onto our framework, we hope to achieve two things. First, we believe this will greatly facilitate understanding for the reader who is familiar with standard methods like the SVM. Second, we pave the way for applying efficient training algorithms developed in Section 4.2 to these single-task formulations, for example yielding a new linear solver for non-sparse Multiple Kernel Learning as a corollary.

Support Vector Machine In the case of the single-task ($\boldsymbol{W} = \mathbf{w}$, $Q = 1$), single kernel SVM ($M = 1$), the primal from Equation 4.9 can be greatly simplified to

$$\inf_{\mathbf{w}\in\mathcal{H}} \quad \frac{1}{2}\|\,\mathbf{w}\,\|^2 \;+\; C\sum_{i=1}^{n}\max\left(0, 1 - y_i\left\langle\,\mathbf{w}, \varphi(x_i)\right\rangle\right),$$

which corresponds to the well-established linear SVM formulation (without bias). Similarly, the dual is readily obtained from Equation 4.10 and is given by

$$\sup_{\mathbf{0} \preceq \boldsymbol{\alpha} \preceq \mathbf{C}} \; -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \; + \; \sum_{i=1}^{n} \alpha_i \,.$$

q-norm MKL by Kloft et al. [2011] is obtained as another special case of our framework. This case is of particular interest, as we will see in Section 4.2, because we obtain a novel linear solver for q-norm MKL as a corollary. By restricting the number of tasks to one (i.e. $T = 1$), $\boldsymbol{W}_m$ becomes $\mathbf{w}_m$ and $Q = 1$. Equation (4.9) reduces to:

$$\inf_{\boldsymbol{\theta} \in \Theta_p, \boldsymbol{W} \in \mathcal{H}} \; \frac{1}{2} \sum_{m=1}^{M} \frac{\| \mathbf{w}_m \|^2}{\theta_m} \; + \; C \sum_{i=1}^{n} \max \left(0, 1 - y_i \sum\nolimits_{m=1}^{M} \left\langle \mathbf{w}_m, \varphi_m(x_i) \right\rangle \right) \,.$$

In agreement with Kloft et al. [2011], we recover the dual formulation from Equation 4.10.

$$\inf_{\boldsymbol{\theta} \in \Theta_p} \; \sup_{\mathbf{0} \preceq \boldsymbol{\alpha} \preceq \mathbf{C}} \; -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} \theta_m k_m(x_i, x_j) \; + \; \sum_{i=1}^{n} \alpha_i \,.$$

Multitask Learning

Here, we first derive the primal and dual formulations of regularization-based multitask learning as a special case of our framework and then give an overview of existing variants that can be mapped onto this formulation as a precursor to novel instantiations in Section 4.1.6. In this setting, we deal with Multiple tasks t, but only a single kernel or task similarity measure Q (i.e. $M = 1$). The primal thus becomes:

$$\inf_{\boldsymbol{W} \in \mathcal{H}} \; \frac{1}{2} \|W\|_Q^2 \; + \; C \sum_{i=1}^{n} \max \left(0, 1 - y_i \left\langle \mathbf{w}_{\tau(i)}, \varphi(x_i) \right\rangle \right) \,, \tag{4.11}$$

with corresponding dual

$$\sup_{\mathbf{0} \preceq \boldsymbol{\alpha} \preceq \mathbf{C}} \; -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \tilde{k}(x_i, x_j) \; + \; \sum_{i=1}^{n} \alpha_i \,, \tag{4.12}$$

where the definition of $\tilde{k}$ is given in Equation 4.5. As we will see in the following, the above formulation captures several existing MTL approaches, which can be expressed by choosing different encodings Q for task similarity.

Frustratingly Easy Domain Adaptation In a publication titled *Frustratingly Easy Domain Adaptation*, Daumé [2007] present a simple, yet appealing special case of graph-regularized MTL. They considered the setting of only two tasks (source task and target

task), with a fix task relationship (i.e. the influence of the two tasks on each other was not determined by their actual similarity). Their idea was to assign a higher base-similarity to pairs of examples from the same task than between examples from different tasks. This may be expressed by the following multitask kernel:

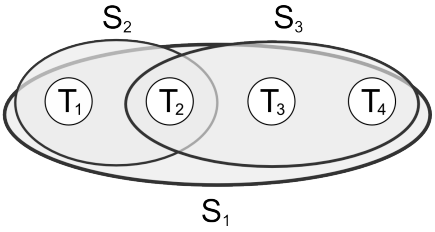
$$\tilde{k}(x,z) = \begin{cases} 2k(x,z) & \tau(x) = \tau(z) \\ k(x,z) & \text{else}\,. \end{cases}$$

From the above, we can readily read off the corresponding Q^{-1} (and compute Q).

$$Q^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \qquad Q = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix}.$$

Given the above, we can express this special case in terms of Equation 4.11 and 4.12. With some elementary algebra, this method can be viewed as *pulling* weight vectors of source $\mathbf{w}_s$ and target $\mathbf{w}_t$ towards a common mean vector $\bar{\mathbf{w}}$ by means of a regularization term. If we generalize this idea to allow for multiple cluster centers, we arrive at *task clustering*, which is described in the following.

Task Clustering Regularization Here, tasks are grouped into M clusters, whereas parameter vectors of tasks within each cluster are pulled towards the respective cluster center $\bar{\mathbf{w}}_m = \frac{1}{T_m}\sum_{t=1}^{T_m} \mathbf{w}_t$, where T_m is the number of tasks in cluster m [Evgeniou et al., 2005]. To understand what Q and Q^{-1} correspond to in terms of Equations 4.11



and 4.12, consider the definition of the multitask regularizer $\mathfrak{R}$ for task clustering.

$$R(\mathbf{w}_1,\ldots,\mathbf{w}_T) = \frac{1}{2}\left(\sum_{t=1}^{T}\lambda\left\|\mathbf{w}_t\right\|^2 + \sum_{m=1}^{M}\left(\rho\left\|\bar{\mathbf{w}}_m\right\|^2 + \sum_{t=1}^{T}\rho_m^t\left\|\mathbf{w}_t - \bar{\mathbf{w}}_m\right\|^2\right)\right) \tag{4.13}$$

$$= \frac{1}{2}\left(\sum_{t=1}^{T}\lambda\left\|\mathbf{w}_t\right\|^2 + \sum_{s,t=1}^{T} G_{s,t}\langle\mathbf{w}_s,\mathbf{w}_t\rangle\right) \tag{4.14}$$

$$= \frac{1}{2}\operatorname{tr}\left(W(\lambda I + G)W^\top\right), \tag{4.15}$$

where M is the number of clusters, $\rho_m^t \geq 0$ encodes assignment of task t to cluster m, ρ controls regularization of cluster centers $\bar{\mathbf{w}}_m$ and G are given by

$$G_{s,t} = \sum_{m=1}^{M}\left(\rho_m^t\delta_{st} - \frac{\rho_m^s\rho_m^t}{\rho + \sum_{r=1}^{T}\rho_m^r}\right).$$

If any task t is assigned to at least one cluster m (i.e. $\forall t \exists m : \rho_m^t > 0$) G is positive definite [Evgeniou et al., 2005] and we can express the above in terms of our primal formulation in Equation 4.11 as $Q = (\lambda I + G)$ and the corresponding dual as $Q^{-1} = (\lambda I + G)^{-1}$, even for $\lambda = 0$. We note that the formulation given in Section 4.1.5 may by expressed via task clustering regularization, by choosing only one cluster (i.e. $M = 1$) and setting $\lambda = 0$, $\rho = 1$ and $\rho_1^{\text{source}} = \rho_1^{\text{target}} = 1$, we get $G_{s,t} = \delta_{s,t} - \frac{1}{3}$, equating to the task similarity matrix Q from the previous section.

Graph-regularized MTL Graph-regularized MTL was established by Evgeniou et al. [2005] and constitutes one of the most influential MTL approaches to date. Their method is based on the following multitask regularizer, which also forms one of the main inspirations for our framework:

$$R(\mathbf{w}_1, ..., \mathbf{w}_T) = \frac{1}{2}\left(\sum_{t=1}^{T} \|\mathbf{w}_t\|^2 + \sum_{s,t=1}^{T} a_{st} \|\mathbf{w}_s - \mathbf{w}_t\|^2\right) \tag{4.16}$$

$$= \frac{1}{2}\left(\sum_{t=1}^{T} \|\mathbf{w}_t\|^2 + \sum_{s,t=1}^{T} l_{s,t}\langle \mathbf{w}_s, \mathbf{w}_t\rangle\right) \tag{4.17}$$

$$= \frac{1}{2}\operatorname{tr}\left(W(I + L)W^\top\right), \tag{4.18}$$

where $A = (a_{st})_{1 \le s,t \le T} \in \mathbb{R}^{T \times T}$ is a given graph adjacency matrix encoding the pairwise similarities of the tasks, $L = D - A$ denotes the corresponding graph Laplacian, where $D_{i,j} := \delta_{i,j} \sum_k A_{i,k}$, and I is a $T \times T$ identity matrix. Note that the number of zero eigenvalues of the graph Laplacian corresponds to the number of connected components [Von Luxburg, 2007]. We may view graph-regularized MTL as an instantiation of our general primal problem, Problem 4.1.1, where we have only one task similarity measure $Q_1 = I + L$ (i.e., $M = 1$). A visualization of this approach is given in Figure 3.3(c). As the graph Laplacian L is not invertible in general, we use its pseudo-inverse $L^\dagger$ to express the dual formulation of the above MTL regularizer.

$$Q_{s,t}^{-1} = L_{s,t}^\dagger = \sum_{i=1}^{r} \sigma_i \boldsymbol{v}_{is}^T \boldsymbol{v}_{it}, \tag{4.19}$$

where r is the rank of L, σ_i are the eigenvalues of L and $V = (\boldsymbol{v}_{s,t})$ is the orthogonal matrix of eigenvectors.

Multitask Kernels In contrast to graph-regularized MTL, where task relations are captured by an adjacency matrix or graph Laplacian as discussed in the previous paragraph, task relationships may directly be expressed in terms of a kernel on tasks K_{tasks}. This relationship has been illuminated in Section 4.1.4, where we have seen that the kernel on tasks corresponds to Q^{-1} in our dual MTL formulation. A formulation involving a combination of several MTL kernels with a fix weighting was explored by Jacob and Vert [2008a] in the context of Bioinformatics. In its most basic form, the authors considered a multitask kernel of the form

$$K((x,t),(z,s)) = K_{\text{base}}(x,z) \cdot K_{\text{tasks}}(t,s).$$

Furthermore, the authors considered a sum of different multi-task kernels, among them the corner cases $K_{\text{Dirac}}(t,s) = \delta_{s,t}$ (independent tasks) and the uniform kernel $K_{\text{Uni}}(t,s) = 1$ (uniformly related tasks). In general, their dual formulation is given by

$$K((x,t),(z,s)) = \sum_{m=1}^{M} K_{\text{base}}(x,z) \cdot K_{\text{tasks}}^{(m)}(t,s).$$

The above is a very interesting special case and can easily be expressed within our general framework. For this, consider the dual formulation given in Equation 4.10 for $Q^{(m)-1} = K_{\text{tasks}}^{(m)}$ and $\theta_1 = \ldots = \theta_M = 1$. In other words, the above also constitutes a form of multitask multiple kernel learning, however, without actually learning the kernel weights Θ_m. Nevertheless, the choice and discussion of different multitask kernels $K_{\text{tasks}}^{(m)}$ in [Jacob and Vert, 2008a] is of high relevance with respect to the family of methods explored in this work.

4.1.6 Proposing Novel Instances of Multitask Learning Machines

We now move ahead and derive novel instantiations from our general framework. Most importantly, we go beyond previous formulations by learning or refining task similarities from data using MKL *as an engine*.

Multi-cluster MT-MKL

Recall that in task clustering, parameter vectors of tasks within the same cluster are coupled (Equation 4.13). The *strength* of that coupling, however, has be be chosen in advance and remains fixed throughout the learning procedure. We extend the formulation of task clustering by introducing a weighting θ_m to task cluster m and tuning this weighting using our framework. We decompose G over clusters and arrive at the following MTL regularizer

$$R(\mathbf{w}_1,\ldots,\mathbf{w}_T) = \frac{1}{2}\left(\sum_{m=1}^{M}\|\mathbf{w}_m\|^2 + \sum_{m=1}^{M}\theta_m\sum_{s,t=1}^{T}G_{s,t}^m\langle\mathbf{w}_s,\mathbf{w}_t\rangle\right) \tag{4.20}$$

$$= \frac{1}{2}\sum_{m=1}^{M}\operatorname{tr}\left(\theta_m W(I+G^m)W^\top\right), \tag{4.21}$$

where G^m is given by

$$G_{s,t}^m = \rho_m^t\delta_{st} - \frac{\rho_m^s\rho_m^t}{\rho + \sum_{r=1}^{T}\rho_m^r}.$$

Note that, if not all tasks belong to the same cluster, G^m will not be invertible. Therefore, we need to express the mapping onto the dual of our general framework from Equation 4.10 in terms of the pseudo-inverse (see Equation 4.19) of G_m: $Q_m^{-1} = G_m^\dagger$.

Hierarchical Multi-Cluster MT-MKL

An important special case of *Multi-Cluster MT-MKL* is given by the scenario where task relationships are described by a hierarchical structure $\mathcal{G}$, such as a tree or a directed acyclic graph. Assuming hierarchical relations between tasks is particularly relevant to computational biology where often different tasks correspond to different organisms. In this context, we expect that the longer the common evolutionary history between two organisms, the more beneficial it is to share information between these organisms in a MTL setting. The tasks correspond to the leaves or terminal nodes and each inner node n_m defines a cluster m, by grouping tasks of all terminal nodes that are descendants of the current node n_m. As before, task clusters G can be used in the way discussed in the previous section. We expect that learning the contributions of the individual levels of the hierarchy makes sense for cases, where the edge lengths of $\mathcal{G}$ are unequal.

Powerset MT-MKL

With no prior information given, a natural choice is to take into account all possible subsets of tasks. Given a set of tasks $\mathcal{T}$, this corresponds to considering the power set $\mathcal{P}$ of $\mathcal{T}$ (excluding the empty set) $\mathcal{I}_{\mathcal{P}} = \{S | S \in \mathcal{P}(\mathcal{T}) \wedge S \neq \emptyset\}$. Clearly, this yields an exponential number (i.e. 2^T) of task clusters G^m of which only a few will be relevant. To identify the relevant task sets, we propose to use an L1-regularized MKL approach to yield a sparse solution. Most subset weights will be set to zero, yielding only a few relevant subsets with weights greater than zero. While L1-regularization of MKL results in a sparse combination of kernels, it does not address the computational complexity of the optimization problem over this exponential search space. With the current implementation, the method is limited to approximately 10 tasks depending on the number of training examples and available resources. However, there are possible extensions to handle the case where the number of tasks may become prohibitive, for instance, as proposed by Gehler and Nowozin [2008]. The idea is to iteratively generate new kernels based on the current solution $(\beta, \mathbf{w})$. These methods are known to converge to the optimal solution, if one can identify appropriate kernels in a larger set. In the current case, this could be done by solving an integer linear program. The realization of this extension is, however, subject to future work.

Multi-graph MT-MKL

One of the most popular MTL approaches is graph-regularized MTL by Evgeniou and Pontil [2004]. We have seen in Section 4.1.5, that such a graph is expressed as a adjacency matrix A and may alternatively be expressed in terms of its graph Laplacian L. Our extension readily deals with multiple graphs encoding task similarity $A_m = (a_{mst})_{1 \leq s,t \leq T} \in \mathbb{R}^{T \times T}$, which is of interest in cases where - as in Multiple kernel learning - we have access to alternative sources of task similarity and it is unclear which

one is best suited. This concept gives rise to the *multi-graph MTL* regularizer

$$R(\boldsymbol{W}) = \frac{1}{2}\operatorname{tr}\left(\sum\nolimits_{m=1}^{M} W_m(I + L_m)W_m^\top\right),$$

where L_m denotes the graph Laplacian corresponding to A_m. As before, we learn a weighting of the given graphs, therefore determining which measures are best suited to maximize prediction accuracy.

Adaptive MT-MKL

Typically, we are able to obtain a reasonable measure of task similarity from external sources. However, such a measure may require additional fine tuning in order to obtain a task similarity that maximizes prediction accuracy for the problem at hand. We therefore propose a strategy to learn a piecewise linear transformation of a given task similarity using our general framework. We start from a given task similarity matrix A. The objective is to find a decomposition $R_1, \ldots, R_M$ of A, such that A can be recovered as the weighted sum of components $A = \sum_{m=1}^{M} \theta_m R^{(m)}$. The transformation we are looking to learn is parameterized by θ.

For simplicity, we first show how to decompose A for learning a *piecewise constant* function. We define the following binary matrices

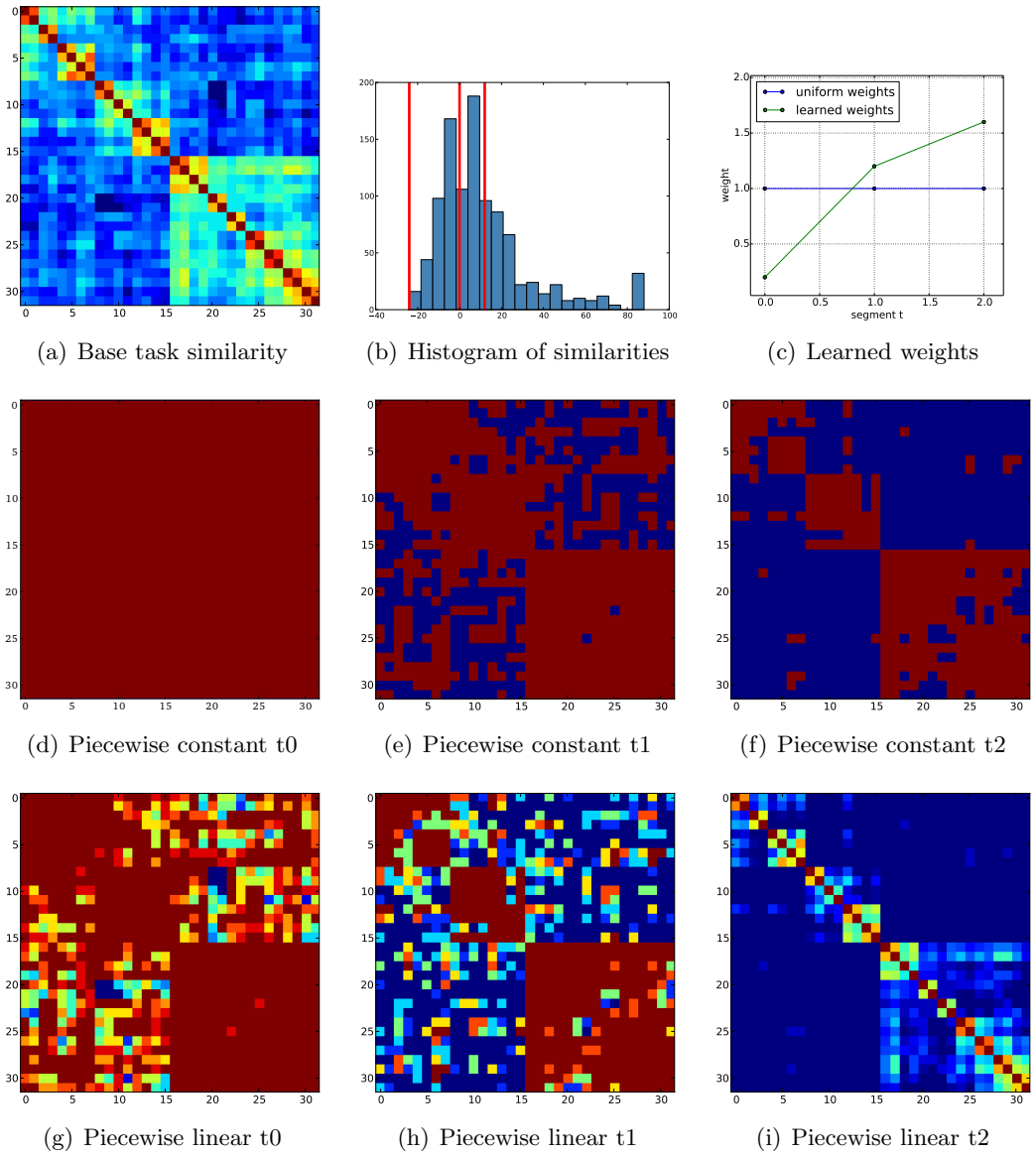
$$R_{s,t}^{(m)} = \begin{cases} 1 & \text{if } A_{s,t} \geq t_m \\ 0 & \text{else}\,, \end{cases} \tag{4.22}$$

where t_m are thresholds $1, \ldots, M$. These thresholds could be for instance chosen as percentiles of the distribution of similarities in A. An illustration of this is given in Figure 4.3, where the initial task similarity in Figure 4.3(a) is decomposed into three parts, Figures 4.3(d), 4.3(e), and 4.3(f), according to the 33rd and 66th percentile as shown in Figure 4.3(b).

For the piece-wise linear formulation, we define

$$\tilde{R}_{s,t}^{(m)} = \begin{cases} \min(A_{s,t} - t_m, t_{m+1} - t_m) & \text{if } A_{s,t} \geq t_m \\ 0 & \text{else}\,. \end{cases} \tag{4.23}$$

Again, this is illustrated in Figure 4.3, where the same initial task similarity in Figure 4.3(a) is decomposed into three *real-valued* parts as shown in Figures 4.3(g), 4.3(h), and 4.3(i). Using our framework, we now learn a weighting Θ of the individual components $\tilde{R}^m$, so we arrive at $Q^{-1} = \sum_{m=1}^{M} \theta_m \tilde{R}^{(m)}$. Learning the weights θ effectively gives rise to a piece-wise linear transformation.



(a) Base task similarity (b) Histogram of similarities (c) Learned weights

(d) Piecewise constant t0 (e) Piecewise constant t1 (f) Piecewise constant t2

(g) Piecewise linear t0 (h) Piecewise linear t1 (i) Piecewise linear t2

Figure 4.3: Learning a transformation of task similarity

4.2 Algorithms

In this section, we present efficient optimization algorithms to solve the primal and dual problems, i.e., Problems 4.1.1 and 4.1.2, respectively. We distinguish the cases of linear and non-linear kernel matrices. For non-linear kernels, we can simply use existing MKL implementations, while, for linear kernels, we develop a specifically tailored large-scale algorithm that allows us to train on problems with millions of data points and dimensions, as demonstrated on several data sets. We can even employ this algorithm

for non-linear kernels, if the kernel admits a sparse, efficiently computable feature representation. For example, this is the case for certain string kernels and polynomial kernels of degree 2 or 3. Our algorithms are embedded into the COFFIN framework [Sonnenburg and Franc, 2010] and integrated into the SHOGUN large-scale machine learning toolbox [Sonnenburg et al., 2010].

4.2.1 General Algorithms for Non-linear Kernels

A very convenient way to numerically solve the proposed framework is to simply exploit existing MKL implementations. To see this, recall from Section 4.1.4 that if we use the multi-task kernels $\tilde{k}_1, \ldots, \tilde{k}_M$ as defined in (4.5) as the set of multiple kernels, the completely dualized MKL formulation (see Problem 4.1.3) is given by,

$$\inf_{\boldsymbol{\theta}\in\Theta_p} \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n:\sum_{i=1}^n \alpha_i y_i = 0} -\frac{1}{2}\left\| \left(\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j \sum_{m=1}^M \theta_m \tilde{k}_m(x_i, x_j) \right)_{1\le m\le M} \right\|_{p^*} - C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C)\,.$$

An efficient optimization approach is by Vishwanathan et al. [2010], who optimize the completely dualized MKL formulation. This implementation comes along without a $\boldsymbol{\theta}$-step, but any of the α_i-steps computations of the α_i-steps are more costly as in the case of vanilla (MT-)SVMs.

Further, combining the partially dualized formulation in Problem 4.1.2 with the definition of multi-task kernels from (4.5), we arrive at an eqivalent problem to (4.7), that is,

$$\inf_{\boldsymbol{\theta}\in\Theta_p} \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n} -\frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j \sum_{m=1}^M \tilde{k}_m(x_i, x_j) \;-\; C\,\mathfrak{L}^*(-\boldsymbol{\alpha}/C)\,,$$

which is exactly the optimization problem of ℓ_p-norm multiple kernel learning as described in Kloft et al. [2009, 2011] and Kloft et al. [2010]. We may thus build on existing research in the field of MKL and use one of the prevalent efficient implementations to solve ℓ_p-norm MKL. Most of the ℓ_p-norm MKL solvers are specifically tailored to the hinge loss. Proven implementations are, for example, the interleaved optimization method of Kloft et al. [2011], which is directly integrated into the SVMLight module [Joachims, 1999] of the SHOGUN toolbox such that the $\boldsymbol{\theta}$-step is performed after each decomposition step, i.e., after solving the small QP occurring in SVMLight, which allows very fast convergence [Sonnenburg et al., 2006].

For an overview of MKL algorithms and their implementations, see the survey paper by Gönen and Alpaydin [2011].

4.2.2 A Large-scale Algorithm for Linear or String Kernels and Beyond

For specific kernels such as linear kernels and string kernels—and, more generally, any kernel admitting an efficient feature space representation—, we can derive a specifically tailored large-scale algorithm.

Overview

From a top-level view, the upcoming algorithm underlies the core idea of alternating the following two steps:

1. the $\boldsymbol{\theta}$ step, where the kernel weights are improved
2. the $\boldsymbol{W}$ step, where the remaining primal variables are improved.

Algorithm 4 (Blueprint of the large-scale optimization algorithm). The MKL module (θ step) is wrapped around the MTL module ($\boldsymbol{W}$ step).

1: **input:** data $x_1, \ldots, x_n \in \mathcal{X}$ and labels $y_1, \ldots, y_n \in \{-1, 1\}$ associated with tasks $\tau(1), \ldots, \tau(n) \in \{1, \ldots, T\}$; feature vectors $\phi_1(x_i), \ldots, \phi_M(x_i)$; task similarity matrices $Q_1, \ldots, Q_M$; optimization precision ε
2: initialize $\theta_m := \sqrt[p]{1/M}$ for all $m = 1, \ldots, M$, initialize $\boldsymbol{W} = \mathbf{0}$
3: **while** optimality conditions are not satisfied within tolerance ϵ **do**
4: $\quad$ $\boldsymbol{W}$ *descent step:* compute new $\boldsymbol{W}$ such that the obj. $\mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) + C\,\mathfrak{L}(\boldsymbol{W})$ decreases
5: $\quad\quad$ $\boldsymbol{W} := \operatorname{argmin}_{\widetilde{\boldsymbol{W}}} \; \mathfrak{R}_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{W}}) + C\,\mathfrak{L}(\widetilde{\boldsymbol{W}})$
6: $\quad$ $\boldsymbol{\theta}$ *step:* compute minimizer $\boldsymbol{\theta} := \operatorname{argmin}_{\widetilde{\boldsymbol{\theta}} \in \Theta_p} \; \mathfrak{R}_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{W}) + C\,\mathfrak{L}(\boldsymbol{W})$ according to (4.24)
7: **end while**
8: **output:** ϵ-accurate optimal hypothesis $\boldsymbol{W}$ and kernel weights $\boldsymbol{\theta}$

These steps are illustrated in Algorithm Table 4. We observe from the table that the variables are split into the two sets $\{\theta_m | m = 1, \ldots, M\}$ and $\{\mathbf{w}_{mt} | m = 1, \ldots, M, t = 1, \ldots, T\}$. The algorithm then alternatingly optimizes with respect to one or the other set until the optimality conditions are approximately satisfied. We will analyze convergence of this optimization scheme later in Section 4.2.3, establishing that Algorithm 4 constitutes a valid block coordinate algorithm. Note that similar algorithms have been used in the context of the group lasso and multiple kernel learning by Roth and Fischer [2008], Xu et al. [2010], and Kloft et al. [2011].

Solving the $\boldsymbol{\theta}$ Step

In this section, we discuss how to compute the update of the kernel weights $\boldsymbol{\theta}$ as carried out in Line 6 of Algorithm 4. Note that for fixed $\boldsymbol{W} \in \mathcal{H}$ it holds

$$\operatorname*{arginf}_{\boldsymbol{\theta} \in \Theta_p} \; \mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) + C\,\mathfrak{L}\big(A(\boldsymbol{W})\big) \;=\; \operatorname*{arginf}_{\boldsymbol{\theta} \in \Theta_p} \; \mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W})\,,$$

where $\mathfrak{R}_{\boldsymbol{\theta}}(\boldsymbol{W}) = \frac{1}{2}\sum_{m=1}^{M} \frac{\operatorname{tr}(W_m Q_m W_m)}{\theta_m}$. Furthermore, by Lagrangian duality,

$$\begin{aligned}
\inf_{\boldsymbol{\theta} \in \Theta_p} \; \frac{1}{2}\sum_{m=1}^{M} \frac{\operatorname{tr}(W_m Q_m W_m)}{\theta_m} \;&=\; \max_{\lambda \geq 0} \; \inf_{\boldsymbol{\theta} \succeq \mathbf{0}} \; \frac{1}{2}\sum_{m=1}^{M} \frac{\operatorname{tr}(W_m Q_m W_m)}{\theta_m} + \lambda \sum_{m=1}^{M} \theta_m^p \\
&=\; \inf_{\boldsymbol{\theta} \succeq \mathbf{0}} \; \underbrace{\frac{1}{2}\sum_{m=1}^{M} \frac{\operatorname{tr}(W_m Q_m W_m)}{\theta_m} + \lambda^* \sum_{m=1}^{M} \theta_m^p}_{=:\psi(\boldsymbol{\theta})}\,,
\end{aligned}$$

where we denote the optimal λ in the above maximization by $\lambda^\star$. The infimum is either attained at the boundary of the constraints or when $\nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}) = 0$, thus the optimal point $\boldsymbol{\theta}^\star$ satisfies $\theta_m^\star = (\operatorname{tr}(W_m Q_m W_m)/\lambda^\star)^{1/(p+1)}$ for any $m = 1, \ldots, M$. Because $\boldsymbol{\theta}^\star \in \Theta_p$, i.e., $\|\boldsymbol{\theta}\|_p = 1$, it follows $\lambda^\star = \left(\sum_{m=1}^M \operatorname{tr}(W_m Q_m W_m)^{p/(p+1)}\right)^{(p+1)/p}$, under the minimal assumption that $\boldsymbol{W} \neq \mathbf{0}$. Thus, because $\operatorname{tr}(W_m Q_m W_m) = \sum_{s,t=1}^T q_{mst} \langle \mathbf{w}_{ms}, \mathbf{w}_{mt} \rangle$,

$$\forall m = 1, \ldots, M: \quad \theta_m^\star = \frac{\sqrt[p+1]{\sum_{s,t=1}^T q_{mst} \langle \mathbf{w}_{ms}, \mathbf{w}_{mt} \rangle}}{\left(\sum_{m=1}^M \sqrt[p+1]{\sum_{s,t=1}^T q_{mst} \langle \mathbf{w}_{ms}, \mathbf{w}_{mt} \rangle}^p\right)^{1/p}}. \tag{4.24}$$

Solving the W Descent Step

To solve the W step as carried out in Line 4 of Algorithm 4, we consider the kernel weights $\{\theta_m | m = 1, \ldots, M\}$ as being fixed and optimize solely with respect to $\boldsymbol{W}$. In fact, we perform the $\boldsymbol{W}$ descent step in the dual, i.e., by optimizing the dual objective of Problem 4.1.2, i.e., solving

$$\sup_{\boldsymbol{\alpha} \in \mathbb{R}^n} \; - \mathfrak{R}_{\boldsymbol{\theta}}^*(A^*(\boldsymbol{\alpha})) \; - \; C \, \mathfrak{L}^*(-(\boldsymbol{\alpha})/C) \, .$$

Although our framework is also valid for other loss functions, for the presentation of the algorithm, we make a specific choice of a proven loss function, that is, the hinge loss $l(a) = \max(0, 1 - a)$, so that by (4.8), the above task becomes

$$\sup_{\boldsymbol{\alpha} \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha} \preceq \mathbf{C}} \; - \mathfrak{R}_{\boldsymbol{\theta}}^*(A^*(\boldsymbol{\alpha})) \; + \; \sum_{i=1}^n \alpha_i \, . \tag{4.25}$$

Our algorithm optimizes (4.25) by dual coordinate ascent, i.e., by optimizing the dual variables α_i one after another (i.e., only a single dual variable α_i is optimized at a time),

$$\sup_{d \in \mathbb{R} \colon 0 \leq \alpha_i + d \leq C} \; - \mathfrak{R}_{\boldsymbol{\theta}}^*(A^*(\boldsymbol{\alpha} + d\boldsymbol{e}_i)) \; + \; \sum_{i=1}^n \alpha_i + d \, ,$$

where we denote the unit vector of ith coordinate in $\mathbb{R}^n$ by $\boldsymbol{e}_i$. As we will see, this task can be performed analytically; however, performed purely in the dual involves computing a sum over all support vectors which is infeasible for large n. Our proposed algorithm is, instead, based on the application of the representer theorem carried out in Section 4.1.3: recall from (4.4) that, for all $m = 1, \ldots, M$ and $t = 1, \ldots, T$, it holds

$$\mathbf{w}_{mt} = \theta_m \sum_{i=1}^n q_{m\tau(i)t}^{(-1)} \alpha_i y_i \varphi_m(x_i) \, .$$

The core idea is to express the update of the α_i in the coordinate ascent procedure solely in terms of the vectors $\mathbf{w}_{mt}$. While optimizing the variables α_i one after another,

we keep track of the changes in the vectors $\mathbf{w}_{mt}$. This procedure is reminiscent of the *dual coordinate ascent* method, but differs in the way the objective is computed. Of course, this implies that we need to manipulate feature vectors, which explains why our approach relies on efficient infrastructure of storing and computing feature vectors and their inner products. If the infrastructure is adequate so that computing inner products in the feature space is more efficient than computing a row of the kernel matrix, our algorithm will have a substantial gain [Sonnenburg, 2008].

Expressing the update of a single variable α_i in terms of the vectors $\mathbf{w}_{mt}$ As argued above, our aim is to express the (analytical) computation of

$$\sup_{d\in\mathbb{R}:\, 0\leq\alpha_i+d\leq C} \quad -\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha}+d\boldsymbol{e}_i)) \;+\; \sum_{i=1}^n \alpha_i + d\,.$$

solely in terms of the vectors $\mathbf{w}_{mt}$. To start the derivation, note that, by (4.3),

$$\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha}+d\boldsymbol{e}_i)) \;=\; \frac{1}{2}\sum_{m=1}^M \theta_m \left\|A^*(\boldsymbol{\alpha}+d\boldsymbol{e}_i)\right\|^2_{Q_m^{-1}}$$

with, by (4.6),

$$\left\|A^*(\boldsymbol{\alpha}+d\boldsymbol{e}_i)\right\|^2_{Q_m^{-1}} \;=\; \sum_{j,\tilde{j}=1}^n \alpha_j\alpha_{\tilde{j}}y_jy_{\tilde{j}}\tilde{k}_m(x_j,x_{\tilde{j}}) + 2dy_i\sum_{j=1}^n \alpha_jy_j\tilde{k}_m(x_i,x_j) + d^2k_m(x_i,x_i)\,,$$

where

$$\tilde{k}_m(x_i,x_j) \;=\; q^{(-1)}_{m\tau(i)\tau(j)}k_m(x_i,x_j)$$

is the mth multi-task kernel as defined in (4.5). Thus,

$$\begin{aligned}
&\operatorname*{argsup}_{d\in\mathbb{R}:\, 0\leq\alpha_i+d\leq C} \quad -\mathfrak{R}^*_{\boldsymbol{\theta}}(A^*(\boldsymbol{\alpha}+d\boldsymbol{e}_i)) \;+\; \sum_{i=1}^n \alpha_i + d \\
=\;&\operatorname*{argsup}_{d\in\mathbb{R}:\, 0\leq\alpha_i+d\leq C} \quad d \;-\; dy_i\sum_{j=1}^n \alpha_jy_j\left(\textstyle\sum_{m=1}^M \theta_m\tilde{k}_m(x_i,x_j)\right) \;-\; \frac{1}{2}d^2\left(\textstyle\sum_{m=1}^M \theta_m\tilde{k}_m(x_i,x_i)\right) \\
=\;&\operatorname*{argsup}_{d\in\mathbb{R}:\, 0\leq\alpha_i+d\leq C} \quad d \;-\; \underbrace{dy_i\left(\textstyle\sum_{m=1}^M \left\langle \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\right\rangle\right) \;-\; \frac{1}{2}d^2\left(\textstyle\sum_{m=1}^M \theta_m\tilde{k}_m(x_i,x_i)\right)}_{=:\psi(d)}\,.
\end{aligned}$$

The optimum of $\psi(d)$ is either attained at the boundaries of the constraint $0 \leq \alpha_i + d \leq C$ or when $\psi'(d) = 0$. Hence, the optimal $d^\star$ can be expressed analytically as

$$d^\star = \max\left(-\alpha_i\,,\;\min\left(C-\alpha_i\,,\;\frac{1-y_i\sum_{m=1}^M \theta_m\left\langle \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\right\rangle}{\sum_{m=1}^M \theta_m\tilde{k}_m(x_i,x_i)}\right)\right)\,. \tag{4.26}$$

Whenever we update an α_i according to

$$\alpha_i^{\text{new}} := \alpha_i^{\text{old}} + d^\star$$

with d computed as in (4.26), we need to also update the vectors $\mathbf{w}_{mt}$, $m = 1, \ldots, M$, $t = 1, \ldots, T$, according to

$$\mathbf{w}_{mt}^{\text{new}} := \mathbf{w}_{mt}^{\text{old}} + d\theta_m q_{m\tau(i)t}^{(-1)} y_i \varphi_m(x_i)\,, \tag{4.27}$$

to be consistent with (4.4). Similarly, we need to update the vectors $\mathbf{w}_{mt}$ after each θ step according to

$$\mathbf{w}_{mt}^{\text{new}} := \left(\theta_m^{\text{new}}/\theta_m^{\text{old}}\right) \mathbf{w}_{mt}^{\text{old}}\,. \tag{4.28}$$

To avoid recurrences in the iterates, a $\boldsymbol{\theta}$-step should only be performed if the primal objective has decreased between subsequent $\boldsymbol{\theta}$-steps. Thus, after each $\boldsymbol{\alpha}$ epoch, the primal objective needs to be computed in terms of $\boldsymbol{W}$. As described above, the algorithm keeps $\boldsymbol{W}$ up to date when $\boldsymbol{\alpha}$ changes, which makes this task particular simple.

The resulting large-scale algorithm is summarized in Algorithm Table 5. Data and the labels are input to the algorithm as well as a sub-procedure for efficient computation of feature maps (cf. Section 2.4.2). Lines 2 and 3 initialize the optimization variables. In Line 4 the inverses of the task similarity matrices are pre-computed. Algorithm 5 iterates over Lines 7–16 until the stopping criterion falls under a pre-defined accuracy threshold ε. In Lines 7–11 the line search is computed for all dual variables. Lines 14 and 15 update the primal variables and kernel weights to be consistent with the representer theorem, only if the primal objective has decreased since the last $\boldsymbol{\theta}$-step. We stop Algorithm 5 when the relative change in the objective o is less than ϵ. Notice that we do not optimize the W step to full precision, but instead alternate between one pass over the α_i and a $\boldsymbol{\theta}$ step (see Section 4.2.3 for an analysis of convergence).

Details on the Implementation

We have implemented the optimization algorithms described in the previous section into the general framework of the SHOGUN machine learning toolbox [Sonnenburg et al., 2010]. Besides the described implementations for binary classification, we also provide implementations for novelty detection and regression. Furthermore, the user may choose an optimization scheme, that is, decide whether one of the classic, non-linear MKL solvers shall be used (either the analytic optimization algorithm of Kloft et al. [2011], the cutting plane method of Sonnenburg et al. [2006], or the Newton algorithm by Kloft et al. [2009]), or the novel implementation for efficiently computable feature maps.

In the more conventional family of approaches, the *wrapper algorithms*, an optimization scheme on $\boldsymbol{\theta}$ wraps around a conventional SVM solver (e.g., LIBSVM and SVMLIGHT are integrated into SHOGUN) using a single multi-task kernel. Effectively, this results in alternatingly solving for $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$. For the $\boldsymbol{\theta}$-step, SHOGUN offers the three

Algorithm 5 (Dual-coordinate-ascent-based MTL training algorithm). Generalization of the LibLinear training algorithm to multiple tasks and multiple linear kernels.

input: data $x_1, \ldots, x_n \in \mathcal{X}$ and labels $y_1, \ldots, y_n \in \{-1, 1\}$ associated with tasks $\tau(1), \ldots, \tau(n) \in \{1, \ldots, T\}$; efficiently computable feature maps $\varphi_1, \ldots, \varphi_M$; task similarity matrices $Q_1, \ldots, Q_M$; optimization precision ε
for all $i \in \{1, \ldots, n\}$ initialize $\alpha_i = 0$
for all $m \in \{1, \ldots, M\}$ and $t \in \{1, \ldots, T\}$, initialize $\mathbf{w}_{mt}$ according to (4.4)
for all $m \in \{1, \ldots, M\}$, compute inverse $Q_m^{-1} = \left(q_{mst}^{(-1)}\right)_{1 \leq s,t \leq T}$
initialize primal objective $o = nC$
while optimality conditions are not satisfied **do**
    for all $i \in \{1, \ldots, n\}$
        compute d according to (4.26)
        update $\alpha_i := \alpha_i + d$
        for all $m \in \{1, \ldots, M\}$ and $t \in \{1, \ldots, T\}$, update $\mathbf{w}_{mt}$ according to (4.27)
    end for
    store primal objective $o^{\text{old}} = o$ and compute new primal objective o
    if primal objective has decreased, i.e., $o < o^{\text{old}}$
        for all $m \in \{1, \ldots, M\}$, compute θ_m from $\mathbf{w}_{m1}, \ldots, \mathbf{w}_{mT}$ according to (4.24)
        for all $m \in \{1, \ldots, M\}$ and $t \in \{1, \ldots, T\}$, update $\mathbf{w}_{mt}$ according to (4.28)
    end if
end while
output: ϵ-accurate optimal hypothesis $\boldsymbol{W} = (\mathbf{w}_{mt})_{1 \leq m \leq M, 1 \leq t \leq T}$ and kernel weights $\boldsymbol{\theta} = (\theta_m)_{1 \leq m \leq M}$

choices listed above. The second, much faster approach performs interleaved optimization and thus requires modification of the core SVM optimization algorithm. This is currently either integrated into the chunking-based SVRlight and SVMlight module. Lastly, the completely new optimization scheme as described in Algorithm Table 5 is implemented and connected with the module for computing the $\boldsymbol{\theta}$-step.

Note that the implementations for non-linear kernels come with the option of either pre-computing the kernel or computing the kernel on the fly for large-scale data sets. For truly large-scale MT-MKL, a linear or string kernel should be used. This is implemented as an internal interface the COFFIN module of SHOGUN Sonnenburg and Franc [2010], which we have described in Section 2.4.2.

4.2.3 Convergence Analysis

In this section, we establish convergence of Algorithm 4 under mild assumptions. To this end, we build on the existing theory of convergence of the block coordinate descent method. Classical results usually assume that the function to be optimized is strictly convex and continuously differentiable. This assertion is frequently violated in machine learning when, e.g., the hinge loss is employed. In contrast, we base our convergence analysis on the work of Tseng [2001] concerning the convergence of the block coordinate descent method. The following proposition is a direct consequence of Lemma 3.1 and

Theorem 4.1 in Tseng [2001].

Proposition 4.2.1. *Let $f : \mathbb{R}^{d_1+\cdots+d_R} \to \mathbb{R} \cup \{\infty\}$ be a function. Put $d = d_1 + \cdots + d_R$. Suppose that f can be decomposed into $f(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r) = f_0(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r) + \sum_{r=1}^{R} f_r(\boldsymbol{a}_r)$ for some $f_0 : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ and $f_r : \mathbb{R}^{d_r} \to \mathbb{R} \cup \{\infty\}$, $r = 1, \ldots, R$. Initialize the block coordinate descent method by $\boldsymbol{a}^0 = (\boldsymbol{a}_1^0, \ldots, \boldsymbol{a}_R^0)$. Let $(r_k)_{k\in\mathbb{N}} \subset \{1, \ldots, R\}$ be a sequence of coordinate blocks. Define the iterates $\boldsymbol{a}^k = (\boldsymbol{a}_1^k, \ldots, \boldsymbol{a}_R^k)$, $k > 0$, by*

$$\boldsymbol{a}_{r_k}^{k+1} \in \operatorname*{argmin}_{\mathfrak{A}\in\mathbb{R}^{d_{r_k}}} f\big(\boldsymbol{a}_1^{k+1}, \cdots, \boldsymbol{a}_{r_k-1}^{k+1}, \mathfrak{A}, \boldsymbol{a}_{r_k+1}^{k}, \cdots, \boldsymbol{a}_R^k\big)\,, \quad \boldsymbol{a}_r^{k+1} := \boldsymbol{a}_r^k\,, \quad r \neq r_k\,, \quad k \in \mathbb{N}_0\,. \tag{4.29}$$

Assume that

(A1) f is convex and proper (i.e., $f \not\equiv \infty$)

(A2) the sublevel set $\mathcal{A}^0 := \{\boldsymbol{a} \in \mathbb{R}^d : f(\boldsymbol{a}) \leq f(\boldsymbol{a}^0)\}$ is compact and f is continuous on $\mathcal{A}^0$
*(*ASSURES EXISTENCE OF MINIMIZER IN (4.29)*)*

(A3) $\operatorname{dom}(f_0) := \{a \in \mathbb{R}^d : f_0(\boldsymbol{a}) < \infty\}$ is open and f_0 is Gâteaux differentiable (e.g., continuously differentiable) on $\operatorname{dom}(f_0)$
*(*YIELDS REGULARITY—I.E., ANY COORDINATE-WISE MINIMUM IS A MINIMUM OF f*)*

(A4) it exists a number $T \in \mathbb{N}$ so that, for each $k \in \mathbb{N}$ and $r \in \{1, \ldots, R\}$, there is $\tilde{k} \in \{k, \ldots, k + T\}$ with $r_{\tilde{k}} = r$.
*(*ENSURES THAT EACH COORDINATE BLOCK IS OPTIMIZED "SUFFICIENTLY OFTEN"*)*

Then the minimizer in (4.29) exists and any cluster point of the sequence $(\boldsymbol{a}^k)_{k\in\mathbb{N}}$ minimizes f over $\mathcal{A}$.

Corollary 4.2.2. *Assume that*

(B1) the data is represented by $\phi_m(x_i) \in \mathbb{R}^{e_m}$, $i = 1, \ldots, n$, $e_m < \infty$, $m = 1, \ldots, M$.

(B2) the loss function l is convex, finite in 0, and continuous on its domain $\operatorname{dom}(l)$

(B3) the task similarity matrices $Q_1, \ldots, Q_T$ are positive definite

(B3) any iterate $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ traversed by Algorithm 4 has $\theta_m > 0$, $m = 1, \ldots, M$

(B4) the exact search specified in Line 5 of Algorithm 4 is performed

Then Algorithm 4 is well-defined and any cluster point of the sequence traversed by the Algorithm 4 is a minimal point of Problem 4.1.1.

Proof The corollary is obtained by applying Proposition 4.2.1 to Problem 4.1.1, that is,

$$\inf_{\boldsymbol{W},\boldsymbol{\theta}\colon\, \boldsymbol{\theta}\in\Theta_p} \; \frac{1}{2}\sum_{m=1}^{M} \frac{\|W_m\|_{Q_m}^2}{\theta_m} \; + \; C\sum_{i=1}^{n} l\Big(y_i \sum_{m=1}^{M} \big\langle\, \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\big\rangle\Big)\,, \tag{4.30}$$

where $\Theta_p = \{\boldsymbol{\theta} \in \mathbb{R}^M : \theta_m \geq 0, m = 1,\dots,M, \|\boldsymbol{\theta}\|_p \leq 1\}$ and, by (B1), $\boldsymbol{W} \in \mathbb{R}^{eT}$, $e = e_1 + \cdots + e_M$. Note that (4.30) can be written unconstrained as

$$\inf_{\boldsymbol{W},\boldsymbol{\theta}} \; f(\boldsymbol{W},\boldsymbol{\theta})\,, \quad \text{where} \;\; f(\boldsymbol{W},\boldsymbol{\theta}) \;:=\; f_0(\boldsymbol{W},\boldsymbol{\theta}) + f_1(\boldsymbol{W}) + f_2(\boldsymbol{\theta})\,, \tag{4.31}$$

by putting

$$f_0(\boldsymbol{W},\boldsymbol{\theta}) \;:=\; \frac{1}{2}\sum_{m=1}^{M} \frac{\|W_m\|_{Q_m}^2}{\theta_m} \; + \; I_{\{\boldsymbol{\theta}\succ\mathbf{0}\}}(\boldsymbol{\theta})$$

as well as

$$f_1(\boldsymbol{W}) \;:=\; C\sum_{i=1}^{n} l\Big(y_i \sum_{m=1}^{M} \big\langle\, \mathbf{w}_{m\tau(i)}, \varphi_m(x_i)\big\rangle\Big)\,, \quad f_2(\boldsymbol{\theta}) \;:=\; I_{\{\|\boldsymbol{\theta}\|_p\leq 1\}}(\boldsymbol{\theta})\,, \tag{4.32}$$

where I is the indicator function, $I_S(s) = 0$ if $s \in S$ and $I_S(s) = \infty$ elsewise. Note that we use the shorthand $\boldsymbol{\theta} \succ \mathbf{0}$ for $\theta_m > 0$, $m = 1,\dots,M$.

Assumption (B4) ensures that applying the block coordinate descent method to (4.30) and (4.31) problems yields precisely the sequence of iterates. Thus, in order to prove the corollary, it suffices to validate that (4.31) fulfills Assumptions (A1)–(A4) in Proposition 4.2.1.

Validity of (A1) Recall that Algorithm 4 is initialized with $\boldsymbol{W}^0 = \mathbf{0}$ and $\theta_m^0 = \sqrt[p]{1/M}$, $m = 1,\dots,M$, so it holds

$$f(\boldsymbol{W}^0,\boldsymbol{\theta}^0) \;=\; \underbrace{f_0(\boldsymbol{W}^0,\boldsymbol{\theta}^0)}_{=0} \; + \underbrace{f_1(\boldsymbol{W}^0)}_{=Cn\,l(0)} + \; \underbrace{f_2(\boldsymbol{\theta}^0)}_{=0} \;=\; Cn\,l(0) \;<\; \infty\,, \tag{4.33}$$

hence $f \not\equiv \infty$, so f is proper. Furthermore, $\mathrm{dom}(f_0) = \{(\boldsymbol{W},\boldsymbol{\theta}) : \boldsymbol{\theta} \succ \mathbf{0}\}$ is convex, and f_0 is convex on $\mathrm{dom}(f_0)$, so f_0 is a convex function. By (B2), the loss function l is convex, so f_1 is a convex function. The domain $\mathrm{dom}(f_2) = \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\|_p \leq 1\}$ is convex, and $f_2 \equiv 0$ on its domain, so f_2 is a convex function. Thus the sum $f = f_0 + f_1 + f_2$ is a convex function, which shows (A1).

Validity of (A2) Let $(\widetilde{\boldsymbol{W}},\widetilde{\boldsymbol{\theta}}) \in \mathcal{A}^0 := \{(\boldsymbol{W},\boldsymbol{\theta}) : f(\boldsymbol{W},\boldsymbol{\theta}) \leq f(\boldsymbol{W}^0,\boldsymbol{\theta}^0)\}$. We have $f_0, f_1, f_2 \geq 0$, so, for all $m = 1,\dots,M$,

$$\begin{aligned} \frac{\|\widetilde{W}_m\|_{Q_m}}{2\theta_m} \;&\leq\; f_0(\widetilde{\boldsymbol{W}},\widetilde{\boldsymbol{\theta}}) \;\leq\; f_0(\widetilde{\boldsymbol{W}},\widetilde{\boldsymbol{\theta}}) + \underbrace{f_1(\widetilde{\boldsymbol{W}})}_{\geq 0} + \underbrace{f_2(\widetilde{\boldsymbol{\theta}})}_{\geq 0} \overset{\text{by (4.31)}}{=} f(\widetilde{\boldsymbol{W}},\widetilde{\boldsymbol{\theta}}) \\ &\leq\; f(\boldsymbol{W}^0,\boldsymbol{\theta}^0) \overset{\text{by (4.33)}}{\leq} Cn\,l(0)\,, \end{aligned} \tag{4.34}$$

which implies $\|\widetilde{W}_m\|^2_{Q_m} \leq 2\theta_m Cn\, l(0)$. Similar, because $f_0 \geq 0$, we have $f_2(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{\theta}}) \leq Cn\, l(0) < \infty$, which, by (4.32), implies $\|\widetilde{\boldsymbol{\theta}}\|_p \leq 1$ and thus $\widetilde{\theta}_m \leq 1$, $m = 1, \ldots, M$. Hence, by (4.34), $\|\widetilde{W}_m\|^2_{Q_m} \leq 2Cn\, l(0)$, $m = 1, \ldots, M$. Because $Q_1, \ldots, Q_M$ are positive definite, $\nu := \min_{m=1,\ldots,M} \operatorname{tr}(Q_m) > 0$. Thus, for any $m = 1, \ldots, M$,

$$\begin{aligned}\|\widetilde{W}_m\|^2 &= \operatorname{tr}(\widetilde{W}_m^* \widetilde{W}_m) = \operatorname{tr}(\widetilde{W}_m^* \widetilde{W}_m) \operatorname{tr}(Q_m) / \operatorname{tr}(Q_m) \leq \operatorname{tr}(\widetilde{W}_m^* W_m Q_m) / \operatorname{tr}(Q_m) \\ &\leq \nu^{-1} \operatorname{tr}(\widetilde{W}_m^* \widetilde{W}_m Q_m) = \nu^{-1} \operatorname{tr}(\widetilde{W}_m Q_m \widetilde{W}_m^*) = \nu^{-1} \|\widetilde{W}_m\|^2_{Q_m} \leq 2\nu^{-1} Cn\, l(0)\,.\end{aligned}$$

Thus

$$\|(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{\theta}})\|^2 = \|\widetilde{\boldsymbol{W}}\|^2 + \|\widetilde{\boldsymbol{\theta}}\|^2 \leq 2\nu^{-1} CMn\, l(0) + M < \infty\,.$$

Thus $\sup_{(\widetilde{W}, \widetilde{\boldsymbol{\theta}}) \in \mathcal{A}^0} \|(\widetilde{W}, \widetilde{\boldsymbol{\theta}})\| < \infty$, which shows that $\mathcal{A}^0$ is bounded. Furthermore, $\mathcal{A}^0 \subset \operatorname{dom}(f) = \operatorname{dom}(f_0) \cap \operatorname{dom}(f_1) \cap \operatorname{dom}(f_2)$ and f_0, f_1, f_2 are continuous on their respective domains. Thus f is continuous on $\operatorname{dom}(f)$ and thus also on its subset $\mathcal{A}^0$. It holds $\mathcal{A}^0 = f^{-1}\big(]-\infty, f(\boldsymbol{W}^0, \boldsymbol{\theta}^0)]\big)$, i.e., $\mathcal{A}^0$ is the preimage of closed set under a continuous function; thus $\mathcal{A}^0$ is closed. Any closed and bounded subset of $\mathbb{R}^d$ is compact. Thus $\mathcal{A}^0$ is compact, which was to show.

Validity of (A3) and (A4) Clearly, $\operatorname{dom}(f_0) = \{(\boldsymbol{W}, \boldsymbol{\theta}) : \boldsymbol{\theta} \succ \mathbf{0}\}$ is open and f_0 is continuously differentiable on $\operatorname{dom}(f_0)$. Thus it is Gâteaux differentiable on $\operatorname{dom}(f_0)$. Finally, assumption (A4) is trivially fulfilled as Algorithm 4 employs a simple alternating rule for traversing the blocks of coordinates.

In summary, Proposition 4.2.1 can thus be applied to Problem 4.1.1, which yields the claim of the corollary. ■

4.3 Applications

In this section, we present a range of experiments in different application domains to illuminate various aspects and special cases of the presented framework, summarizing the work that was done using the described machinery. The main points we aim to show in this chapter are, 1) successful applications of graph-regularized multitask learning 2) the speed-up we achieved by our solver 3) the validity of the extensions that we derived from our general framework. For this, we will discuss the following experiments:

- **Graph-regularized 3D Shape Reconstruction**: By combining graph-regularized MTL with a robust loss function from support vector regression, we develop a practical tool for high-throughput analysis of $3D$ fluorescence microscopy data.
- **Joint 2D/3D Model Learning**: In a second application to bioimaging, we show how a *Clustered MTL formulation* allows the combination of $2D$ and $3D$ images to save labeling cost.

- **Runtime Experiments**: We compare the runtime of our solver with traditional methods on a range of data sets.
- **MHC-I binding prediction**: We show a successful application of graph-regularized multitask learning to a highly relevant problem for developing immunotherapies.
- **Learning the Similarity**: We assess the validity of the extensions we developed from our general framework on synthetic data, as well as on biological sequences.

4.3.1 Graph-regularized 3D Shape Reconstruction

To demonstrate the applicability of the graph-regularization framework, we present an application to a real-world problem from fluorescence microscopy. In close collaboration with biologists, we have developed a computational pipeline [Widmer et al., 2013a] to automatically segment nuclei in $3D$ microscopy data. The pipeline was successfully used for high-throughput image analysis [Heinrich et al., 2013]. Our method is an adaptation of graph-regularized multitask learning [Evgeniou et al., 2005], which we have shown to be a special case of our general framework in Section 4.1.5. As a side-effect, the set of experiments presented in this section provide a visual illustration of graph-regularization. We combine a MTL regularizer with a robust loss function (see Table 2.1) as used in support vector regression to be resilient to image contaminations and therefore minimize the need for manual post-processing.

Background Imaging data, such as those from microscopic experiments, is a unique source of information in biology. Through fluorescent staining, they enable the investigation of tissue composition, cell shapes and also sub-cellular localization. A challenge, however, is that manual and consistent measurements of such data is still time consuming and this remains an obstacle in large scale experiments. Methods that assist in processing such complex, large data are therefore needed. These methods should not only speed up these measurements steps but also increase the reproducibility of the measurements. In this project, we focus on the challenge of detecting cell nuclei from fluorescent microscopy images. In fluorescence microscopy, it is common practice for biologists to manually segment cells based on 3D visualization and then later quantify the signal within this segmentation (usually in a different staining channel). In particular, we will address nuclear segmentation for anisotropic and highly noisy 3D microscopic images with possible staining defects – a very challenging problem that cannot be handled robustly by conventional computer vision methods such as blob detection, deformable model (e.g. level set) or combinational optimization (e.g. graph cut), because the staining defects normally lead to missing intensities within the body of the true nucleus. A robust approach, therefore, is highly desired.

Though microscopes are sometimes equipped with software to assist researchers on this task, more often than not, the existing software only provides very rough polygon fits based on intensity values. A major drawback of these software solutions is that 3D

information is not taken into account, which means segmentations are performed for each layer individually. Furthermore, each object in a bigger volume (with potentially hundreds of cells) has to be processed separately, making this step a major bottleneck. Finally, any prior knowledge about the structure of the objects of interest is ignored as fits are usually non-parametric. This is sub-optimal when segmenting cells or nuclei, as these objects have known structure that can be exploited. Due to these drawbacks, signal quantification is an extremely time consuming task, and truly large scale quantification experiments become prohibitive. We propose a new method that addresses these shortcomings. It can be applied to images containing multiple cells and exploits the fact that nuclei commonly have an ellipsoid shape. The method adapts graph-regularized transfer learning Evgeniou et al. [2005] to the problem of parametric fitting in several layers in combination with a robust loss function, as used in support vector regression, to minimize the need for manual post-processing.

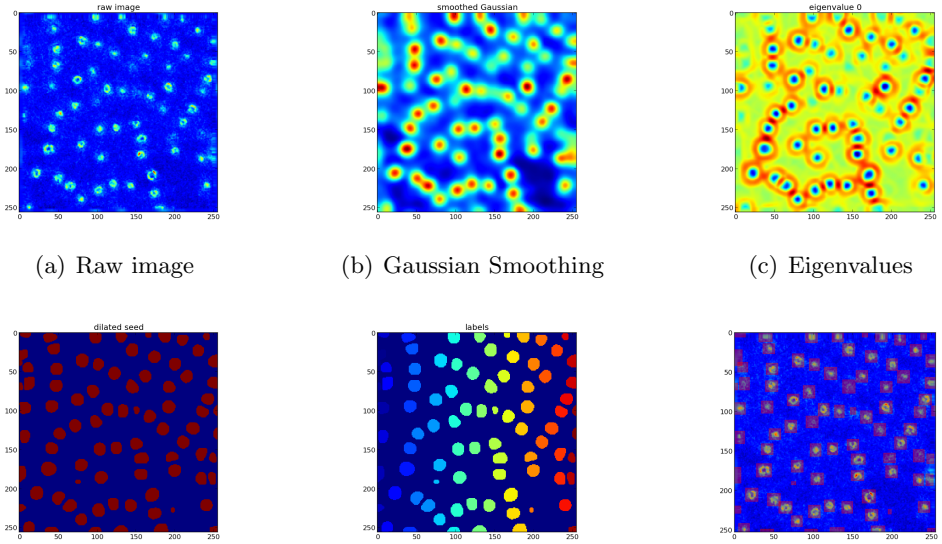
Computational Pipeline

Our proposed method provides biologists with a tool for high-throughput quantification experiments. We achieve this by first automatically detecting individual cells or nuclei in a larger volume and then fitting parametric objects to these sub-volumes that combine information from several layers. The preprocessing is step based on the recently published multi-scale hessian eigenvalue thresholding [Lou et al., 2012a]. As illustrated in Figure 4.4, the preprocessing step will result in a set of sub-volumes that are subsequently processed by our fitting procedure. The fitting procedure fits a stack of ellipses to each sub-volume, one ellipse for each layer, while trying to be robust to image contamination and maximizing smoothness between neighboring layers.

Graph-regularization Clearly, parametric objects in individual z-layers should not be fitted independently; instead, the transition from one layer to the next should be smooth. An illustration of the benefit of this is shown in Figure 4.5. We propose to enforce smoothness using a graph-regularizer as defined in Equation 4.16, treating different z-layers in the volume as different tasks. We define the corresponding adjacency matrix as

$$A_{i,j} = \begin{cases} 1, \text{ if } |i-j| = 1 \\ \frac{1}{2}, \text{ if } |i-j| = 2 \\ 0, \text{ else.} \end{cases}$$

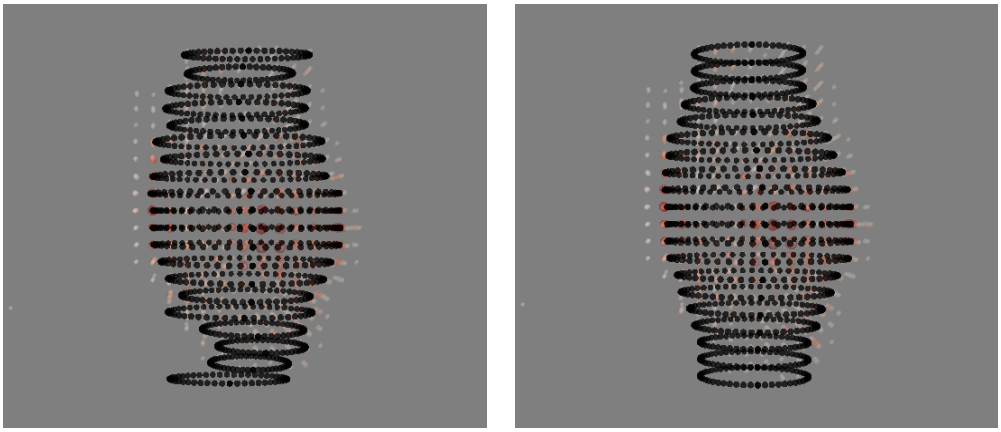
A robust loss Our method is optimized for the detection of cell nuclei, which are membrane enclosed organelles in eukaryotic cells that contain most of the cell's genetic material. The shape of these objects resembles a deformed ellipsoid. We argue that we can incorporate this prior knowledge about the shape by fitting parametric geometric objects such as an $3D$ ellipsoid or stack of $2D$ ellipses. An example how this could be beneficial in providing robust fits in the face of missing data points can be seen in Figure 4.6.



(a) Raw image (b) Gaussian Smoothing (c) Eigenvalues

(d) Seeds (e) Labeled Seeds (f) Boxes

Figure 4.4: In a preprocessing step, we first identify individual cells using Hessian eigenvalue thresholding as described in [Lou et al., 2012a].

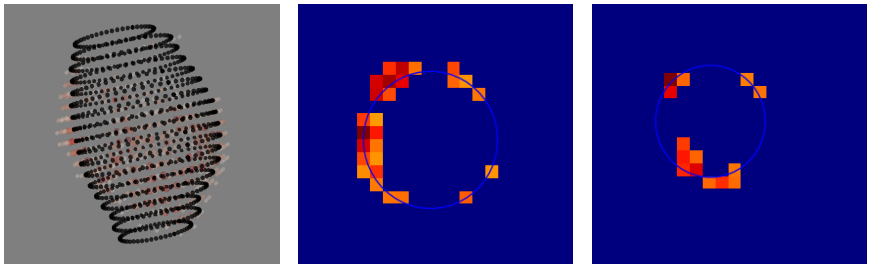


(a) No regularization (b) Graph regularization

Figure 4.5: Smoothness

An ellipse in $2D$ is parametrized by a center point $c = [c_x, c_y]^\mathsf{T}$ and two radii $\boldsymbol{r} = [r_x, r_y]^\mathsf{T}$. The points $[x, y]^\mathsf{T}$ on the ellipse centered at c are described by the equation

$$\frac{(x - c_x)^2}{r_x^2} + \frac{(y - c_y)^2}{r_y^2} = 1.$$



(a) Parametric fit (b) Layer 15 fit (c) Layer 20 fit

Figure 4.6: The case for using parametric objects

An alternate parameterization (general conic) is given by

$$ax^2 + bxy + cy^2 + dx + ey + f = 0,$$

describing an ellipse if $b^2 - 4ac < 0$ Rosin [1996]. Let

$$\boldsymbol{x} = [x^2, xy, y^2, x, y, 1],^{\mathsf{T}} \qquad \Theta = [a, b, c, d, e, f]^{\mathsf{T}},$$

then be points on the ellipse that satisfy $\boldsymbol{x}^{\mathsf{T}}\Theta = 0$. The algebraic distance f of a point $\boldsymbol{x}$ to the ellipse parametrized by Θ is defined as:

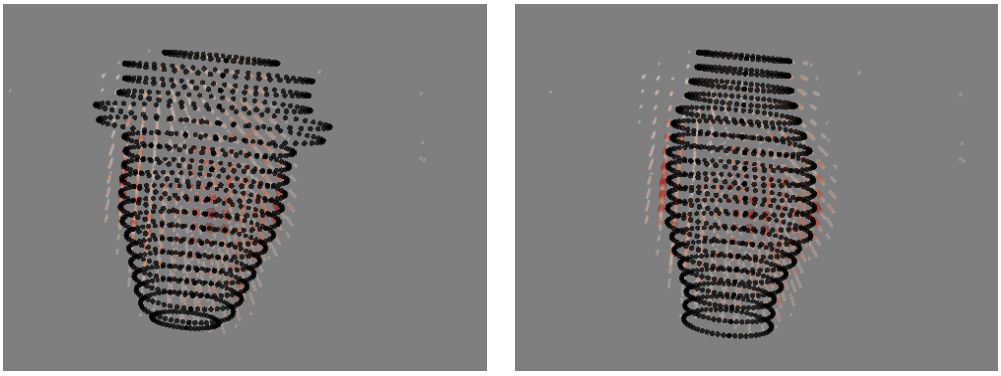
$$f(\boldsymbol{x}, \Theta) = \boldsymbol{x}^{\mathsf{T}}\Theta. \tag{4.35}$$

The algebraic distance is an approximation of the Euclidean distance that has the advantage that it is much easier to compute. Using this parameterization, the fitting of an ellipse may be formulated as minimizing the convex function of algebraic distance (see Equation 4.35), leading to the following formulation:

$$\min_{\Theta} \sum_{i=1}^{N} L(\Theta^{\mathsf{T}}\boldsymbol{x}_i) \tag{4.36}$$
$$\text{s.t. solution non-degenerate,}$$

where L is a convex loss function. Additional constraints are necessary to avoid degenerate solutions. In order to avoid the trivial solution $\Theta = 0$ and recognizing that any multiple of a solution Θ represents the same conic, the parameter vector Θ is constrained in one way or another [Fitzgibbon et al., 1999]. Many authors suggest $\|a\|^2 = 1$, others $a + c = 1$ or $f = 1$ [Fitzgibbon et al., 1999].

It is well established that the squared loss is particularly prone to outliers, as distance is penalized quadratically. An example of this sensitivity to outliers is shown in Figure 4.7, where a few outliers are sufficient to considerably distort the fit. We therefore propose to use the ε-insensitive loss function for the problem at hand. The ε-insensitive loss has its background in the context of support vector regression [Schölkopf



(a) Squared Loss (b) Robust Loss

Figure 4.7: A severe effect of a few outliers for the squared loss is shown in (a). For the same data set, the ε-insensitive loss achieves a more robust solution as shown in (b).

and Smola, 2002; Vapnik, 1995]. It is also known as dead-zone penalty in other contexts [Boyd, S.P. and Vandenberghe, 2004] and is often used when a more robust error function is needed.

$$L_\varepsilon(r) = \begin{cases} |r| - \varepsilon, & \text{if } |r| > \varepsilon \\ 0 & \text{else.} \end{cases} \tag{4.37}$$

Two important properties make it appealing for the problem at hand. First, it does not penalize points that are within a rim of the stacked ellipsoid. This captures the intuition that the nuclear membrane has a certain thickness and we therefore do not want to penalize points that are within the membrane. Second, the loss is affine (linear minus some offset that depends on ε). This means that outliers are not penalized as severely as with the squared loss, yielding a more robust error function. Putting it all together yields the formulation

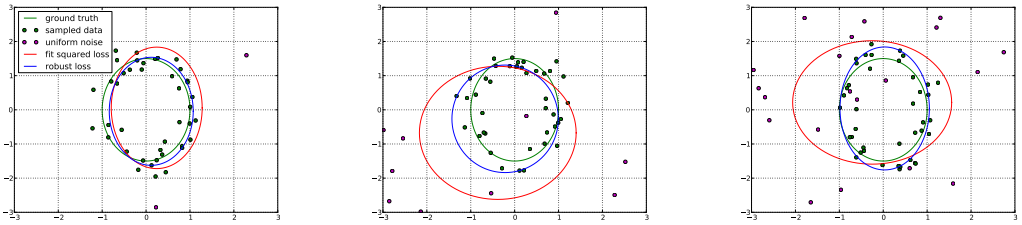
$$\begin{aligned} \min_{\Theta_1,\ldots,\Theta_Z} \; & \sum_{s=1}^{Z}\sum_{t=1}^{Z} A_{s,t}\|\Theta_s - \Theta_t\|^2 + C\sum_{t=1}^{Z}\sum_{i=1}^{N_t} L_{eps}(\Theta_z^\mathsf{T}\boldsymbol{x}_i) \\ \text{s.t. } & \text{solution non-degenerate,} \end{aligned} \tag{4.38}$$

which is a special case of the general framework presented in Section 4.1.

Evaluation

Robustness Analysis To compare robust loss and squared loss in the context of parametric fits, we set up an experiment using synthetic data. For this, we first sampled n points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ from an ellipse parametrized by Θ. Next, uniformly distributed points were sampled in the interval $[-3, 3]$ to simulate random noise and contaminations. Based on all sampled points, two fits were obtained: one using the squared loss and one using the robust loss. Examples of these fits are shown in 4.8(a), 4.8(b) and

4.8(c). A systematic comparison of the two losses is shown in Figure 4.9, where the error with respect to the ground truth (i.e., $\|\Theta_{fit} - \Theta\|$) is shown as a function of the number of uniformly sampled points. We observe that the error increases much later when using the robust loss function as opposed to using the squared loss. More details on data generation and experimental setup are provided in Section D.1 in the appendix.



(a) low noise (b) medium noise (c) high noise

Figure 4.8: Fits for different noise regimes. The dots that are sampled from the noise distribution are shown in (purple) and the ones sampled from the underlying ellipse (green) are shown in (black). The true squared loss fit is shown in (red) and the one using the robust loss is shown in (blue).

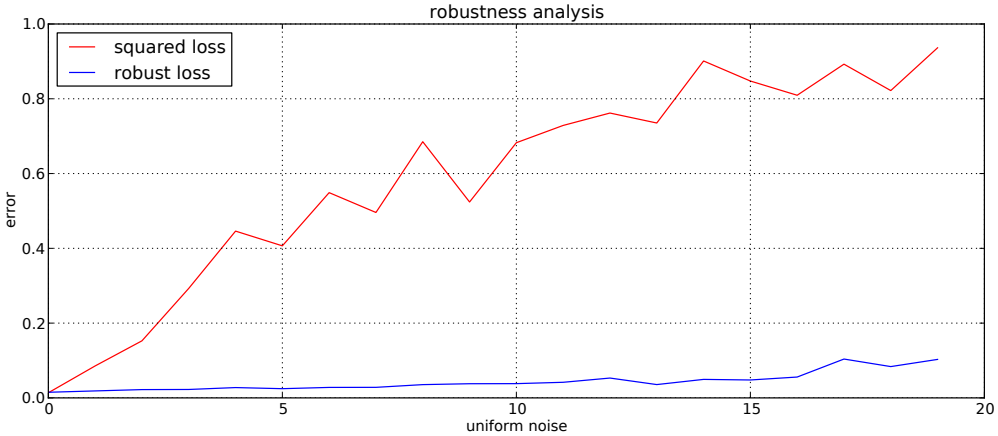
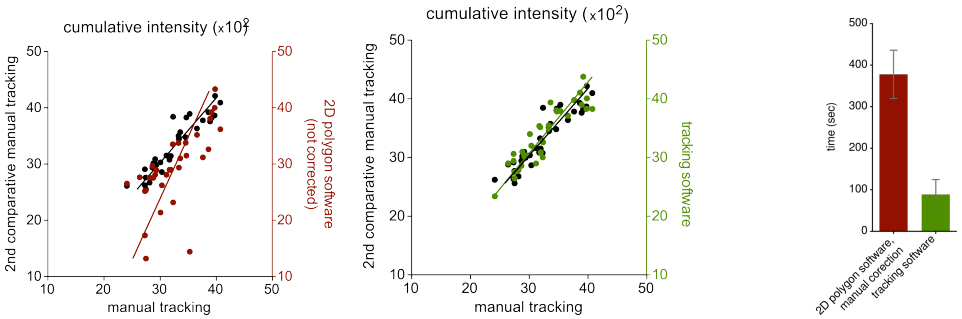


Figure 4.9: Comparison of the fitting error of the squared and robust loss for different mixtures of the distributions of the ellipsoid and noise.

Evaluation in Practice In order to evaluate the quality of our fits on real data, we compared them to manually curated segmentations obtained using the software of the microscope manufacturer. The results are shown in Figure 4.10(b). We observe that our approach has an almost perfect correlation to the manually curated ground truth, while the existing microscope software shows a considerable deviation (see Figure 4.10(a)). As we are ultimately interested in allowing for high-throughput experiments, we quan-

tified the time taken to perform an experiment using our approach and the microscope software and report a large speed-up, as shown in Figure 4.10(c). Note that the experiment was conducted by fitting each cell individually. Taking into account the proposed preprocessing pipeline and setting up batch processing will almost entirely automate the procedure. More background on the software engineering aspect, which was indispensable for the practical application of our approach, is given in Section 5.4.



(a) Microscope software (b) Our approach (c) Time gain

Figure 4.10: Results on real data comparing the $2D$ polygon fit by the microscope manufacturer (a) to our approach (b). Taking into account manual post processing, the total time taken for an experiment is shown in (c).

Conclusion

The combination of a well-designed preprocessing pipeline with a robust-loss function and graph-regularizer based on our general framework has proven to be a valuable tool in a series of experiments [Heinrich et al., 2013], where our program was used for high-throughput processing of microscopy data. This paved the way to a high-impact publication, which may not have been possible by manual analysis.

4.3.2 Joint $2D/3D$ Model Learning

Here, we present a second successful application of MTL to biological imaging [Lou et al., 2012b; Widmer et al., 2013c]. The goal is to jointly learn prediction models from well annotated $2D$ data and sparsely annotated $3D$ data. Current biological research is exhibiting a significant trend towards using $3D$ imaging techniques to monitor complex biological activities at the molecular and cellular level. This has catalyzed the emergence of a new field which is referred to as bioimage informatics, which aims at advancing image analysis to cope with the increasing complexity and quantity of data from $3D$ imaging. Despite the prevailing employment of $3D$ imaging and the heavy focus on advancing $3D$ image analysis, we raise the question: *should we just ignore all the available* $2D$ *data?* In this application, we show that existing $2D$ annotations can be used to facilitate the processing of the emerging $3D$ data by treating $2D$ and $3D$ as *two tasks* in a multitask learning setting. As a majority of experiments and analyses

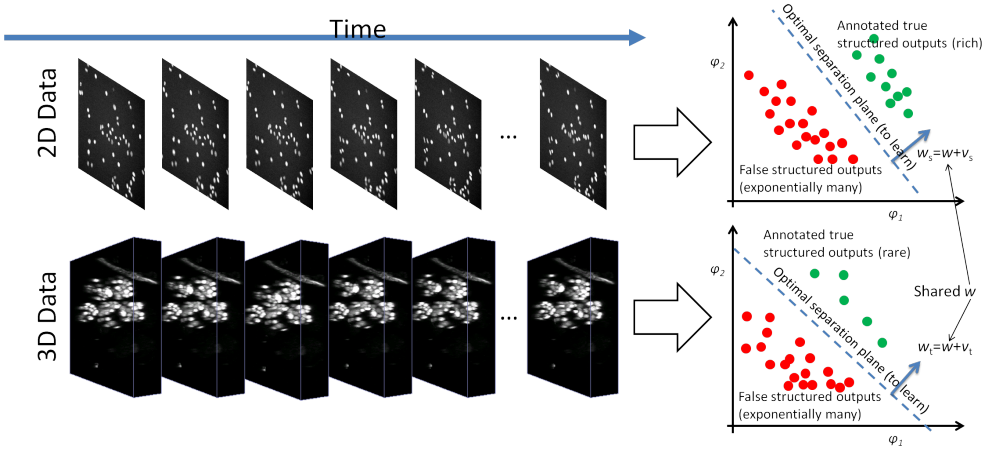


Figure 4.11: $2D$ and $3D$ image sequences exhibit different yet cognate distributions in the joint feature space. Prediction models share a component and are trained jointly [Lou et al., 2012b].

to date have been performed in the realm of $2D$ data, it is well understood we have access to rich annotations, while the recent $3D$ data is crude and manually annotating it costly in terms of time and money. Therefore, our aim is to train a high-quality prediction model for the $3D$ data using as little annotation as possible. A concrete example is shown in Fig. 4.11 in the context of cell tracking from time-lapse experiments. These two experiments capture similar biological processes (cell movement, division, etc.); therefore, we can expect that similar or identical features [Lou et al., 2012a; Lou and Hamprecht, 2011] can be employed for both tasks. However, due to differences in the underlying biological entities and the experimental conditions, they also exhibit a certain degree of variation (Fig. 4.11, right). To exploit the similarity while respecting the differences, we learn the prediction models for $2D$ and $3D$ jointly, and penalize their difference to the average parameter vector, a method which we have introduced in Section 4.1.5. We can express this as a model with a shared common component $\mathbf{w}$ but also have distinct, domain-dependent parts (viz. $\mathbf{w}_s - \mathbf{w}$ and $\mathbf{w}_t - \mathbf{w}$, the deviation of domain-specific parameters to their shared base). Since images are intrinsically structured data, the models employed in this application are structured prediction models [Tsochantaridis et al., 2005], which we have introduced in Section 2.3.4. The complete model contains several contributions for dealing with partial annotations that go beyond the scope of this thesis, therefore the reader is referred the complete description in [Lou and Hamprecht, 2012].

Joint Modeling and Learning We consider our learning problem a maximum margin risk minimization problem:

$$\min_{\mathbf{w}_s, \mathbf{w}_t} \quad \frac{\lambda_1}{2} \sum_{t=1}^{2} \| \bar{\mathbf{w}} - \mathbf{w}_t \|^2 + \frac{\lambda_2}{2} \sum_{t=1}^{2} \| \mathbf{w} \|^2 + \sum_{t=1}^{2} \mathcal{L}_t(\mathbf{w}_t), \tag{4.39}$$

where $\hat{\mathbf{w}} = \frac{1}{2}\sum_{t=1}^{2}\mathbf{w}_t$ is the average parameter vector and $\mathcal{L}_{\mathrm{s}}(\mathbf{w}_{\mathrm{s}})$ and $\mathcal{L}_{\mathrm{t}}(\mathbf{w}_{\mathrm{t}})$ are the empirical loss from the source and target domain, respectively. Clearly, this corresponds to the *Clustered MTL* instantiation of our framework given in Equation 4.1.5. The two hyper parameters λ_1 and λ_2 have two uses: firstly, they control the regularization to avoid over-fitting; secondly, the ratio $\frac{\lambda_1}{\lambda_2}$ controls the similarity between the two domains (higher means more related). For more details, please see Section D.2 in the appendix.

Results The $2D$ data in our experiment were acquired using a Nikon's TE2000 inverted microscope while the $3D$ data were acquired using a confocal microscope by Zeiss. Results are shown in Table 4.2. Our baseline is a model trained using fully annotated $3D$ data, which is, expectantly very expensive in terms of annotation efforts. This baseline approach yields a test loss of 3.69% (first row), which is significant better than simply training on the $2D$ data (second row). To address this issue, we tried 25% partial annotation which brings apparent improvement yet not sufficient (third row). By joining modeling and learning (fourth row), we bring the test loss down to 3.82% $\pm$ 0.09 and the relative error down to 3.59% $\pm$ 2.57%. We conclude that using a variant of our general framework is well suited to considerably reduce annotation cost for $3D$ data by leveraging $2D$ annotations, while maintaining high model quality.

Table 4.2: Comparison among several training settings. The baseline is a model trained using all fully annotated $3D$ data – the most expensive one in terms of annotation cost. Training on $2D$ data only, though being the cheapest approach, yields a significant increase of errors (39.8% w.r.t. the baseline). The same applies to training on partially annotated $3D$ data (21.9% increased error w.r.t. the baseline (much cheaper, though). By combining the data sets from these two approaches into a *MTL* setting, we reduced the relative error to only 3.6% at the same annotation cost.

Methods	Test loss	Relative to baseline
Trained on fully annotated $3D$ data (as baseline)	3.69%	0.0%
Trained on fully annotated $2D$ data	5.16%	+39.8%
Trained on partially (25%) annotated $3D$ data	4.41% $\pm$ 0.33	+21.88%$\pm$9.0%
Joint trained on $2D$ (full) and $3D$ (partial, 25%)	3.82% $\pm$ 0.09	+3.59%$\pm$2.57%

4.3.3 Execution Time Experiments

One of our main contributions on the algorithms side is the derivation of a Liblinear-inspired dual-coordinate descent strategy for graph-regularized multitask learning. Here, we evaluate the runtime of our proposed algorithm (described in Algorithm Table 5), which we have implemented[1] (along with a LibLinear-style shrinking strategy) in C++ as a part of the SHOGUN machine learning toolbox Sonnenburg et al. [2010]. We compare our solver with the state-of-the-art, that is, SVMLight (as integrated into the SHOGUN toolbox) using the multitask kernel (MTK) as defined in Section 4.1.5. We experiment on the following five data sets summarized in Table 4.3:

[1]For implementation details, see: `http://bioweb.me/mtl-dcd-solver`

	dim	#examples	#tasks
Gauss2D	2	$1 \cdot 10^5$	2
Breast Cancer	44	474	3
MNIST-MTL	784	$9.0 \cdot 10^3$	3
Land Mine	9	$1.5 \cdot 10^4$	29
Splicing	$6 \cdot 10^6$	$6.4 \cdot 10^6$	4

Table 4.3: Overview of the data sets used in this experiment.

- *Gauss2D* A controlled, synthetic data set consisting of a balanced sample from two isotropic Gaussian distributions.
- *Breast Cancer* A classic benchmark data set consisting of a genetic signature of 60 genes used to predict the response to chemotherapy.
- *MNIST-MTL* A multi-task data set derived from the well-known MNIST data[2] by considering the three separate tasks "1 vs. 0", "7 vs. 9", and "2 vs. 8".
- *Landmine* A classic multi-task data set, where the different tasks correspond to detecting land mines under various conditions [Xue et al., 2007].
- *Splicing* This is the most challenging data set: a large-scale, multiple-genomes, biological data set, where the goal is to detect splice-sites in various organisms, each organism corresponding to a task. The features are derived from raw DNA strings by means of a weighted-degree string kernel [Sonnenburg et al., 2007b].

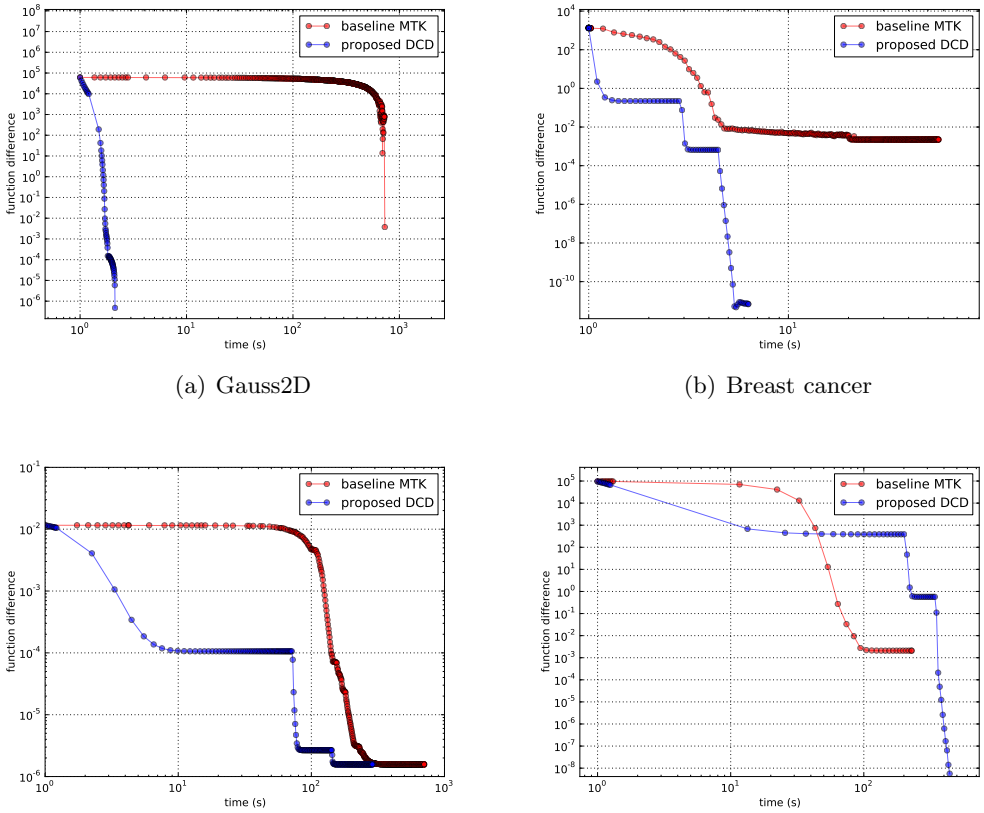
The above data sets are taken from various application domains including computer vision, biomedicine and computational genomics. They cover many different settings such as small and large dimensionality, various numbers of examples and tasks. Our corpus includes controlled synthetic data, established real-world benchmark data and challenging multiple-genomes splice data. The first four data sets contain real valued data, for which we used linear kernels and corresponding standard scalar products.
To compare our implementation with SVMLight using the multi-task kernel (MTK), we measure the *function difference*

$$\Delta := |\mathrm{obj}^* - \widehat{\mathrm{obj}}|\,,$$

where obj^* the true optimal objective and $\widehat{\mathrm{obj}}$ the actual objective achieved by the solver (for DCD and MTK these are primal and dual objectives, respectively). The true objective obj^* is computed up to a duality gap of $< 10^{-10}$. All experiments are performed on a 4GB AMD64 machine using a single core.

The results are shown in Figure 4.12, where the function difference of the four real-valued data sets is shown as a function of the execution time. First of all, we observe

[2] `http://yann.lecun.com/exdb/mnist/`



(a) Gauss2D

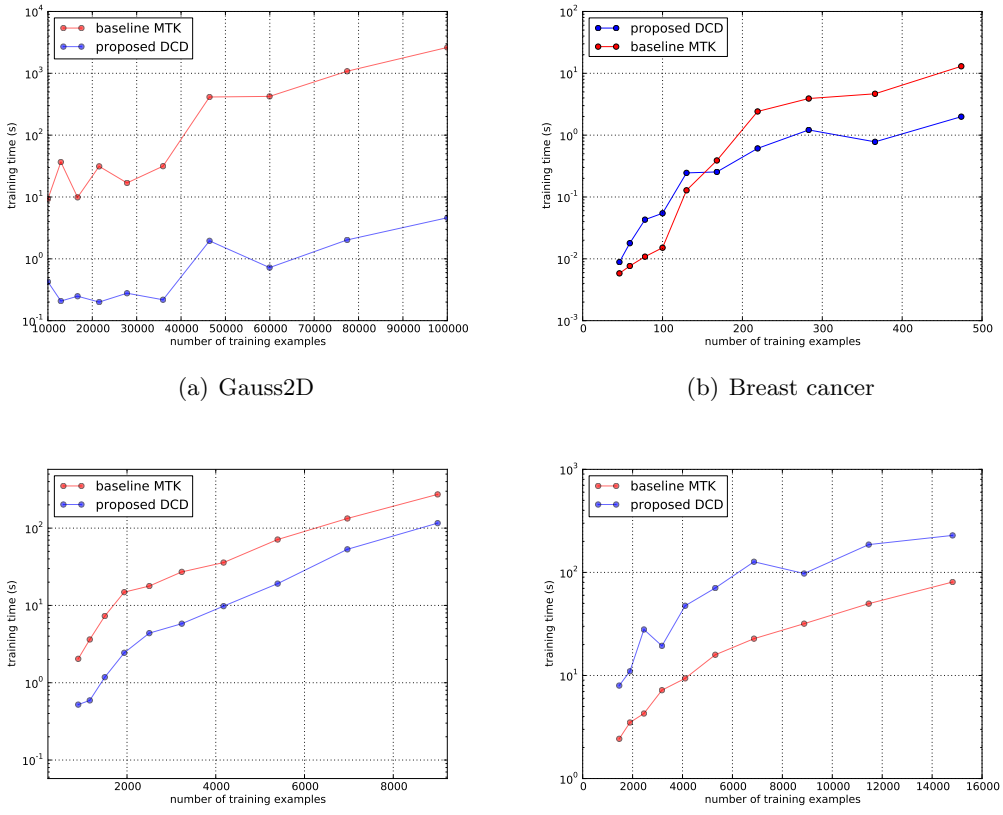
(b) Breast cancer

(c) MNIST-MTL

(d) Land Mine

Figure 4.12: Results of the runtime experiment in terms of the function difference as a function of the execution time [Widmer et al., 2012].

that in all four cases the two solvers suffer from an initialization phase in which the function value improves only slowly. For *Gauss2D* the convergence properties of the two methods (e.g., steepness of the decrease in function difference) are very similar, but our proposed DCD solver being up to three magnitudes faster. Furthermore, we observe that, for two of the four data sets, the MTK baseline fails to decrease the function difference beyond a threshold ranging from 10^{-2} to 10^{-4}, while the proposed DCD algorithm nicely converges to a precision of 10^{-7} to 10^{-10} (cf. Figure 4.12 (b) – (d)). Our method is outperformed by the MTK algorithm on the *landmine* data set (see Subfigure 4.12(d)), which indicates that our strategy is at a disadvantage if the number of tasks is large relative to the number of training examples, due to the update rule given by Equation 4.26 and should only be used if the number of tasks is moderate. For the remaining three data sets, we can observe that if we stop both algorithms at some arbitrary time point, our method tends to output a solution that is more precise than the MTK baseline by usually several orders of magnitudes (up to ten orders for,



(a) Gauss2D

(b) Breast cancer

(c) MNIST-MTL

(d) Land Mine

Figure 4.13: Results of the second runtime experiment: required time to train a multi-task SVM to a relative precision of 10^{-4} for various sample sizes n [Widmer et al., 2012].

e.g., *Gauss2D, and Breast Cancer*).

In a second experiment, we measure the training time a solver needs to reach a given precision (we chose 10^{-4}) as a function of the training set size. The results of this experiment are shown in Figure 4.13. We observe that for 3 out of 4 data sets, the proposed DCD methods requires less computation time than the MTK solver. For the synthetic data set the difference is the most drastic, being of the order of up to 2.5 magnitudes. Confirming the observation from the last set of experiment, our method is again outperformed on the *landmine* data set, where the number of tasks is high. The reason for this is the loop over tasks in update rule in Equation 4.26. A possible way to overcome this limitation is to sparsify the dual task similarity measure, such that each task is only connected to a constant number of other tasks, greatly limiting the runtime of the inner loop over related tasks in our solver.

Finally, we study a large splice-site data set (for a description of the data, please review Section 3.4.2), where the goal is to detect splice-sites in various organisms, each

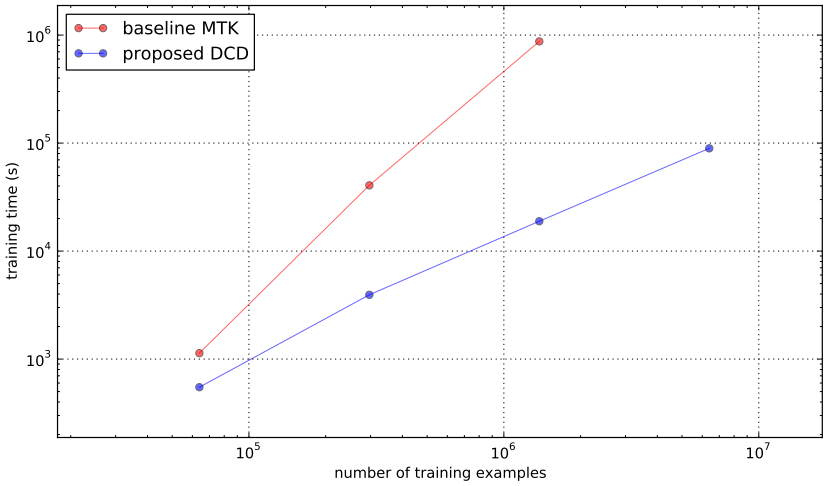


Figure 4.14: Results of the large-scale splice-site detection experiment [Widmer et al., 2012].

organism corresponding to one task. For the MTK solver, the features are derived from raw DNA strings by means of a weighted-degree string kernel [Sonnenburg et al., 2007b] of degree 8 (as described in Section 2.4.1). For the DCD solver, we combine the proposed algorithmic methodology with the COFFIN framework [Sonnenburg and Franc, 2010] as described in Section 2.4.2. The results of this experiment are shown in Figure 4.14. We observe that the proposed DCD solver is capable of dealing with millions of training points, while the MTK baseline is limited to rather moderate training set sizes of up to hundreds of thousands training points. This experiment demonstrates that we are now able to train on very large genomic sequences in reasonable time, finally allowing for truly large-scale multi-task learning. Additional information, including source-code and data sets are provided in Section D.3 in the appendix.

4.3.4 MHC-I Binding Prediction

The first application of graph-regularized multitask Learning involved MHC-I binding prediction, an important problem from immunology and systems biology. MHC class I molecules are key players in the human immune system. They bind small peptides derived from intracellular proteins and present them on the cell surface for surveillance by the immune system. Prediction of such MHC class I binding peptides is a vital step in the design of peptide-based vaccines and therefore one of the major problems in computational immunology. Thousands of different types of MHC class I molecules exist, each displaying a distinct binding specificity. We consider these *different MHC types to be different tasks* [Heckerman et al., 2007; Jacob and Vert, 2008a] in a MTL setting.

Background

MHC-I molecules are membrane-bound proteins with a closed binding groove that holds peptides in an extended conformation (Figure 4.15B). They typically bind peptides that contain eight to ten amino acids (AAs) with a preference for nine AAs. The corresponding gene complex is highly polymorphic. To date, more than 3,000 different

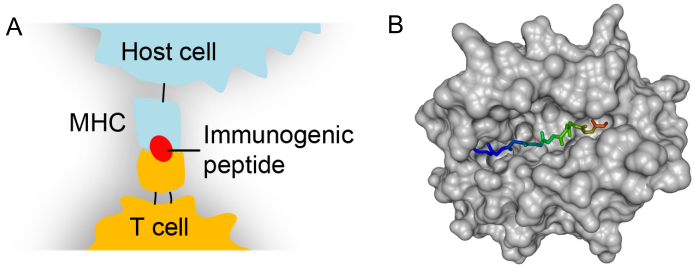


Figure 4.15: Peptide/MHC-I complex. A) An MHC-I molecule presents an immunogenic peptide on the cell surface where it is recognized by a T cell. B) The structure of a nonameric peptide complexed with an MHC-I molecule. The binding groove is closed at both ends and the peptide is bound in an extended conformation. (PDB-ID: 3L3D (http://www.pdb.org)Berman et al. [2000], plotted with BALLView Moll et al. [2006])

MHC-I alleles are known, each coding for an MHC-I molecule binding a specific range of peptides. Any human has up to six different types of MHC-I molecules. This implies that a peptide that is capable of inducing an immune response in one individual might never be presented on the cell surface in another. In order to design vaccines effective for a given population, it is therefore necessary to accurately predict MHC-I binding for a wide range of different MHC alleles [Toussaint and Kohlbacher, 2009].

Many computational methods for the classification of peptides into MHC-I binders or non-binders have been proposed; they range from matrices [Rammensee et al., 1999; Reche et al., 2002] to machine learning [Adams and Koziol, 1995; Dönnes and Elofsson, 2002]. All of these methods require a certain amount of experimental binding data for each allele under consideration. However, a major problem in MHC-I binding prediction is the lack of data. There is little to no experimental binding data available for the vast majority of the known alleles, which further complicates the development of accurate prediction methods. In 2008, Jacob and Vert [2008a] proposed a kernel-based approach that tries to overcome the lack of training data by sharing binding information across alleles. Building upon their work, we combine two approaches to improve MHC-I binding prediction. First, we develop an improved string kernel that is particularly well suited for dealing with amino acid sequences by taking into account physio-chemical properties of amino acids (see Figure 4.16), which allows more accurate predictions for alleles with little binding data. Details on the design of this specialized string kernel are not further elaborated as it goes beyond the scope of this thesis, therefore we refer the reader to the full publications [Toussaint et al., 2010; Widmer et al., 2010c] for more detail. Second, we consider an enhanced multitask learning algorithm,

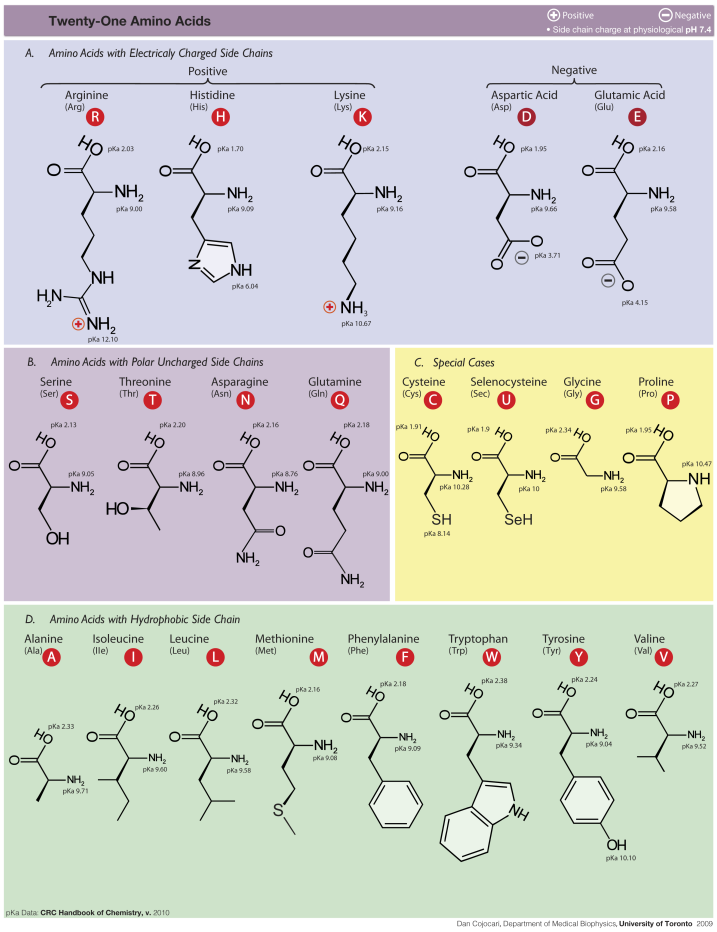


Figure 4.16: We show a table of amino acids, grouped by their physio-chemical properties (see A in the Appendix for image licenses). The main idea of the enhanced string kernel is that two amino acids that are very similar in terms of their physio-chemical properties are more likely to engage in similar binding behavior than amino acids that have little in common. This idea is incorporated into the string kernel by using a similarity measure based on physio-chemical properties, instead of the hamming distance.

which can be used to improve prediction performance for an allele by utilizing binding data of similar alleles. We are able to show that the combination of both approaches outperforms existing methods.

Method

Clearly, we expect information sharing between tasks to be fruitful, if the so-called *binding pocket* of the molecules exhibit similar properties. These properties are encoded in the protein sequence of the MHC proteins. Of particular relevance are those amino acids that come in contact with the peptide in the binding pockets. In this context, a string representation of the amino acids within the binding pocket is referred to as

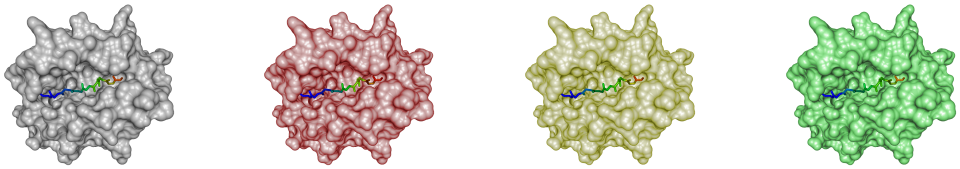


Figure 4.17: We jointly consider multiple different MHC-I in a MTL setting.

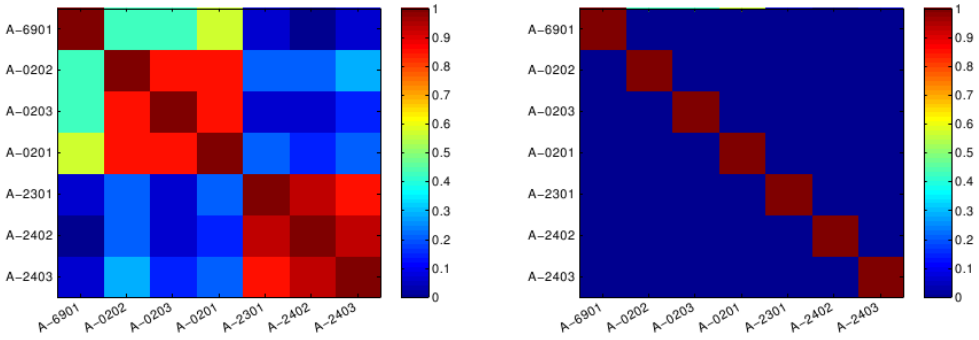
pseudo-sequence (the *pseudo* prefix emphasizes the fact that the amino acids in the pseudo-sequence are typically not adjacent in the chain of the MHC-I peptide). The approach used by Jacob and Vert [2008a] is based on the multitask kernel formulation, which we discussed in Section 4.1.5:

$$K^{\text{MT}}((x,s),(z,t)) = K^{\text{allele}}(s,t) \cdot K^{\text{peptide}}(x,z),$$

where $K^{\text{allele}}(s,t)$ is a string kernel defined on the pseudo-sequence of the allele's binding pocket and $K^{\text{peptide}}(x,z)$ is a string kernel defined on the amino acid sequence of the binding (or non-binding) peptide. Extending the above framework, we define our own multitask kernel formulation in terms of the novel kernel and allow the trade-off between in-domain and out-of-domain information to be parameterized by weights β_1 and β_2, which we tune as part of the learning procedure.

$$K^{\text{MT-WD-RBF}}((x,s),(z,t)) = \beta_1 \cdot K^{\text{WD}}(s,t) \cdot K^{\text{WD-RBF}}(x,z) + \beta_2 \cdot \delta_{s,t} \cdot K^{\text{WD-RBF}}(x,z),$$

where the first summand corresponds to the out-of-domain kernel and the second to the in-domain kernel. An illustration for the above composite kernel, consisting of an



(a) out-of-domain task kernel matrix (b) in-domain task kernel matrix

Figure 4.18: Shown on the left is the kernel matrix of the task kernel based on the pseudo-sequence of binding pockets. The in-domain kernel matrix on the right corresponds to the idendity matrix, which disallows any *cross-talk* between the tasks [Widmer et al., 2010c].

in-domain and out-out-domain component is given in Figure 4.18. For more details on

the experimental setup, please see Section D.4 in the appendix or consult the original publication [Widmer et al., 2010c].

Results

In this experiment, we use the IEDB benchmark data set, which consists of 35 human alleles [Peters et al., 2006]. We compare the following methods: *Plain (WDK)* uses a single-task classifier in conjunction with a regular WD-kernel on peptides, *MTL (WDK)* uses a multitask kernel with a regular WD-kernel on peptides, *MTL (WDK-RBF)* uses a multitask kernel with the novel WDK-RBF kernel defined on peptides and finally *MTL-B (WDK-RBF)* is the same multitask kernel as the last one, except mixing weights between in-domain and out-of-domain components are tuned. Figure 4.19

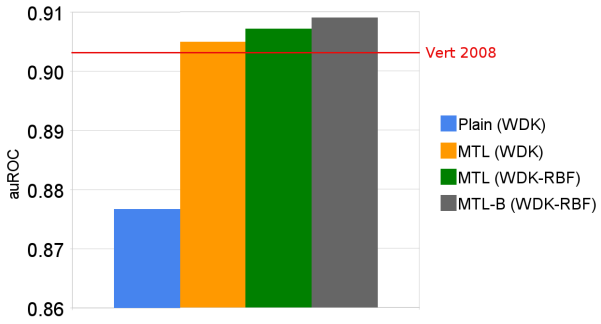


Figure 4.19: Prediction accuracy in auROC for a 5-fold cross-validation averaged over splits and tasks on the IEDB benchmark data set as reported in [Widmer et al., 2010c], comparing the methods described in the main text.

shows the results from a 5-fold cross-validation averaged over splits and tasks. In agreement with previous reports [Jacob and Vert, 2008a], we observe that using multitask learning considerably boosts the prediction accuracy over its single-task equivalent. Furthermore, we see a moderate improvement from using our special-purpose string kernel to capture peptide similarities. Finally, we observe that fine-tuning the weights between in-domain and out-of domain components further improves accuracy. This very observation led us to further investigate the *learning* of composite multitask kernels which we have now formalized in our general framework in Section 4.1 and will be further illuminated in the following section.

4.3.5 Learning the Similarity

In this final set of experiments, we investigate strategies to infer or refine task similarity measures on synthetic and real data sets.

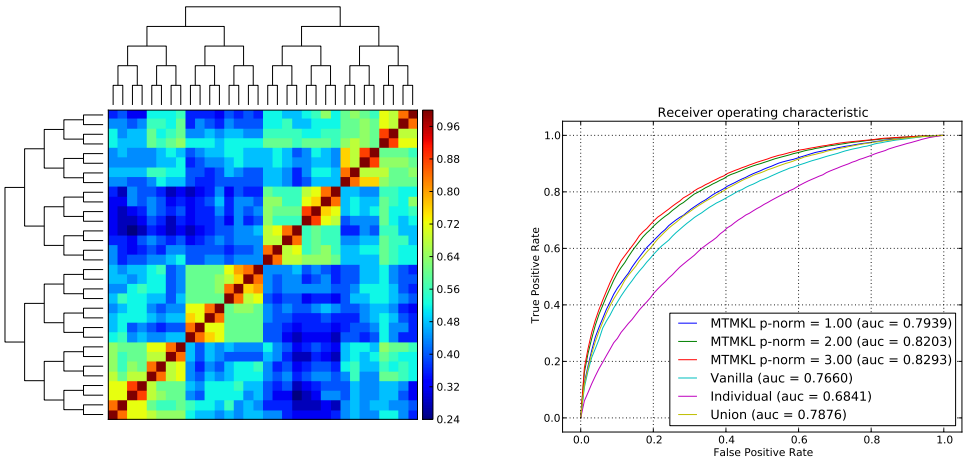
Toy experiment for Hierarchical MT-MKL

In this experiment, we evaluate Hierarchical MT-MKL as described in Section 4.1.6 on an artificial data set motivated by biological evolution. At the core of this example is the binary classification of examples generated from two 100-dimensional isotropic Gaussian distributions with a standard deviation of 20. The difference of the mean vectors μ_{pos} and μ_{neg} is captured by a difference vector μ_d. We set $\mu_{pos} = 0.5\mu_d$ and $\mu_{neg} = -0.5\mu_d$. To turn this into a MTL setting, we start with a single $\mu_d = (1, \ldots, 1)^T$ and apply mutations to it. These mutations correspond to flipping the sign of m dimensions in μ_d, where $m = 5$. Inspired by biological evolution, mutations are applied in a hierarchical fashion according to a binary tree of depth 4 (corresponding to $2^4 = 32$ leaves). Starting at the root node, we apply subsequent mutations to the μ_d at the inner nodes and work down the tree until each leaf carries its own μ_d. We sample 10 training points and 1000 test points for each class and for each of the 32 tasks. The similarity between the μ_d at the leaves is computed by taking the dot product between all pairs and is shown in Figure 4.20(a). Clearly, this information is valuable when deciding which tasks (corresponding to leaves in this context) should be coupled and will be referred to as the *true* task similarity matrix in the following. We use Hierarchical MT-MKL by creating adjacency matrices for each inner node and subsequently learning a weighting using MT-MKL.

We compare MT-MKL with $p = 1, 2, 3$ to the following baseline methods (see also Section D.5.1 in the appendix): *Union* that combines data from all tasks into a single group, *Individual* that treats each task separately and *Vanilla MTL* that uses MTL with the same weight for all matrices. We report the mean (averaged over tasks) ROC curve for each of the above methods in Figure 4.20(b). From Figure 4.20(b) we observe that the baseline *Individual* performs worst by a large margin, suggesting that combining information from several tasks is clearly beneficial for this data set. Next, we observe that a simple way of combining tasks (i.e. *Union*) already considerably improves performance. Furthermore, we observe that learning weights of hierarchically inferred task grouping in fact improves performance compared to *Vanilla* for non-sparse MT-MKL (i.e. $p = 2, 3$). Of all methods, non-sparse MT-MKL is most accurate for all recall values.

MHC-I Binding Prediction using Powerset MT-MKL

In this experiment, we investigate whether we can use our general framework to learn a task similarity *ab-initio*, that is without the use of prior information using Power-set MT-MKL as described in Section 4.1.6. Due to the exponential nature of the Power-set MT-MKL algorithm, we restrict ourselves to a subset of the MHC-I data used in Section 4.3.4. The subset consists of peptide sequences of length $l = 9$ for 7 tasks. In total, we are given 7367 examples (A_2403=254, A_6901=833, A_0201=3089, A_0202=1447, A_0203=1443, A_2402=197, A_2301=104). For cross-validation, the data was split randomly into 5 splits of the same size. Additional information is provided in Section D.5.2 in the appendix. Unlike the setting of the previous experiment, we are not given a hi-



(a) Task similarity matrix

(b) Performance HMTKL vs Baselines

Figure 4.20: In 4.20(a), we show the similarity matrix between all 32 tasks as generated by the procedure outlined in the main text. Comparison of MT-MKL to baselines *Vanilla MTL*, *Union*, *Individual* is shown in 4.20(b), where ROC curves are averaged over tasks for each method.

erarchical structure relating tasks at hand. To demonstrate that meaningful groups of tasks can be identified by Powerset MT-MKL, we do not assume any prior knowledge of task relationships. Please note, however, that we do have access to the pseudo-sequences of the MHC-I binding pockets (see Section 4.3.4). We use these sequences to evaluate the task similarities obtained by our approach.

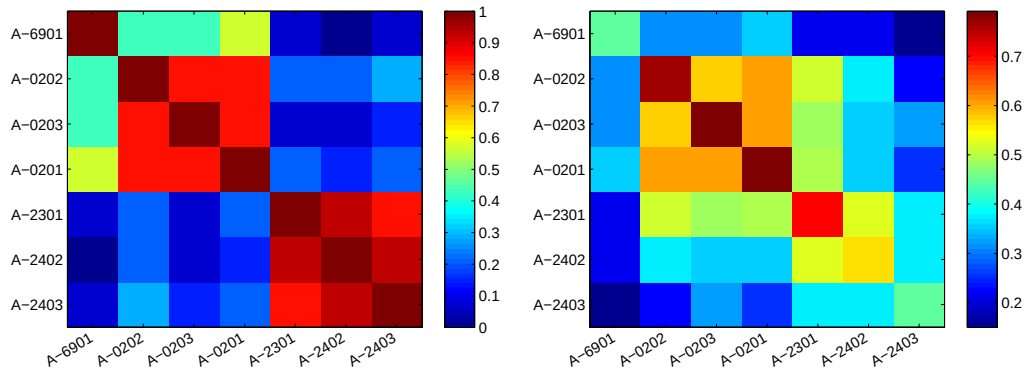
We report the area under the precision recall curve (auPRC) in Table 4.4.

auPRC	Plain	Union	Vanilla MTL	Powerset MT-MKL
cross-validation	0.668	0.637	0.676	0.692
test set	0.671	0.576	0.679	0.699

Table 4.4: Results for the MHC experiment in auPRC for the model selection step and the final prediction on the test set. Reported is the average performance over all tasks.

We observe that simply combining the data for different tasks to obtain a single model (*Union*) does not outperform the naïve method of obtaining an individual classifier for each task (*Plain*). This hints at rather large differences between the tasks. Learning the weights with MKL further improves performance compared to the *Vanilla MTL* approach, which already outperforms the other two baselines.

From Table 4.5, we observe that the tasks *A_0201*, *A_0202*, *A_0203*, are often grouped in the same task set, which is in agreement with domain knowledge (i.e. the grouping based on the similarity of pseudo-sequences). We compute the similarity of between



(a) External task similarity

(b) Learned task similarity

Figure 4.21: Comparison between learned similarities and similarities based on the comparison of allele sequences. The learned similarity of A-2301 with A-0203, A-0201 and A-0202 in (b) can be attributed to structural features that cannot be easily inferred from the allele sequence [Widmer et al., 2010a].

Task Set	weight
A_0201, A_0202, A_0203, A_6901	0.186
A_0201, A_0202, A_0203, A_2301	0.178
A_0202, A_0203, A_2301, A_2402, A_2403	0.110
A_0201, A_0203, A_2301, A_2402, A_2403	0.091
A_0201, A_0202, A_2301, A_2402, A_6901	0.074
A_0201, A_0202, A_2301, A_2402, A_2403	0.066

Table 4.5: List of task sets and their respective weights β_i that were assigned by 1-norm MKL.

pairs of tasks by summing the weights of groups that contain both items of the pair. For evaluation of the learned similarities, we compare them to the hamming distance (or similarity) between the amino acids in the binding pocket of the MHC-I molecules. In Figure 4.21, we compare the inferred task similarity to the similarity based on the sequence of MHC molecules (binding pocket sequence similarity). We find a good agreement between the inferred task similarity and the molecule-based similarity. Using MKL, we successfully identify groups of tasks among which information sharing is sensible, thus allowing for a smarter combination of information from different tasks in the absence of prior knowledge. The improvement in performance over the *Vanilla MTL* method is relatively small. However, we are able to demonstrate that we can identify as sensible task structure *ab-initio* using our general framework.

4.4 Discussion

We have presented a general framework that subsumes a multitude of approaches for transfer learning between related tasks in a regularized risk minimization framework. Using modern techniques from mathematical optimization, we established the primal dual relationship and investigated the relation to various special cases, including the SVM, multiple kernel learning as well as various multitask learning approaches. This helps to understand the relationship between different MTL approaches and allows to develop novel instantiations. By combining ideas from multitask learning and multiple kernel learning, we described a number of novel formulations that allow the joint learning of multiple task, while simultaneously weighting the entities that describe task similarities, such as multiple clusterings, graphs or using an approach to learn a piecewise linear transformation of task-similarities. This branch of multitask learning is based on the assumption that model parameters are similar. If this assumption is violated and similarity between unrelated tasks is enforced, performance will suffer from *negative transfer*. The ability to learn a weighting of task similarity measures using multiple kernel learning may be viewed as a strategy to avoid negative transfer in the face of uncertainty with respect to task relatedness, therefore alleviating a central problem in multitask learning. Furthermore, we contribute a fast dual coordinate descent numerical optimization procedure for the special case of the hinge-loss, while keeping the choice of regularizer general. We believe that this is a valuable contribution for several reasons. First, it brings solvers for multitask learning on par with state-of-the-art linear SVM solvers, allowing us to combine multitask learning with recent advances in linear solvers such as the COFFIN framework [Sonnenburg and Franc, 2010]. Second, in the age of Big Data, new data points are generated at ever increasing pace. This is particularly true in the field of computational biology, where advances in measurement technology (e.g. RNAseq) even outpace Moore's law. Therefore, the ability to update models, be it through domain adaptation as discussed in Chapter 3 or dual coordinate descent solvers that lead to way to stochastic gradient descent methods is highly desirable. Lastly, we are able to derive a linear solver for multiple kernel learning as a special case of our framework, which is another novel contribution. We have presented a number of applications from computational biology, immunology, and bioimaging, where we can show that instances of the MTL framework developed as part of this thesis are beneficial in real world applications. The experiment on MHC-I binding prediction is a perfect example for a broader class of biological prediction problems. Here, data is available for more than one biological entity, namely different MHC-I alleles. We show that the combination of data from different tasks, as well as tuning the trade-off between in-domain and out-of-domain information improves the quality of our model. This is but one example of a problem from computational biology, where data is available for more than one biological entity. As more and more data is available from *cell biology*, we will be able to develop computational models for an ever increasing number of biological sub-systems. In this context, we believe there is a growing demand for methods such as the ones presented in this thesis to bring together

information from related biological entities. Moreover, we presented two applications of graph-regularized mutitask learning for bioimaging. We are able to show that ideas from MTL can be used to detect 3-dimensional biological objects in images from fluorescence microscopy. Here, we not only developed a practical tool for biologists that led to a high-impact publication, but also provided a helpful visualization of the effect that multitask regularization has on related models. The other application to bioimaging is a prime example for the use of multitask learning to bring down labeling cost. When going from $2D$ to $3D$, we demonstrate that we can use instances of our general framework to combine the large number of $2D$ annotations with the small number of $3D$ annotation, therefore greatly reducing the cost for label generation. This principle is highly valuable when learning on a budget, as oftentimes a large fraction of the experimental expenses may be attributed to labeled data generation. Furthermore, this application constitutes a novel contribution in the sense that it combines the use of MTL regularizers with a loss function from structured output learning. In a number of runtime experiments, we evaluate our newly presented solvers in a range of settings. As expected, our method excels when the number of training examples is large and the number of tasks is moderate. For large number of tasks, we observe that it is better to use more traditional solvers in conjunction with multitask kernels. Lastly, we explore different strategies to learn or refine task similarities. In two experiments, we demonstrate the use of our general framework to learn or refine task similarity. In the first experiment, we learn a weighting of groups of tasks that come from a hierarchical structure, where we manage to outperform unweighted counterparts and baseline methods. Lastly, we demonstrate a principled way to learn task similarity from scratch using a *power-set* approach that, although not efficient in its current form, successfully reconstructs task similarity close to the ones obtained from external biological knowledge, which remained hidden to the learning algorithm. These results encourage us to further pursue this line of research and perform additional experiments in an even broader range of settings as future research.

5 Software

An integral part of the work of a computational biologist is the development, release and maintenance of software. In the following, we will review our considerations with respect to efficiency of computation, usability and our strategy for making programs available. A overview of the software that was developed as part of this thesis, including the publications for which the respective program was developed or used is given in Table 5.1.

Program Name	Type	License	Publications
SHOGUN	ML toolbox	GPLv3	[Sonnenburg et al., 2010] [Schweikert* et al., 2009] [Widmer et al., 2010b] [Widmer et al., 2010a] [Toussaint et al., 2010] [Görnitz* et al., 2011] [Widmer et al., 2012] [Lisitsyn et al., 2013] [Widmer et al., 2014, in prep.]
pythongrid	cluster computing	GPLv2	[Widmer and Ong, 2013] [Schweikert* et al., 2009] [Widmer et al., 2010b] [Widmer et al., 2010a] [Widmer et al., 2010c] [Görnitz* et al., 2011] [Widmer et al., 2014, in prep.]
GRED	image analysis tool	GPLv2	[Widmer et al., 2013a] [Heinrich et al., 2013]
ExpViz	micro array vis.	GPLv2	[Laubinger et al., 2008]
TileViz	tiling array vis.	GPLv2	[Zeller et al., 2009]
WormViz	micro/tiling array vis.	GPLv2	[Spencer et al., 2011]
splashRNA	shRNA binding pred.	GPLv2	[Pelossof* et al., 2014, in prep.]
fastLMM select	GWAS tool	MSRL	[Lippert et al., 2014a] [Lippert et al., 2014b]
Tapkee	dim-reduction library	BSD	[Lisitsyn et al., 2013]
pyMTL toolbox	MTL library	GPLv2	[Widmer et al., 2014, in prep.]

Table 5.1: Table of software contributions as part of this thesis.

5.1 Computational Considerations

In scientific computing, two antagonistic objectives are of central importance. One motivation is to *rapidly prototype* ideas, thus requiring a programming environment needs to be flexible, modular and sometimes interactive. On the other hand, programs need to be *efficient*, highly optimized, and give tight control over the underlying hardware in order to be able to deal with scientific problems in the age of Big Data. We meet both requirements by pursuing a hybrid programming model. High-level code such as data parsing, cleaning (i.e. *data munging*), work distribution, performance evaluation and visualization is done in the scripting language *python*, which is rapidly becoming the work horse of scientific computing. The *heavy-lifting* such as linear algebra operations and most importantly solvers for optimization problems are written in the *low-level* language *C++*, which runs directly on the hardware and therefore allows for highly efficient implementations. Clearly, this hybrid model comes at the cost of additional complexity of having to combine two languages. However, the generation of *language bindings* for *C++* libraries is greatly facilitated by tools such as *SWIG* (Simple Wrapper Interface Generator), which allows to automatically map the class hierarchy from the source language (e.g. *C++*) directly onto a target language (e.g. *python*). We make use of the SWIG-based language bindings that are part of SHOGUN [Sonnenburg et al., 2010], for generating *python*-bindings for all of our solvers for domain adaptation and multitask learning, allowing us to pursue the described hybrid model. Using this hybrid model not only allows rapid prototyping, but also facilitates the distribution of computation in a cluster computing environment, as we will discuss in the following.

5.2 Grid Computing

In machine learning experiments, a lot of time is typically spent on performing model selection and performance evaluation via regular or nested cross-validation (see Section 2.3.5), resulting in hundreds or thousands of computations. These computations are essentially independent of each other and therefore lend themselves to parallelization in a cluster environment. This form of parallelism is often referred to as *embarrassingly parallel* [Foster, 1995], as it is straight-forward to perform parallel execution. A paradigm that has become synonymous to *big data analysis* in the last years is *map-reduce* [Dean and Ghemawat, 2008]. In a nutshell, the idea is that different arguments are mapped onto the same function, producing different outputs. These outputs correspond to key-value pairs, which are *reduced* to the final result in a second step. Given the hybrid structure of our computational model and the constraint that the available grid computer is running the Sun Grid Engine (SGE), we developed *pythongrid*, a lightweight tool that enables us to perform experiments using the map-reduce paradigm in python. *pythongrid* builds upon the DRMAA library, which provides a communication layer to SGE and takes care of scheduling pythongrid jobs to the SGE queue. The typical work flow within pythongrid is shown in Figure 5.1. One of its main features is the built-in fault recovery layer based on the high-performance networking library

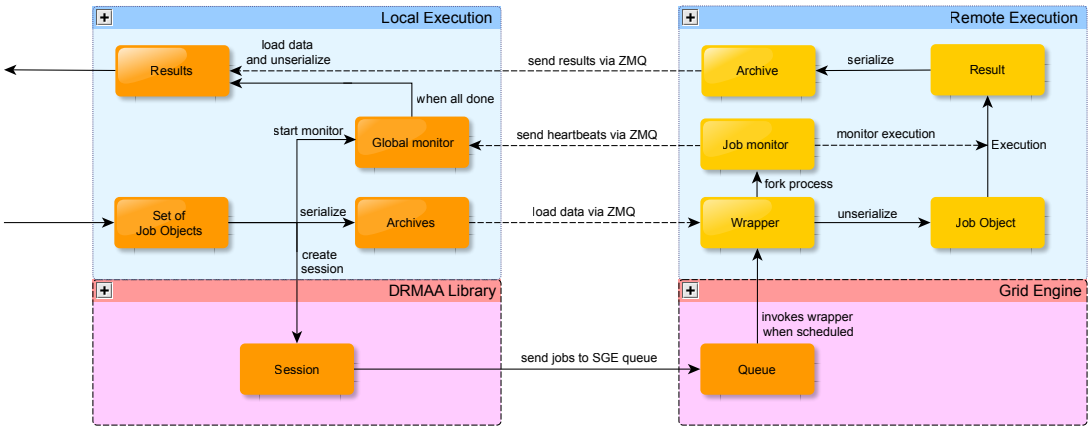


Figure 5.1: Workflow of pythongrid: A call to *map* generates a set of job objects, each carrying a pointer to the function to be executed and the corresponding arguments. After the job objects have been created, three things happen. First, the objects are serialized such that they can be transferred over the network. Second, the jobs are sent to SGE using the DRMAA library and third, a new monitoring process is created that will interpret heartbeats from worker nodes. Once a job is scheduled by SGE, it will load the serialized job object via ZMQ, unserialize and start execution. Additionally, a local monitoring process is forked off (recording memory footprint and CPU load) that sends back *heart-beat* messages to the global monitor process on the submission host via ZMQ. If an unexpected failure is detected, the global monitor will resubmit the failed job. Finally, the return values of the execution will be serialized and send back to the submission host via ZMQ. Once all jobs have finished, the set of results can be *reduced* to its final form.

ZMQ [Hintjens, 2013]. We use ZMQ to send heart-beats from the worker node to the monitoring process on the submission host to monitor the worker process. In case a job fails, pythongrid attempts to detect the reason (e.g. exception, out-of-memory, node failure) and resubmits the job if appropriate. Furthermore, it does not have a dependency on network file systems such as NFS (these often cause considerable problems in practice, when too many i/o operations are performed at the same time), as data input and output is handled by ZMQ using inter-process communication over the network.

5.3 Availability

We believe that making scientific code available is an (often neglected) integral part of the scientific process [Sonnenburg et al., 2007c]. Some of the core contributions presented in this thesis are solvers for domain adaptation and multitask learning (see Figure 5.2 for details). All of them are implemented as part of the SHOGUN machine learning toolbox (see Table 5.1), which is open-source software licensed under the GPL3 and can be freely downloaded from github. Contributions to SHOGUN that go beyond the incorporation of methods that are part of this thesis, but extend to the implementation of various machine learning algorithms (e.g. the dimensionality-reduction library *Tapkee* [Lisitsyn et al., 2013] was developed as part of Google Summer of Code).

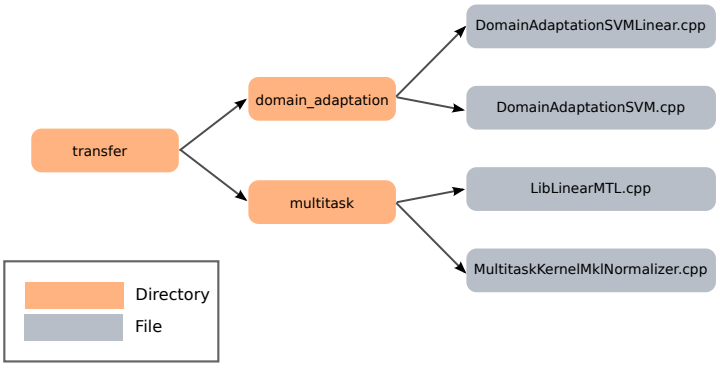


Figure 5.2: Location of key implementations within SHOGUN for domain adaptation and multitask learning. The different methods described in Chapter 4 use the same core algorithms implemented in *LibLinearMTL.cpp* for the linear case and *MultitaskKernelMklNormalizer.cpp* for the multitask kernel case. Similarly, a linear and kernel-based implementation of domain adaptation is available in *DomainAdaptationSVMLinear.cpp* and *DomainAdaptationSVM.cpp*, respectively.

pythongrid is open-source software as well and has been publicly available on Google code. It is also registered on the python package index, which greatly facilitates installation for users. Since its publication to Google code at the beginning of 2013, it has had over 1,200 downloads at the time of writing[1,2].

5.4 User Interaction

While most of the software written as part of this thesis can be used via a command-line interface or as part of a library, some of the tools are intended to be used directly

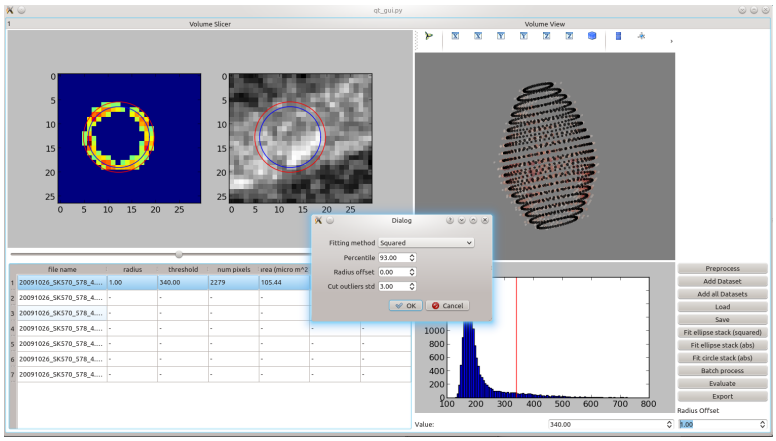


Figure 5.3: Graphical user interface of GRED

[1] `https://crate.io/?has_releases=on&q=pythongrid`
[2] `https://code.google.com/p/pythongrid/downloads/list`

by biologists to facilitate the design and analysis of experiments. In these cases, it is not sufficient to provide an API or command line interface, but rather one needs to invest a considerable amount of time in order to develop an intuitive and easy-to-use interface. Two of the projects with a particularly strong focus on user interaction are *GRED*, mathematical aspects discussed in Section 4.3.1. and *splashRNA*, an example for successfully learning from biological sequences in Section 2.4.3. GRED is based on the PyQT library for creating graphical user interfaces, in conjunction with Matplotlib and Mayavi to visualize $2D$ and $3D$ data, respectively. A screen-shot of the GRED GUI is shown in Figure 5.3. We implemented a rich-client interface for this project as data sets are typically large and direct user interaction is important to immediately judge the quality of our model. For splashRNA, which will be made available to biologists worldwide, we created an easy-to-use web-service. A screen-shot of the web-service is shown in Figure 5.4. Using a python-based web-server, we created a front-end that allows users to perform predictions using trained learning machines based on SHOGUN. That way, the users can benefit from our software and the models we have trained without having to go through the process of download and installation.

Figure 5.4: Web-based user interface of splashRNA

6 Conclusion

We have addressed the central research question of how to bring together information from several related biological entities to obtain better models. To that end, we have presented regularization-based strategies for domain adaptation and multitask learning. Starting from a regularization-based strategy to adapt an existing model to a new task, we then formulated a general framework for multitask learning that subsumes many prominent formulations as special cases and therefore advances the understanding of their similarities and differences. In various successful applications of domain adaptation and multitask learning to problems from computational immunology, sequence biology and bioimaging, we have shown that combining information from different tasks in a multitask learning formulation is beneficial. Empirically, we learned from numerous experiments that particular attention needs to be paid to the choice of the task similarities measure, as this will greatly affect how well multitask learning strategies work with in contrast to traditional approaches. We made headway in that direction by developing a method that is able to learn or refine task similarity matrices from data using the recently proposed non-sparse multiple kernel learning, but the question of learning the task similarity matrix will clearly be subject to further research.

We would like to emphasize that multitask learning is no silver bullet (there is no *free lunch*), but it works well under certain conditions, which we call the *dynamic range of multitask learning*. A first condition is that the task must not be *saturated* in the sense that additional training examples from the same task should result in improved performance. This means that a learning curve (plotting predictive performance against number of training examples) will saturate at some point. If this is the case, we do not expect that additional out-of-domain information will improve prediction accuracy and multitask learning will not be able to improve performance on the saturated task. Thus, inspecting the learning curve may be seen as a proxy for estimating the difficulty of the problem. The second condition is that tasks should neither be too different nor too similar. If tasks are too different we are better off learning tasks independently, because as combining information between vastly different tasks will lead to *negative transfer*. On the other hand, if tasks are too similar, MTL methods fail to considerably outperform a single global model trained on the union of all available training examples. The methods we developed that automatically learn a weighting of similarity measure may be used as a strategy to avoid *negative transfer*.

As mentioned in Section 4.4, we believe the development of efficient solvers for multitask learning formulations meets growing demand in the age of Big Data. With more and more data becoming available on the web, as well as in the biological sciences, particular attention needs to be paid to the development of efficient algorithms, such as the ones presented in this thesis. To that end, methods we have presented to update

existing models in Chapter 3 and the fast dual-coordinate descent solver derived in Chapter 4 (that works in a very similar fashion to stochastic gradient descent algorithms) constitute relevant contributions.

With MHC-I binding prediction, prokaryotic gene finding and the recognition of genomic sequence signals, we have presented three examples where multitask learning methods improve prediction performance by combining information from several biological entities. In our view, however, these problems are prototypes of a broader range of biological prediction problems, where it will be beneficial to simultaneously learn models for related problems. Due to rapid progress in sequencing technology, we have access to novel ways of *looking inside a cell*, allowing us to build models for a plethora of molecular mechanisms. From our current point of view, it will take many more years of research to build better and better models for various molecular sub-systems of a cell, which can eventually be brought together to obtain a more complete understanding. In particular, the organisms-as-tasks approach investigated in this thesis may be of considerable value when bringing together information to model each of these sub-systems. For instance, large genome projects (such as the 10k Genomes project [Haussler et al., 2008]) may indeed greatly benefit from a methodology that allows the learning of joint models between poorly and well annotated organisms to increase overall annotation quality. Furthermore, we see great potential in using multitask learning in cancer research. Traditionally, cancer types are oftentimes discriminated solely based on their physical location (e.g. breast-cancer, colon cancer etc). In recent years, however, considerable research effort went into the identification of tumor sub-types and the development of targeted therapies that are specific for these sub-types (i.e. personalized medicine). Clearly, different tumor sub-types will have different degrees of similarity. Further, there may be certain sub-types of different tissues that are share some of the same mechanisms and thus may respond to similar treatment. We expect that, as more data and more fine-grained information on sub-types becomes available, multitask learning methods will help to advance the field of personalized medicine by providing the machinery for building models to predict the response of related tumor sub-types to different therapies. We expect that the MTL methods and software that we presented can indeed make a difference for this class of problems.

In a broader perspective, multitask learning may be viewed as a principle strategy of reducing cost for the generation of labeled training data. Using transfer learning, we combine available information for different tasks and may therefore reduce cost to generate additional labels. A prime example for this is the application to bioimaging presented in Section 4.3.2, where multitask learning allowed us to reuse existing $2D$ annotations in place of investing considerable time and money to generate more $3D$ annotations. The same holds true for other applications in particular in computation biology, where where generating training labels can be very expensive, but data for related problems may be readily available. By providing fast solvers for the proposed methods, we aim at achieving higher accuracies for each CPU cycle spent. From this perspective, methods presented in this thesis may be regarded as a contribution to *learning on a budget*.

Bibliography

Adams, H. and J. Koziol. "Prediction of binding to MHC class I molecules". In: *Journal of Immunological Methods* 185.2 (1995), pp. 181–190.

Agarwal, A., H. Daumé III, and S. Gerber. "Learning Multiple Tasks using Manifold Regularization". In: *Advances in neural information processing systems*. NIPS, 2010, pp. 46–54.

Ahn, W.-K. and W. F. Brewer. "Psychological Studies of Explanation-Based Learning". In: *Investigating explanation-based learning*. Springer, 1993, pp. 295–316.

Alamgir, M., M. Grosse-Wentrup, and Y. Altun. "Multitask Learning for Brain-Computer Interfaces". In: *AISTATS'10: 13th International Conference on Articial Intelligence and Statistics*. Cambridge, MA, USA: MIT Press, 2010, pp. 17–24.

Altun, Y., I. Tsochantaridis, and T. Hofmann. "Hidden Markov Support Vector Machines". In: *Proc. ICML*. Vol. 3. 2003, pp. 3–10.

Ando, R. and T. Zhang. "A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data". In: *Journal of Machine Learning Research* 6 (2005), pp. 1817–1853.

Argyriou, A., T. Evgeniou, and M. Pontil. "Convex multi-task feature learning". In: *Machine Learning* 73.3 (2008), pp. 243–272.

Argyriou, A., C. Micchelli, and M. Pontil. "When is there a representer theorem? Vector versus matrix regularizers". In: *Journal of Machine Learning Research* 10 (2009), pp. 2507–2529.

Argyriou, A., T. Evgeniou, and M. Pontil. "Multi-Task Feature Learning". In: *Advances in Neural Information Processing Systems*. MIT Press, 2006, pp. 41–48.

Aronszajn, N. "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

Bakker, B. and T. Heskes. "Task clustering and gating for bayesian multitask learning". In: *Journal of Machine Learning Research* 4 (2003), pp. 83–99.

Bauschke, H. and Y. Lucet. "What is a Fenchel conjugate?" In: *Notices of the AMS* 59 (2012), pp. 44–46.

Bauschke, H. H. and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in mathematics. New York: Springer, 2011.

Baxter, J. "A Model of Inductive Bias Learning". In: *Journal of Artificial Intelligence Research* 12 (2000), pp. 149–198.

Behr, J., A. Kahles, Y. Zhong, V. T. Sreedharan, P. Drewe, and G. Rätsch. "MITIE: Simultaneous RNA-Seq-based transcript identification and quantification in multiple samples". In: *Bioinformatics* 29.20 (2013), pp. 2529–2538.

Ben-David, S. and R. Schuller. "Exploiting task relatedness for multiple task learning". In: *Lecture notes in computer science* (2003), pp. 567–580.

Bennett, K. and E. Parrado-Hernández. "The interplay of optimization and machine learning research". In: *Journal of Machine Learning Research* 7 (2006), pp. 1265–1281.

Bennett, K. and O. L. Mangasarian. "Robust linear programming discrimination of two linearly inseparable sets". In: *Optimization Methods and Software* 1.1 (1992), pp. 23–34.

Berman, H. M., J Westbrook, Z Feng, G Gilliland, T. N. Bhat, H Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Data Bank". In: *Nucleic Acids Research* 28 (2000), pp. 235–242.

Bi, J., T. Xiong, S. Yu, M. Dundar, and R. B. Rao. "An improved multi-task learning approach with applications in medical diagnosis". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 117–132.

Bickel, S., J. Bogojeska, T. Lengauer, and T. Scheffer. "Multi-task learning for HIV therapy screening". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 56–63.

Blanchard, G., G. Lee, and C. Scott. "Generalizing from Several Related Classification Tasks to a New Unlabeled Sample". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2178–2186.

Bohnert, R., J. Behr, and G. Rätsch. "Transcript quantification with RNA-Seq data". In: *BMC Bioinformatics* 10.Suppl 13 (2009), P5.

Bonilla, E. V., K. M. A. Chai, and C. K. I. Williams. "Multi-task Gaussian Process Prediction". In: *NIPS*. Curran Associates, Inc., 2007.

Boser, B., I. Guyon, and V. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory COLT 92*. COLT '92 6.8 (1992), pp. 144–152.

Boyd, S.P. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Breiman, L. "Statistical Modeling: The Two Cultures". In: *Statistical Science* 16.3 (2001), pp. 199–231.

Brown, P. J. and J. V. Zidek. "Adaptive multivariate ridge regression". In: *The Annals of Statistics* 8.1 (1980), pp. 64–74.

Caruana, R. "Multitask Learning: A Knowledge-Based Source of Inductive Bias". In: *ICML*. Morgan Kaufmann, 1993, pp. 41–48.

Caruana, R. "Multitask learning". In: *Machine Learning* 28.1 (1997), pp. 41–75.

Cawley, G. C. and N. L. C. Talbot. "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation". In: *Journal of Machine Learning Research* 11.3 (2010), pp. 2079–2107.

Chang, C. and C. Lin. "LIBSVM". In: *ACM Transactions on Intelligent Systems and Technology* 2.3 (2011), pp. 1–27.

Chapelle, O. "Training a support vector machine in the primal". In: *Neural Computation* 19.5 (2007), pp. 1155–1178.

Chen, A. H. and Z.-W. Huang. "A new multi-task learning technique to predict classification of leukemia and prostate cancer". In: *Medical Biometrics*. Springer, 2010, pp. 11–20.

Chen, J., L. Tang, J. Liu, and J. Ye. "A convex formulation for learning shared structures from multiple tasks". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. New York, New York, USA: ACM Press, 2009, pp. 1–8.

Chervonenkis, A. Y. and V. N. Vapnik. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*. 1971.

Collins, M., R. E. Schapire, and Y. Singer. "Logistic regression, AdaBoost and Bregman distances". In: *Machine Learning* 48.1-3 (2002), pp. 253–285.

Cortes, C. and V. Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

Cressie, N. "Statistics for spatial data". In: *Terra Nova* 4.5 (1992), pp. 613–617.

Crick, F. "Central dogma of molecular biology." In: *Nature* 227.5258 (1970), pp. 561–563.

Dahlmeier, D. and H. T. Ng. "Domain adaptation for semantic role labeling in the biomedical domain". In: *Bioinformatics* 26.8 (2010), pp. 1098–1104.

Dai, W., Q. Yang, G.-R. Xue, and Y. Yu. "Boosting for transfer learning". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 193–200.

Daumé, H. "Frustratingly easy domain adaptation". In: *Annual meeting-association for computational linguistics*. Vol. 45. 2007, pp. 256–263.

Daumé III, H. "Bayesian multitask learning with latent hierarchies". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2009, pp. 135–142.

Dean, J. and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

Do, T.-M.-T. "Regularized Bundle Methods for Large-scale Learning Problems with an Application to Large Margin Training of Hidden Markov Models". PhD thesis. l'Université Pierre & Marie Curie, 2010.

Dönnes, P. and A. Elofsson. "Prediction of MHC class I binding peptides, using SVMHC". In: *BMC Bioinformatics* 3 (2002), p. 25.

Evgeniou, T., C. Micchelli, and M. Pontil. "Learning multiple tasks with kernel methods". In: *Journal of Machine Learning Research* 6.1 (2005), pp. 615–637.

Evgeniou, T. and M. Pontil. "Regularized multi-task learning". In: *International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 109–117.

Fellmann, C. et al. "Functional identification of optimized RNAi triggers using a massively parallel sensor assay." In: *Molecular cell* 41.6 (2011), pp. 733–746.

Fitzgibbon, A., M. Pilu, and R. Fisher. "Direct least square fitting of ellipses". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21.5 (1999), pp. 476–480.

Foster, I. *Designing and building parallel programs*. Vol. 95. Addison-Wesley Reading, 1995.

Franc, V. and S. Sonnenburg. "Optimized Cutting Plane Algorithm for Large-Scale Risk Minimization". In: *Journal of Machine Learning Research* 10 (2009), pp. 2157–2192.

Fürnkranz, J. and T. Joachims, eds. *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Omnipress, 2010. ISBN: 978-1-60558-907-7.

Gehler, P. and S. Nowozin. "Infinite Kernel Learning". In: *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*. 2008.

Geman, S., E. Bienenstock, and R. Doursat. "Neural networks and the bias/variance dilemma". In: *Neural Computation* 4.1 (1992), pp. 1–58.

Gentile, C. and M. K. Warmuth. "Linear Hinge Loss and Average Margin". In: *Advances in Neural Information Processing Systems*. The MIT Press, 1998, pp. 225–231.

Golub, G. H., M. Heath, and G. Wahba. "Generalized cross-validation as a method for choosing a good ridge parameter". In: *Technometrics* 21.2 (1979), pp. 215–223.

Gönen, M. and E. Alpaydin. "Multiple Kernel Learning Algorithms". In: *Journal of Machine Learning Research* 12 (2011), pp. 2211–2268.

Görnitz*, N., C. Widmer*, G. Zeller, A. Kahles, S. Sonnenburg, and G. Rätsch. "Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation". In: *NIPS*. 2011, pp. 2690–2698.

Gretton, A., A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. "Covariate shift by kernel mean matching". In: *Dataset shift in machine learning* 3.4 (2009), p. 5.

Guyon, I. "A practical guide to model selection". In: *Proceedings of the machine learning summer school*. Vol. 11. Springer, 2009, pp. 1–37.

Guyon, I., A. Saffari, G. Dror, and G. Cawley. "Model Selection: Beyond the Bayesian/Frequentist Divide". In: *Journal of Machine Learning Research* 11 (2010), pp. 61–87.

Haussler, D. et al. "Genome 10K: a proposal to obtain whole-genome sequence for 10,000 vertebrate species." In: *The Journal of Heredity* 100.6 (2008), pp. 659–674.

Hazan, T. and R. Urtasun. "A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction". In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2010, pp. 838–846.

Heckerman, D., C. Kadie, and J. Listgarten. "Leveraging information across HLA alleles/supertypes improves epitope prediction". In: *Journal of Computational Biology* 14.6 (2007), pp. 736–746.

Heckman, J. J. "Sample selection bias as a specification error". In: *Econometrica: Journal of the Econometric Society* (1979), pp. 153–161.

Heinrich, S. et al. "Determinants of robustness in spindle assembly checkpoint signalling." In: *Nature Cell Biology* 15.11 (2013), pp. 1328–1339.

Henikoff, S and J. G. Henikoff. "Amino acid substitution matrices from protein blocks". In: *Proceedings of the National Academy of Sciences of the United States of America* 89.22 (1992), pp. 10915–10919.

Hintjens, P. *ZeroMQ: Messaging for Many Applications*. O'Reilly, 2013.

Hinton, G. E., S. Osindero, and Y.-W. Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

Hoerl, A. E. "Application of ridge analysis to regression problems". In: *Chemical Engineering Progress* 58.3 (1962), pp. 54–59.

Hsieh, C., K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. "A dual coordinate descent method for large-scale linear SVM". In: *Proceedings of the 25th international conference on Machine learning* (2008), pp. 408–415.

Institute, N. H. G. R. *Understanding the Human Genome Project*. 2011. URL: `http://www.genome.gov/EdKit/bio2j.html`.

Jacob, L. and J. Vert. "Efficient peptide-MHC-I binding prediction for alleles with few known binders." In: *Bioinformatics (Oxford, England)* 24.3 (2008), pp. 358–66.

Jacob, L., F. Bach, and J. Vert. "Clustered multi-task learning: A convex formulation". In: *Arxiv preprint arXiv:0809.2085* (2008).

Jacob, L. and J.-P. Vert. "Protein-ligand interaction prediction: an improved chemogenomics approach". In: *Bioinformatics* 24.19 (2008), pp. 2149–2156.

Jalali, A., S. Sanghavi, C. Ruan, and P. K. Ravikumar. "A dirty model for multi-task learning". In: *Advances in Neural Information Processing Systems*. 2010, pp. 964–972.

Jean, G., A. Kahles, V. T. Sreedharan, F. D. Bona, and G. Rätsch. "RNA-Seq Read Alignments with PALMapper". In: *Current protocols in bioinformatics* (2010), pp. 11–6.

Jebara, T. "Multi-task feature and kernel selection for SVMs". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 55.

Joachims, T. "Making large-Scale SVM Learning Practical". In: *Advances in Kernel Methods Support Vector Learning*. Advances in Kernel Methods - Support Vector Learning (1999), pp. 169–184.

Keerthi, S. S. and D. DeCoste. "A modified finite Newton method for fast solution of large scale linear SVMs". In: *Journal of Machine Learning Research* 6 (2005), pp. 341–361.

Kimeldorf, G. and G. Wahba. "Some results on Tchebycheffian spline functions". In: *Journal of Mathematical Analysis and Applications* 33.1 (1971), pp. 82–95.

Kloft, M., U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. "Efficient and Accurate Lp-Norm Multiple Kernel Learning". In: *Advances in Neural Information Processing Systems 22*. MIT Press, 2009, pp. 997–1005.

Kloft, M., U. Brefeld, S. Sonnenburg, and A. Zien. "lp-Norm Multiple Kernel Learning". In: *Journal of Machine Learning Research* 12 (2011), pp. 953–997.

Kloft, M., U. Rückert, and P. L. Bartlett. "A Unifying View of Multiple Kernel Learning". In: *ECML/PKDD (2)*. Vol. 6322. Lecture Notes in Computer Science. Springer, 2010, pp. 66–81.

Koenker, R. *Quantile regression*. 38. Cambridge university press, 2005.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *ICML*. Morgan Kaufmann, 2001, pp. 282–289.

Laubinger, S. et al. "At-TAX: a whole genome tiling array resource for developmental expression analysis and transcript identification in Arabidopsis thaliana". In: *Genome Biology* 9.7 (2008), pp. 1–16.

Leiva-Murillo, J. M., L. Gómez-Chova, and G. Camps-Valls. "Multitask Remote Sensing Data Classification". In: *IEEE T. Geoscience and Remote Sensing* 51.1 (2013), pp. 151–161.

Leslie, C. S., E. Eskin, and W. S. Noble. "The spectrum kernel: A string kernel for SVM protein classification." In: *Pacific symposium on biocomputing*. Vol. 7. World Scientific. 2002, pp. 566–575.

Lewis, A. "Convex analysis on the Hermitian matrices". In: *SIAM Journal on Optimization* (1996).

Lippert, C., O. Weissbrod, C. Widmer, N. Fusi, C. Kadie, R. Davidson, J. Listgarten, and D. Heckerman. "An evaluation of methods for variant and principle-component selection with mixed models for GWAS". In: *Scientific Reports* (2014), (in review).

Lippert, C., J. Xiang, D. Horta, C. Widmer, C. Kadie, J. Listgarten, and D. Heckerman. "Greater Power and Computational Efficiency for Testing Sets of Markers Genome-Wide". In: *Bioinformatics* (2014), (in revision).

Lisitsyn, S., C. Widmer, and F. Garcia. "Tapkee: An Efficient Dimension Reduction Library". In: *Journal of machine learning research* 14 (2013), pp. 2355–2359.

Liu, J., S. Ji, and J. Ye. "Multi-Task Feature Learning Via Efficient L2,1-Norm Minimization". In: *UAI 2009*. UAI '09. AUAI Press, 2009, pp. 339–348.

Liu, Q., Q. Xu, V. W. Zheng, H. Xue, Z. Cao, and Q. Yang. "Multi-task learning for cross-platform siRNA efficacy prediction: an in-silico study". In: *BMC bioinformatics* 11.1 (2010), pp. 1–16.

Lou, X. and F. Hamprecht. "Structured Learning from Partial Annotations". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. ICML 2012. Omnipress, 2012, pp. 1519–1526. arXiv: `1206.6421`.

Lou, X., U. Koethe, J. Wittbrodt, and F. A Hamprecht. "Learning to Segment Dense Cell Nuclei with Shape Prior". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1012–1018.

Lou, X., C. Widmer, M. Kang, G. Rätsch, and A. Hadjantonakis. "Structured Domain Adaptation Across Imaging Modality: How 2D Data Helps 3D Inference". In: *NIPS Machine Learning in Computational Biology (NIPS-MLCB)*. 2012.

Lou, X. and F. A. Hamprecht. "Structured Learning for Cell Tracking". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1296–1304.

MacKay, D. J. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), pp. 448–472.

Makhorin, A. *GLPK (GNU linear programming kit)*. 2006.

Mei, S., W. Fei, and S. Zhou. "Gene ontology based transfer learning for protein subcellular localization". In: *BMC bioinformatics* 12.1 (2011), pp. 1–12.

Micchelli, C. and M. Pontil. "On learning vector-valued functions". In: *Neural Computation* 17.1 (2005), pp. 177–204.

Mika, S., G. Rätsch, and K.-R. Müller. "A Mathematical Programming Approach to the Kernel Fisher Algorithm". In: *Advances in neural information processing systems*. 2000, pp. 591–597.

Moll, A., A. Hildebrandt, H.-P. Lenhof, and O. Kohlbacher. "BALLView: a tool for research and education in molecular modeling". In: *Bioinformatics* 22.3 (2006), pp. 365–366.

Neural Networks: Tricks of the Trade - Second Edition. Vol. 7700. Lecture Notes in Computer Science. Springer, 2012.

Müller, K.-R., A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. "Predicting Time Series with Support Vector Machines". In: *ICANN*. 1997, pp. 999–1004.

Müller, K.-R., S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. "An introduction to kernel-based learning algorithms". In: *IEEE Transactions on Neural Networks* 12.2 (2001), pp. 181–201.

Nashed, M. and G. Wahba. "Regularization and approximation of linear operator equations in reproducing kernel spaces". In: *American Mathematical Society* 80.6 (1974), pp. 1213–1218.

Nguyen, X., M. J. Wainwright, and M. I. Jordan. "Divergences, surrogate loss functions and experimental design". In: *Advances in Neural Information Processing Systems*. 2005, pp. 1011–1018.

Nie, F., H. Huang, X. Cai, and C. H. Q. Ding. "Efficient and Robust Feature Selection via Joint l2,1-Norms Minimization". In: *Advances in Neural Information Processing Systems*. 2010, pp. 1813–1821.

Nowozin, S. and C. H. Lampert. "Structured Learning and Prediction in Computer Vision". In: *Foundations and Trends® in Computer Graphics and Vision* 6 (2011), pp. 185–365.

Obozinski, G., B. Taskar, and M. Jordan. "Joint covariate selection and joint subspace selection for multiple classification problems". In: *Statistics and Computing* 20.2 (2010), pp. 231–252.

Obozinski, G., B. Taskar, and M. Jordan. *Joint covariate selection for grouped classification*. Tech. rep. ID: 743. Department of Statistics, U. of California Berkeley, 2007.

Pan, S. and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on Knowledge and Data Engineering* (2009), pp. 1345–1359.

Pelossof*, R., V. Thapar*, C. Widmer*, C. Fellmann, A. Greaves-Tunnell, G. Rätsch, S. Lowe, and C. Leslie. "splashRNA: Large-scale learning produces an accurate predictor of potent shRNAs and identifies novel shRNA processing features." In: *to be determined* (2014, in prep.).

Peters, B et al. "A Community Resource Benchmarking Predictions of Peptide Binding to MHC-I Molecules". In: *PLoS Comput Biol* 2.6 (2006), e65.

Platt, J. "Advances in kernel methods". In: Cambridge, MA, USA: MIT Press, 1999. Chap. Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

Poggio, T. and F. Girosi. "Regularization algorithms for learning that are equivalent to multilayer networks." In: *Science* 247.4945 (1990), pp. 978–982.

Rammensee, H., J. Bachmann, N. Emmerich, O. Bachor, and S. Stevanovic. "SYFPEITHI: Database for MHC ligands and peptide motifs". In: *Immunogenetics* 50 (1999), pp. 213–219.

Rätsch, G. and S. Sonnenburg. "Large Scale Hidden Semi-Markov SVMs". In: *Advances in Neural Information Processing Systems*. MIT Press, 2006, pp. 1161–1168.

Rätsch, G., S. Sonnenburg, and B. Schölkopf. "RASE: recognition of alternatively spliced exons in *C.elegans*". In: *ISMB (Supplement of Bioinformatics)*. 2005, pp. 369–377.

Rätsch, G., S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. J. Sommer, and B. Schölkopf. "Improving the *Caenorhabditis elegans* Genome Annotation Using Machine Learning". In: *PLoS Computational Biology* 3.2 (2007), pp. 313–322.

Reche, P. A., J.-P. Glutting, and E. L. Reinherz. "Prediction of MHC class I binding peptides using profile motifs". In: *Human Immunology* 63.9 (2002), pp. 701–709.

Rieck, K. and P. Laskov. "Linear-time computation of similarity measures for sequential data". In: *Journal of Machine Learning Research* 9 (2008), pp. 23–48.

Rifkin, R. M. and R. A. Lippert. "Value Regularization and Fenchel Duality". In: *Journal of Machine Learning Research* 8 (2007), pp. 441–479.

Rockafellar, R. T. *Convex Analysis, volume 28 of Princeton Mathematics Series*. 1970.

Rockafellar, R. T. "Lagrange multipliers and optimality". In: *SIAM Rev.* 35.2 (1993), pp. 183–238.

Rosin, P. "Assessing error of fit functions for ellipses". In: *Graphical models and image processing* 58.5 (1996), pp. 494–502.

Rosset, S., J. Zhu, and T. Hastie. "Boosting as a regularized path to a maximum margin classifier". In: *Journal of Machine Learning Research* 5 (2004), pp. 941–973.

Roth, V. and B. Fischer. "The Group-Lasso for generalized linear models: uniqueness of solutions and efficient algorithms". In: *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML 2008)*. Vol. 307. ACM, 2008, pp. 848–855.

Samek, W., A. Binder, and M. Kawanabe. "Multi-task Learning via Non-sparse Multiple Kernel Learning". In: *Computer Analysis of Images and Patterns*. Lecture Notes in Computer Science 6854.c (2011), pp. 335–342.

Samek, W., C. Vidaurre, K.-R. Müller, and M. Kawanabe. "Stationary common spatial patterns for brain–computer interfacing". In: *Journal of Neural Engineering* 9.2 (2012), p. 026013.

Schölkopf, B. and A. Smola. *Learning with Kernels*. Vol. 64. Adaptive Computation and Machine Learning. MIT Press, 2002. Chap. Robust Est, pp. 489–489.

Schölkopf, B., A. J. Smola, and K.-R. Müller. "Kernel Principal Component Analysis". In: *ICANN*. Vol. 1327. Lecture Notes in Computer Science. Springer, 1997, pp. 583–588.

Schölkopf, B., R. Herbrich, and A. J. Smola. "A generalized representer theorem". In: *Computational learning theory*. Springer. 2001, pp. 416–426.

Schölkopf, B., J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

Schölkopf, B., J. C. Platt, and T. Hoffman, eds. *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. MIT Press, 2007. ISBN: 0-262-19568-2.

Schölkopf, B., Z. Luo, and V. Vovk. *Empirical Inference - Festschrift in Honor of Vladimir N. Vapnik*. Springer, Heidelberg, 2013.

Schultheiß, S. J., W. Busch, J. Lohmann, O. Kohlbacher, and G. Rätsch. "KIRMES: kernel-based identification of regulatory modules in euchromatic sequences". In: *Bioinformatics* 25.16 (2009), pp. 2126–2133.

Schweikert*, G., C. Widmer*, B. Schölkopf, and G. Rätsch. "An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis". In: *Advances in Neural Information Processing Systems 21*. NIPS, 2009, pp. 1433–1440.

Schweikert, G. et al. "mGene: accurate SVM-based gene finding with an application to nematode genomes." In: *Genome research* 19.11 (2009), pp. 2133–43.

Shawe-Taylor, J., R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, eds. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. 2011.

Shi, Q., J. Petterson, G. Dror, J. Langford, A. Smola, and S. Vishwanathan. "Hash kernels for structured data". In: *Journal of Machine Learning Research* 10 (2009), pp. 2615–2637.

Smola, A., S. Vishwanathan, and Q. Le. "Bundle Methods for Machine Learning". In: *Advances in Neural Information Processing Systems 20*. 2008, pp. 1377–1384.

Smola, A. J., B. Schölkopf, and K.-R. Müller. "The connection between regularization operators and support vector kernels". In: *Neural Networks* 11.4 (1998), pp. 637–649.

Sonnenburg, S., G. Rätsch, C. Schäfer, and B. Schölkopf. "Large Scale Multiple Kernel Learning". In: *Journal of Machine Learning Research* 7 (2006), pp. 1531–1565.

Sonnenburg, S. "Machine Learning for Genomic Sequence Analysis". supervised by K.-R. Müller and G. Rätsch. PhD thesis. TU-Berlin, 2008.

Sonnenburg, S. and V. Franc. "COFFIN: A Computational Framework for Linear SVMs". In: *ICML*. Omnipress, 2010, pp. 999–1006.

Sonnenburg, S., G. Schweikert, P. Philips, J. Behr, and G. Rätsch. "Accurate splice site prediction using support vector machines". In: *BMC Bioinformatics* 8.Suppl-10 (2007).

Sonnenburg, S., G. Rätsch, and K. Rieck. "Large Scale Learning with String Kernels". In: *Large Scale Kernel Machines*. Cambridge, MA.: MIT Press, 2007, pp. 73–103.

Sonnenburg, S., M. L. Braun, C. S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, F. Pereira, and C. E. Rasmussen. "The Need for Open Source Software in Machine Learning". In: *Journal of Machine Learning Research* 8 (2007), pp. 2443–2466.

Sonnenburg, S., A. Zien, P. Philips, and G. Rätsch. "POIMs: positional oligomer importance matrices - understanding support vector machine-based signal detectors". In: *ISMB (Supplement of Bioinformatics)*. Vol. 24. 13. Oxford Univ Press, 2008, pp. 6–14.

Sonnenburg, S. et al. "The SHOGUN machine learning toolbox". In: *Journal of Machine Learning Research* 99 (2010), pp. 1799–1802.

Spencer, W. et al. "A spatial and temporal map of C. elegans gene expression". In: *Genome Research* 21.2 (2011), pp. 325–341.

Spiegelhalter, D. J., A. P. Dawid, S. L. Lauritzen, and R. G. Cowell. "Bayesian analysis in expert systems". In: *Statistical science* (1993), pp. 219–247.

Srivastava, V. and T. Dwivedi. "Estimation of seemingly unrelated regression equations: A brief survey". In: *Journal of Econometrics* 10.1 (1979), pp. 15–32.

Stanke, M. and B. Morgenstern. "AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints". In: *Nucleic Acids Research* 33.Web-Server-Issue (2005), pp. 465–467.

Steinwart, I., D. Hush, and C. Scovel. "Training SVMs without offset". In: *Journal of Machine Learning Research* 12 (2011), pp. 141–202.

Stone, M. "Cross-Validatory Choice and Assessment of Statistical Predictions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 36.2 (1974), pp. 111–147.

Sugiyama, M. and M. Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press, 2012.

Sugiyama, M. and K.-R. Müller. "Input-dependent estimation of generalization error under covariate shift". In: *Statistics & Decisions* 23.4/2005 (2005), pp. 249–279.

Sugiyama, M., M. Krauledat, and K.-R. Müller. "Covariate shift adaptation by importance weighted cross validation". In: *Journal of Machine Learning Research* 8 (2007), pp. 985–1005.

Tamada, Y., H. Bannai, S. Imoto, T. Katayama, M. Kanehisa, and S. Miyano. "Utilizing evolutionary information and gene expression data for estimating gene networks with Bayesian network models". In: *Journal of bioinformatics and computational biology* 3.06 (2005), pp. 1295–1313.

Teo, C., S. Vishwanathan, A.Smola, and Q. Le. "Bundle Methods for Regularized Risk Minimization". In: *Journal of Machine Learning Research* 11 (2010), pp. 311–365.

Teo, C. H., A. Smola, S. V. N. Vishwanathan, and Q. V. Le. "A scalable modular convex solver for regularized risk minimization". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 727–736.

Thrun, S. "Is learning the n-th thing any easier than learning the first?" In: *Advances in neural information processing systems* (1996), pp. 640–646.

Tibshirani, R. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B Methodological*. B 58.1 (1996), pp. 267–288.

Tikhonov, A. N. "On the stability of inverse problems". In: *Doklady Akademii Nauk Sssr* 39.5 (1943), pp. 195–198.

Tomioka, R. *Three strategies to derive a dual problem*. 2010. URL: `http://www.ibis.t.u-tokyo.ac.jp/RyotaTomioka/Notes/DerivingDual?action=AttachFile&do=get&target=Three-strategies-20100517.pdf`.

Toussaint, N. and O. Kohlbacher. "Towards in silico design of epitope-based vaccines." In: *Expert Opinion on Drug Discovery* 4.10 (2009), pp. 1047–1060.

Toussaint, N., C. Widmer, O. Kohlbacher, and G. Rätsch. "Exploiting physico-chemical properties in string kernels." In: *BMC bioinformatics* 11 Suppl 8.Suppl 8 (2010), S7.

Tseng, P. "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *J. Optim. Theory Appl.* 109.3 (2001), pp. 475–494.

Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun. "Large Margin Methods for Structured and Interdependent Output Variables". In: *Journal of Machine Learning Research* 6 (2005), pp. 1453–1484.

Vanderbei, R. J. "Interior-Point Methods: Algorithms and Formulations". In: *ORSA Journal on Computing* 6.1 (1994), pp. 32–34.

Vapnik, V. N. and A. Y. Chervonenkis. "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". In: *Theory of Probability and its Applications* 16.2 (1971), pp. 264–280.

Vapnik, V. "Estimation of Dependences Based on Empirical Data [in Russian]". In: *Nauka* (1979).

Vapnik, V., S. E. Golowich, and A. Smola. "Support vector method for function approximation, regression estimation, and signal processing". In: *Advances in neural information processing systems* (1997), pp. 281–287.

Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, 1995.

Vert, J.-P., N. Foveau, C. Lajaunie, and Y. Vandenbrouck. "An accurate and interpretable model for siRNA efficacy prediction." In: *BMC bioinformatics* 7.1 (2006), p. 520.

Vishwanathan, S. V. N., Z. Sun, N. Ampornpunt, and M. Varma. "Multiple Kernel Learning and the SMO Algorithm". In: *Advances in Neural Information Processing Systems 23*. 2010, pp. 2361–2369.

Von Luxburg, U. "A Tutorial on Spectral Clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416.

Watkins, C. "Dynamic alignment kernels". In: *Advances in Neural Information Processing Systems* (1999), pp. 39–50.

Widmer, C and C. Ong. *pythongrid high-level python wapper for the Sun Grid Engine (SGE) using DRMAA and ZMQ*. 2013. URL: `https://code.google.com/p/pythongrid/`.

Widmer, C. and G. Rätsch. "Multitask Learning in Computational Biology". In: *JMLR W&CP. ICML 2011 Unsupervised and Transfer Learning Workshop.* 27 (2012), pp. 207–216.

Widmer, C., Y. Altun, and G. Rätsch. "Inferring latent task structure for Multitask Learning by Multiple Kernel Learning." In: *BMC bioinformatics* 11 Suppl 8.Suppl 8 (2010), S5.

Widmer, C., J. Leiva, Y. Altun, and G. Rätsch. "Leveraging Sequence Classification by Taxonomy-based Multitask Learning". In: *Research in Computational Molecular Biology*. Springer, 2010, pp. 522–534.

Widmer, C., N. Toussaint, Y. Altun, O. Kohlbacher, and G. Rätsch. "Novel machine learning methods for MHC Class I binding prediction". In: *Pattern Recognition in Bioinformatics*. Springer, 2010, pp. 98–109.

Widmer, C., P. Drewe, X. Lou, S. Umrania, S. Heinrich, and G. Rätsch. "GRED: Graph-Regularized 3D Shape Reconstruction from Highly Anisotropic and Noisy Images". In: *arXiv preprint arXiv:1309.4426* (2013).

Widmer, C., M. Kloft, and G. Rätsch. "Multi-task Learning for Computational Biology: Overview and Outlook". In: *Empirical Inference - Festschrift in Honor of Vladimir N. Vapnik*. Springer, 2013, pp. 117–127.

Widmer, C., M. Kloft, X. Lou, and G. Rätsch. "Regularization-based Multitask Learning With applications to Genome Biology and Biomedical Imaging." In: *Künstliche Intelligenz* (2013).

Widmer, C., M. Kloft, and G. Raetsch. "A general framework for mutitask multiple kernel learning." In: *Journal of Machine Learning Research* (2014, in prep.).

Widmer, C., M. Kloft, N. Görnitz, and G. Rätsch. "Efficient Training of Graph-Regularized Multitask SVMs". In: *ECML/PKDD (1)*. Vol. 7523. Lecture Notes in Computer Science. Springer, 2012, pp. 633–647.

Wikipedia. *DNA Transcription*. 2011. URL: `http://upload.wikimedia.org/wikipedia/commons/3/36/DNA_transcription.svg`.

Williams, C. K. "Prediction with Gaussian processes: From linear regression to linear prediction and beyond". In: *Learning in graphical models*. Springer, 1998, pp. 599–621.

Xu, Q. and Q. Yang. "A Survey of Transfer and Multitask Learning in Bioinformatics." In: *JCSE* 5.3 (2011), pp. 257–268.

Xu, Q., S. J. Pan, H. H. Xue, and Q. Yang. "Multitask learning for protein subcellular location prediction". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 8.3 (2011), pp. 748–759.

Xu, Z., R. Jin, H. Yang, I. King, and M. R. Lyu. "Simple and Efficient Multiple Kernel Learning by Group Lasso". In: *ICML*. Omnipress, 2010, pp. 1175–1182.

Xue, Y., X. Liao, L. Carin, and B. Krishnapuram. "Multi-Task Learning for Classification with Dirichlet Process Priors". In: *Journal of Machine Learning Research* 8 (2007), pp. 35–63.

Yu, Y., H. Cheng, D. Schuurmans, and C. Szepesvari. "Characterizing the Representer Theorem". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 570–578.

Zeller, G., S. Henz, C. Widmer, T. Sachsenberg, G. Rätsch, D. Weigel, and S. Laubinger. "Stress-induced changes in the Arabidopsis thaliana transcriptome analyzed using whole-genome tiling arrays." In: *The Plant journal : for cell and molecular biology* 58.6 (2009), pp. 1068–82.

Zellner, A. "An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias". In: *Journal of the American statistical Association* 57.298 (1962), pp. 348–368.

Zhang, K., J. W. Gray, and B. Parvin. "Sparse multitask regression for identifying common mechanism of response to therapeutic targets". In: *Bioinformatics* 26.12 (2010), pp. i97–i105.

Zhang, Y. and D.-Y. Yeung. "A Convex Formulation for Learning Task Relationships in Multi-Task Learning". In: *UAI*. AUAI Press, 2010, pp. 733–442.

Zhang, Y. and D.-Y. Yeung. "Multi-Task Learning using Generalized t-Process". In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 964–971.

Zhou, J., J. Chen, and J. Ye. "Clustered Multi-Task Learning Via Alternating Structure Optimization". In: *Advances in Neural Information Processing Systems*. 2011, pp. 702–710.

Zhu, J., S. Rosset, T. Hastie, and R. J. Tibshirani. "1-Norm Support Vector Machines". In: *Advances in Neural Information Processing Systems*. 2004, pp. 49–56.

Zien, A., G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. "Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites". In: *Bioinformatics* 16.9 (2000), pp. 799–807.

Publication List

Complete list of publications by the author. Equal contributions are indicated by *.

Publications at Conferences:

Görnitz*, N., C. Widmer*, G. Zeller, A. Kahles, S. Sonnenburg, and G. Rätsch. "Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation". In: *NIPS*. 2011, pp. 2690–2698.

Lou, X., C. Widmer, M. Kang, G. Rätsch, and A. Hadjantonakis. "Structured Domain Adaptation Across Imaging Modality: How 2D Data Helps 3D Inference". In: *NIPS Machine Learning in Computational Biology (NIPS-MLCB)*. 2012.

Schweikert*, G., C. Widmer*, B. Schölkopf, and G. Rätsch. "An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis". In: *Advances in Neural Information Processing Systems 21*. NIPS, 2009, pp. 1433–1440.

Widmer, C., J. Leiva, Y. Altun, and G. Rätsch. "Leveraging Sequence Classification by Taxonomy-based Multitask Learning". In: *Research in Computational Molecular Biology*. Springer, 2010, pp. 522–534.

Widmer, C., N. Toussaint, Y. Altun, O. Kohlbacher, and G. Rätsch. "Novel machine learning methods for MHC Class I binding prediction". In: *Pattern Recognition in Bioinformatics*. Springer, 2010, pp. 98–109.

Widmer, C., M. Kloft, N. Görnitz, and G. Rätsch. "Efficient Training of Graph-Regularized Multitask SVMs". In: *ECML/PKDD (1)*. Vol. 7523. Lecture Notes in Computer Science. Springer, 2012, pp. 633–647.

Journal Publications:

Heinrich, S. et al. "Determinants of robustness in spindle assembly checkpoint signalling." In: *Nature Cell Biology* 15.11 (2013), pp. 1328–1339.

Laubinger, S. et al. "At-TAX: a whole genome tiling array resource for developmental expression analysis and transcript identification in Arabidopsis thaliana". In: *Genome Biology* 9.7 (2008), pp. 1–16.

Lippert, C., O. Weissbrod, C. Widmer, N. Fusi, C. Kadie, R. Davidson, J. Listgarten, and D. Heckerman. "An evaluation of methods for variant and principle-component selection with mixed models for GWAS". In: *Scientific Reports* (2014), (in review).

Lippert, C., J. Xiang, D. Horta, C. Widmer, C. Kadie, J. Listgarten, and D. Heckerman. "Greater Power and Computational Efficiency for Testing Sets of Markers Genome-Wide". In: *Bioinformatics* (2014), (in revision).

Lisitsyn, S., C. Widmer, and F. Garcia. "Tapkee: An Efficient Dimension Reduction Library". In: *Journal of machine learning research* 14 (2013), pp. 2355–2359.

Pelossof*, R., V. Thapar*, C. Widmer*, C. Fellmann, A. Greaves-Tunnell, G. Rätsch, S. Lowe, and C. Leslie. "splashRNA: Large-scale learning produces an accurate predictor of potent shRNAs and identifies novel shRNA processing features." In: *to be determined* (2014, in prep.).

Sonnenburg, S. et al. "The SHOGUN machine learning toolbox". In: *Journal of Machine Learning Research* 99 (2010), pp. 1799–1802.

Spencer, W. et al. "A spatial and temporal map of C. elegans gene expression". In: *Genome Research* 21.2 (2011), pp. 325–341.

Toussaint, N., C. Widmer, O. Kohlbacher, and G. Rätsch. "Exploiting physico-chemical properties in string kernels." In: *BMC bioinformatics* 11 Suppl 8.Suppl 8 (2010), S7.

Widmer, C. and G. Rätsch. "Multitask Learning in Computational Biology". In: *JMLR W&CP. ICML 2011 Unsupervised and Transfer Learning Workshop.* 27 (2012), pp. 207–216.

Widmer, C., Y. Altun, and G. Rätsch. "Inferring latent task structure for Multitask Learning by Multiple Kernel Learning." In: *BMC bioinformatics* 11 Suppl 8.Suppl 8 (2010), S5.

Widmer, C., P. Drewe, X. Lou, S. Umrania, S. Heinrich, and G. Rätsch. "GRED: Graph-Regularized 3D Shape Reconstruction from Highly Anisotropic and Noisy Images". In: *arXiv preprint arXiv:1309.4426* (2013).

Widmer, C., M. Kloft, and G. Rätsch. "Multi-task Learning for Computational Biology: Overview and Outlook". In: *Empirical Inference - Festschrift in Honor of Vladimir N. Vapnik*. Springer, 2013, pp. 117–127.

Widmer, C., M. Kloft, X. Lou, and G. Rätsch. "Regularization-based Multitask Learning With applications to Genome Biology and Biomedical Imaging." In: *Künstliche Intelligenz* (2013).

Widmer, C., M. Kloft, and G. Raetsch. "A general framework for mutitask multiple kernel learning." In: *Journal of Machine Learning Research* (2014, in prep.).

Zeller, G., S. Henz, C. Widmer, T. Sachsenberg, G. Rätsch, D. Weigel, and S. Laubinger. "Stress-induced changes in the Arabidopsis thaliana transcriptome analyzed using whole-genome tiling arrays." In: *The Plant journal : for cell and molecular biology* 58.6 (2009), pp. 1068–82.

Appendix

A Image licenses

A.1 Sources for Figure 1.1

Organisms images are based on (changes were made):

- – URL: http://commons.wikimedia.org/wiki/File:Biology_organism_collage.png
  – License: CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/deed.en

Tumor images are based on (changes were made):

- – URL: https://www.flickr.com/photos/26016306@N03/2477654635/in/photostream/
  – Author: M.L. Cohen
  – License: CC BY-NC-SA 2.0 https://creativecommons.org/licenses/by-nc-sa/2.0/
- – URL: http://de.wikipedia.org/wiki/Papill%C3%A4rer_glioneuronaler_Tumor#mediaviewer/File:Papillary_glioneuronal_tumor.jpg
  – Reference: Radotra BD, Kumar Y, Bhatia A, Mohindra S., Radotra BD, Kumar Y, Bhatia A, Mohindra S. (Papillary glioneuronal tumor: a new entity awaiting inclusion in WHO classification. Diagn Pathol. 2, 6. 2007).
  – License: CC BY 2.0 http://creativecommons.org/licenses/by/2.0/
- – URL: http://www.radpod.org/2007/07/07/dermoid-cyst-2/
  – Reference : Outwater EK, Siegelman ES, and Hunt JL. Ovarian Teratomas: Tumor Types and Imaging Characteristics. RadioGraphics 2001; 21: 475.
  – License: CC BY-NC 2.5 http://creativecommons.org/licenses/by-nc/2.5/

Images of MHC-I molecules were taken from (changes were made):

- – Source: Adams, Erin J., et al. "Structural elucidation of the m157 mouse cytomegalovirus ligand for Ly49 natural killer cell receptors." Proceedings of the National Academy of Sciences 104.24 (2007): 10128-10133.
  – URL: http://www.pnas.org/content/104/24/10128/F2.expansion.html
  – License: Granted permission by email on September 26th, 2014. Copyright (2007) National Academy of Sciences, U.S.A.

A.2 Sources for Figure 2.1

Images of Transcription:

- – URL: http://www.pnas.org/content/104/24/10128/F2.expansion.html
  – Reference: Adams, Erin J., et al. "Structural elucidation of the m157 mouse cytomegalovirus ligand for Ly49 natural killer cell receptors." Proceedings of the National Academy of Sciences 104.24 (2007): 10128-10133.
  – License: Anyone may, without requesting permission, use original figures or tables published in PNAS for noncommercial and educational use (i.e., in a review article, in a book that is not for sale) provided that the original source and the applicable copyright notice are cited. http://www.pnas.org/site/aboutpnas/rightperm.xhtml
- – URL: http://www.genome.gov/EdKit/bio2j.html
  – License: Public domain (http://www.genome.gov/copyright.cfm)

A.3 Sources for Figure 2.2

Images of shRNA data generation process were taken from:

- – Source: Fellmann, Christof, et al. "Functional identification of optimized RNAi triggers using a massively parallel sensor assay." Molecular cell 41.6 (2011): 733-746.
  – URL: http://www.cell.com/cms/attachment/616311/4975399/mmc1.pdf
  – License: Granted permission via RightsLink on October 14th, 2014. License number 3487810917398. Copyright (2011) Elsevier

A.4 Sources for Figure 4.1

- Kidney tumor:
  – URL: http://en.wikipedia.org/wiki/Lung_cancer#mediaviewer/File:LungCACXR.PNG
  – License: CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/
- Lung tumor:
  – URL: http://en.wikipedia.org/wiki/Renal_oncocytoma#mediaviewer/File:Onkozytom_der_Niere.jpg
  – License: CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/
- Brain tumor (see above).

A.5 Sources for Figure 4.16

- Amino Acids
  - URL: http://en.wikipedia.org/wiki/Protein_(nutrient)#mediaviewer/File:Amino_acids.png
  - License: CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/

B Derivation Details

B.1 Source-regularized SVM

$$\begin{aligned} 2 \cdot \mathfrak{R}(\mathbf{w} \,|\, \mathbf{w}_s) =& (1-B)\| \mathbf{w} \|^2 + B\| \mathbf{w} - \mathbf{w}_s \|^2 \\ =& (1-B)\, \mathbf{w}^T \mathbf{w} + B(\mathbf{w} - \mathbf{w}_s)^T(\mathbf{w} - \mathbf{w}_s) \\ =& (1-B)\, \mathbf{w}^T \mathbf{w} + B\, \mathbf{w}^T \mathbf{w} - 2B\, \mathbf{w}^T \mathbf{w}_s + B\, \mathbf{w}_s^T \mathbf{w}_s \\ =& \mathbf{w}^T \mathbf{w} - 2B\, \mathbf{w}^T \mathbf{w}_s + const \end{aligned}$$

C Domain-Adaptation Experimental Details

C.1 Synthetic Dataset

The synthetic data set was generated by applying subsequent mutations to the parameters of a generative model, subject to certain constraints. The source code for this procedure may be obtained from:
https://github.com/cwidmer/multitask/blob/master/data_processing.py#L1004. Experiments were performed using the experimental framework available from: https://github.com/cwidmer/multitask.

C.2 Splice-sites from 15 Organisms

Data was generated using mGene for the 15 organisms described in the main text. mGene may be obtained from https://github.com/ratschlab/mGene. For this,

we used current genomes in fasta format and genomic annotations in GFF format. Experiments were performed using the experimental framework available from: `https://github.com/cwidmer/multitask`.
The following implementations are of particular interest:
Top-Down (SVMLight):
`https://github.com/cwidmer/multitask/blob/master/method_hierarchy_svm_new.py`
Top-Down (LibLinear):
`https://github.com/cwidmer/multitask/blob/master/method_hierarchy_liblinear.py`
Individual:
`https://github.com/cwidmer/multitask/blob/master/method_plain_svm.py`
Union:
`https://github.com/cwidmer/multitask/blob/master/method_union_svm.py`

C.3 Gene Finding in Multiple Prokaryotic Genomes

Data The processed data set in MATLAB format is available here:
`https://drive.google.com/file/d/0B-nb9cCVAP6NNlU1WV9uWHp6VG8/edit?usp=sharing`
The raw data may be downloaded from `ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/`. We used the following organisms (including ids):
Escherichia.coli.BW2952.uid59391
Escherichia.fergusonii.ATCC.35469.uid59375
Enterobacter.638.uid58727,
Klebsiella.pneumoniae.MGH.78578.uid57619
Salmonella.enterica.serovar.Heidelberg.SL476.uid58973
Agrobacterium.tumefaciens.C58.uid57865
Rhizobium.leguminosarum.bv..viciae.3841.uid57955
Helicobacter.pylori.26695.uid57787
Mycobacterium.tuberculosis.H37Rv.uid57777
Bifidobacterium.longum.NCC2705.uid57939
Bacillus.anthracis.Ames.uid57909
Bacillus.subtilis.168.uid57675
Bacteroides.thetaiotaomicron.VPI.5482.uid62913
Flavobacterium.psychrophilum.JIP02.86.uid61627
Mycoplasma.genitalium.G37.uid57707
Mycoplasma.pneumoniae.M129.uid57709
Clostridium.botulinum.A.ATCC.19397.uid58927
Streptococcus.thermophilus.CNRZ1066.uid58221
Acidobacterium.capsulatum.ATCC.51196.uid59127
Fusobacterium.nucleatum.ATCC.25586.uid57885

Source Code Source code for data generation from the above raw data is available from: `https://github.com/cwidmer/so-da`
Solvers were implemented as part of mTIM, which is available from:
`https://github.com/nicococo/mTIM`.

D Multitask Learning Experimental Details

D.1 GRED: graph-regularized 3D shape reconstruction

Data We make available an example data set, consisting of a stack of TIF-files capturing both, the green and the red channel. The data set is available from: `https://drive.google.com/file/d/0B-nb9cCVAP6NOThaSFhpd21VcW8/edit?usp=sharing`.

Source Code The source code of the development version of GRED is available here:
`https://github.com/cwidmer/GRED`
Robustness experiments were performed using the following script:
`https://github.com/cwidmer/GRED/blob/master/artificial_data.py`
An extended version of the graph-regularized shape reconstruction algorithm is implemented using the python libraries cvxmod, a high-level modeling tool for optimization problems, cvxopt, a powerful python library for convex optimization problems, and an LP solver from GLPK. The corresponding python module is available here:
`https://github.com/cwidmer/GRED/blob/master/fit_ellipse_stack_conic.py`
Finally, the center piece of the graphical user interface is implemented in:
`https://github.com/cwidmer/GRED/blob/master/qt_gui.py`

D.2 Joint 2D/3D Model Learning

For these experiments, we have extended the tool *PLEASD: A Matlab Toolbox for Structured Learning* to handle multitask regularizers. Details of the optimization procedure involving partial annotations can be looked up in the corresponding source code, which is available at: `https://github.com/xlou/PLEASD`.

D.3 Execution Time Experiments

Data The used data sets are available from the following sources. All of the data sets below can be parsed using the following python module:
`https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/data.py`

Toy is generated by the following script (with fix seed): `https://github.com/`

cwidmer/mtl_dcd_ecml/blob/ecml/dcd_data.py.

Landmine can be downloaded from http://www.cs.columbia.edu/~jebara/code/multisparse/LandmineData.mat.

The original *MNIST* data set can be downloaded from http://yann.lecun.com/exdb/mnist/. In MATLAB format, it can be found here: http://www.cs.nyu.edu/~roweis/data/mnist_all.mat. To yield a multitask setting, the data is further processed into three problems: Discriminate "1" from "0", "7" from "9" and "2" from "8". For each task, the number of training examples is restricted to 3000.

Cancer is availabe from:
https://drive.google.com/file/d/0B-nb9cCVAP6NWUhaRmFBZEg5VFE/edit?usp=sharing. Here, the column "pcr" in the corresponding csv-file is used as label.

Splice is available from:
https://drive.google.com/file/d/0B-nb9cCVAP6NLWtHZ3RyVXZsNnc/edit?usp=sharing. The data format is the compressed "pickle" format. An example script to load this format is available from: https://gist.github.com/cwidmer/7813392. The script to generate the provided splice-site data from raw genome sequences and annotations is available here: https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/generate_splice_data.py

Source Code The timing experiments were performed using a custom branch of SHOGUN, as considerable modifications were required to track convergence during the underlying optimization procedure. This branch is available from:
http://bioweb.me/mtl-dcd
The python code to set up data and the experiments is available under the GPLv2:
https://github.com/cwidmer/mtl_dcd_ecml.
The convergence plots were generated using
https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/track_convergence.py,
and the learning curves using
https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/learning_curves.py.

We provide different implementation of our solver for illustration and debugging purposes. The one benchmarked was the implementation available in SHOGUN, however, we also provide python implementations for the following strategies:

- Dual-coordinate descent with and without shrinking:
  https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/solver_dcd_python.py

- Multitask kernel using QP solver from optimization package openopt: `https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/solver_openopt_mtk.py`
- Primal/dual solver using finite differences: `https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/solver_finite_differences.py`
- Wrapper for Shogun solvers (multitask kernel and dual-coordinate descent): `https://github.com/cwidmer/mtl_dcd_ecml/blob/ecml/solver_shogun.py`

D.4 MHC-I binding prediction

Data The IEDB benchmark data set from Peters et al., 2006 contains quantitative binding data (IC_{50} values) for various MHC alleles, including 35 human MHC alleles. Splits for a 5-fold cross-validation are given. We evaluate the performance of the proposed methods on a subset of this data set: binding data of nonameric peptides with respect to human MHC. Peptides with IC_{50} values greater than 500 nM were considered non-binders, all others binders. The data set can be downloaded from: `http://www.biomedcentral.com/1471-2105/10/394/additional`

Amino acid descriptors A wide range of physico-chemical and other descriptors of AAs have been published. Within this work we use encode each AA by 20 descriptors corresponding to the respective entries of the Blosum50 substitution matrix [Henikoff and Henikoff, 1992].

Performance evaluation procedure For performance evaluation, we employed a two times nested 5-fold cross-validation, i.e. two nested cross-validation loops. The inner loop is used for model selection (kernel and regularization parameters) and the outer loop for performance estimation. Performance is measured by averaging the area under the ROC curve (auROC).

Performance analysis of the improved WD kernel Performances of the WD and the WD-RBF kernel were analyzed on all 35 human MHC alleles contained in the IEDB benchmark.

Performance analysis of the multitask kernel approach Performances of three multitask learning approaches using a) the WD kernel, b) the WD-RBF kernel, and c) the WD-RBF kernel with an additional optimization step were also analyzed on all 35 human MHC alleles contained in the IEDB benchmark.

SVM computations We used the freely available large scale machine learning toolbox SHOGUN [Sonnenburg et al., 2010] for all SVM computations. All used kernels are implemented as part of the toolbox and are part of Shogun-0.9.3.

D.5 Learning the similarity

D.5.1 Toy Data

The toy experiment was performed using the following python script, documenting parameters and other experimental details:
`https://gist.github.com/cwidmer/8752502`
Note that the above will be part of the *pyMTL toolbox*, which we plan to release as part of a forthcoming Journal publication.

D.5.2 Power-set MT-MKL

This experiment was performed using the MHC-I data, described in Section D.4. Further, this experiment was performed using the experimental framework available from:
`https://github.com/cwidmer/multitask`.
The implementations of the Power-set MT-MKL method is available here:
`https://github.com/cwidmer/multitask/blob/master/mkl/method_multitask_kernel_mkl_mask_subset.py`
The above will be part of a modular python library (currently in preparation) to facilitate the use of the methods presented in this thesis.