Tapio Pahikkala

New Kernel Functions and Learning Methods for Text and Data Mining

Turku Centre *for* Computer Science

TUCS Dissertations
No 103, June 2008

New kernel functions and learning methods for text and data mining

Tapio Pahikkala

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Auditorium Beta on June 18th, 2008, at 12 noon.

University of Turku
Department of Information Technology
Joukahaisenkatu 3-5, 20014 Turku

2007

Supervisors

Professor Tapio Salakoski
Department of Information Technology
University of Turku
Finland

Docent Jouni Järvinen
Department of Information Technology
University of Turku
Finland

Doctor Jorma Boberg
Department of Information Technology
University of Turku
Finland

Reviewers

Professor Alessandro Moschitti
Department of Information and Communication Technology
University of Trento
Italy

Professor Juho Rousu
Department of Computer Science
University of Helsinki
Finland

Opponent

Professor Timo Honkela
Laboratory of Computer and Information Science
Helsinki University of Technology
Finland

ISBN 978-952-12-2091-3
ISSN 1239-1883

Abstract

Recent advances in machine learning methods enable increasingly the automatic construction of various types of computer assisted methods that have been difficult or laborious to program by human experts. The tasks for which this kind of tools are needed arise in many areas, here especially in the fields of bioinformatics and natural language processing. The machine learning methods may not work satisfactorily if they are not appropriately tailored to the task in question. However, their learning performance can often be improved by taking advantage of deeper insight of the application domain or the learning problem at hand. This thesis considers developing kernel-based learning algorithms incorporating this kind of prior knowledge of the task in question in an advantageous way. Moreover, computationally efficient algorithms for training the learning machines for specific tasks are presented.

In the context of kernel-based learning methods, the incorporation of prior knowledge is often done by designing appropriate kernel functions. Another well-known way is to develop cost functions that fit to the task under consideration. For disambiguation tasks in natural language, we develop kernel functions that take account of the positional information and the mutual similarities of words. It is shown that the use of this information significantly improves the disambiguation performance of the learning machine. Further, we design a new cost function that is better suitable for the task of information retrieval and for more general ranking problems than the cost functions designed for regression and classification. We also consider other applications of the kernel-based learning algorithms such as text categorization, and pattern recognition in differential display.

We develop computationally efficient algorithms for training the considered learning machines with the proposed kernel functions. We also design a fast cross-validation algorithm for regularized least-squares type of learning algorithm. Further, an efficient version of the regularized least-squares algorithm that can be used together with the new cost function for preference learning and ranking tasks is proposed. In summary, we demonstrate that the incorporation of prior knowledge is possible and beneficial, and novel advanced kernels and cost functions can be used in algorithms efficiently.

Acknowledgements

First of all I want to thank my supervisors Jorma, Jouni, and Tapio. I also thank my colleagues Aleksandr, Antti, Evgeni, Filip, Hanna, and Sampo for inspiring discussions and fruitful cooperation. This thesis would not have been possible without the good spirit and atmosphere in our group. I am grateful for Professor Olli Nevalainen and Docent Tero Aittokallio for encouraging me to begin my research career.

I would also like to thank Professor Alessandro Moschitti and Professor Juho Rousu for their excellent reviews for this thesis. Their critic on my thesis was extremely constructive and helpful. Further, I sincerely thank Professor Timo Honkela for accepting to act as opponent at the disputation of this thesis.

I am grateful for Turku Centre for Computer Science (TUCS) and Department of Information Technology for providing excellent working conditions and financial support. I acknowledge the staff at these institutions for creating pleasant and helpful working environment. I also thank the Nokia Foundation for two scholarships, and Tekes, the Finnish Funding Agency for Technology and Innovation, for financial support of my work.

Finally, I thank my wife Nina, my daughter Anna, and my parents for their love, encouragement, and support during the years I was working with my thesis.

List of original publications

I Tapio Pahikkala, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation. *BMC Bioinformatics*, 6(1):157, 2005.

II Tapio Pahikkala, Sampo Pyysalo, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Kernels incorporating word positional information in natural language disambiguation tasks. In Ingrid Russell and Zdravko Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 442–447, Menlo Park, Ca, 2005. AAAI Press.

III Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Matrix Representations, Linear Transformations, and Kernels for Disambiguation in Natural Language. TUCS Techical Report 890, 2008. Submitted to a journal.

IV Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Fast n-fold cross-validation for regularized least-squares. In Timo Honkela, Tapani Raiko, Jukka Kortela, and Harri Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, pages 83–90, Espoo, Finland, 2006. Otamedia.

V Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. Learning to rank with pairwise regularized least-squares. In Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

VI Heidi Vähämaa, Pekka Ojala, Tapio Pahikkala, Olli S. Nevalainen, Riitta Lahesmaa, and Tero Aittokallio. Computer-assisted identification of multi-trace electrophoretic patterns in differential display experiments. *Electrophoresis*, 28(6):879–893, 2007.

List of related co-authored and co-edited original publications not included in the thesis

Co-authored publications

- Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. Graph kernel for protein-protein interaction extraction. *BioNLP 2008 Workshop*. To appear.

- Tero Aittokallio, Jani S. Malminen, Tapio Pahikkala, Olli Polo, and Olli Nevalainen. Inspiratory flow shape clustering: An automated method to monitor upper airway performance during sleep. *Computer Methods and Programs in Biomedicine*, 85(1):8–18, 2007.

- Tero Aittokallio, Tapio Pahikkala, Pekka Ojala, Timo J. Nevalainen, and Olli Nevalainen. Electrophoretic signal comparison applied to mrna differential display analysis. *BioTechniques*, 34(1):116–122, 2003.

- Filip Ginter, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Extracting protein-protein interaction sentences by applying rough set data analysis. In Husaku Tsumoto, Roman Slowinski, Jan Komorowski, and Jerzy W. Grzymala-Busse, editors, *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing, Uppsala, Sweden*, volume 3066 of *Lecture Notes in Computer Science*, pages 780–785, Heidelberg, 2004. Springer.

- Filip Ginter, Tapio Pahikkala, Sampo Pyysalo, Evgeni Tsivtsivadze, Jorma Boberg, Jouni Järvinen, Aleksandr Mylläri, and Tapio Salakoski. Information extraction from biomedical text: The BioText project. In Margit Langemets and Penjam Priit, editors, *Proceedings of the Second Baltic Conference on Human Language Technologies (HLT 05)*, pages 131–136, 2005.

- Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. In Arie Hasman, Reinhold Haux, Johan van der Lei, Etienne De Clercq, and Francis Roger-France, editors, *Ubiquity: Technologies for Better Health in Aging Societies. Proceedings of MIE2006. The 20th International Conference of the European federation for Medical Informatics*, Studies in Health Technology and Informatics, pages 789–794, Amsterdam, 2006. IOS Press.

- Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. *International Journal of Medical Informatics*, 76(3): S362–S368, 2007.

- Tapio Pahikkala, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Incorporating external information in Bayesian classifiers via linear feature transformations. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Proceedings of the 5th International Conference on NLP (FinTAL 2006)*, volume 4139 of *Lecture Notes in Computer Science*, pages 399–410, Heidelberg, 2006. Springer.

- Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Improving the performance of Bayesian and support vector classifiers in word sense disambiguation using positional information. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 90–97, Espoo, Finland, 2005. Otamedia.

- Tapio Pahikkala, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Transductive ranking via pairwise regularized least-squares. In Paolo Frasconi, Kristian Kersting, and Koji Tsuda, editors, *Workshop on Mining and Learning with Graphs (MLG'07)*, pages 175–178, 2007.

- Tapio Pahikkala, Antti Airola, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Efficient AUC maximization with regularized least-squares. In *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*. To appear.

- Tapio Pahikkala, Evgeni Tsivtsivadze, Jorma Boberg, and Tapio Salakoski. Graph kernels versus graph representations: a case study in parse ranking. In Thomas Gärtner, Gemma C. Garriga, and Thorsten Meinl, editors, *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG'06)*, pages 181–188, 2006.

- Sampo Pyysalo, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Evaluation of two dependency parsers on biomedical corpus targeted at protein-protein interactions. *Recent Advances in Natural Language Processing for Biomedical Applications, special issue of the International Journal of Medical Informatics*, 75 (6):430–442, 2006.

- Sampo Pyysalo, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, Tapio Salakoski, and Jeppe Koivula. Analysis of link grammar on biomedical dependency corpus targeted at protein-protein interactions. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *Proceedings of the JNLPBA workshop at COLING'04*, pages 15–21, 2004.

- Hanna Suominen, Sampo Pyysalo, Marketta Hiissa, Filip Ginter, Shuhua Liu, Dorina Marghescu, Tapio Pahikkala, Barbro Back, Helena Karsten, and Tapio Salakoski. Performance evaluation measures for text mining. In Min Song and Yi-Fang Wu editors, *Handbook of Research on Text and Web Mining Technologies*, To appear.

- Hanna Suominen, Tapio Pahikkala, Marketta Hiissa, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Relevance ranking of intensive care nursing narratives. In Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, 10th International Conference, KES 2006, Part I*, Lecture Notes in Computer Science, pages 720–727, Heidelberg, 2006. Springer.

- Evgeni Tsivtsivadze, Tapio Pahikkala, Antti Airola, Jorma Boberg, and Tapio Salakoski. A sparse regularized least-squares preference learning algorithm. In *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*. To appear.

- Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality kernels for sequential data and their applications to parse ranking. *Applied Intelligence*. To appear.

- Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Kernel methods for text analysis. In *Advances of Computational Intelligence in Industrial Systems*. To appear.

- Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality-convolution kernel and its application to dependency parse ranking. In Moonis Ali and Richard Dapoigny, editors, *Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2006)*, volume 4031 of *Lecture Notes in Computer Science*, pages 610–618, Heidelberg, Germany, 2006. Springer.

- Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Regularized least-squares for parse ranking. In A. Fazel Famili, Joost N. Kok, José Manuel Peña,

Arno Siebes, and A. J. Feelders, editors, *Proceedings of the 6th International Symposium on Intelligent Data Analysis*, volume 3646 of *Lecture Notes in Computer Science*, pages 464–474, Heidelberg, Germany, September 2005. Springer.

Co-edited conference proceedings

- Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors. *Proceedings of the Fifth International Conference on Natural Language Processing FinTAL 06, Turku, Finland*, volume 4139 of *Lecture Notes in Artificial Intelligence*, 2006. Springer, Heidelberg.

Contents

Chapter 1

Introduction

The capability of a machine learning method to take advantage of prior knowledge is often crucial for gaining satisfactory performance. By prior knowledge, we mean a deeper insight of the application domain or the learning problem at hand, for example. Moreover, the computational efficiency of a machine learning method has a strong influence on its usability.

With a learning machine, we refer to a system that may be automatically trained with a given data set to perform a specific task. This is in contrast, for example, to such rule-based artificial intelligence methods that are completely designed and programmed by human experts. Machine learning algorithms can play an essential role in applications that are too difficult or laborious to program by hand (see e.g. [40]). Many such application domains are found, for example, in the fields of data mining, bioinformatics, and natural language processing (NLP). In bioinformatics, there are huge databases from which the machine learning methods can be used to mine biological knowledge. Moreover, machine learning can be used to construct automatic tools that facilitate effective analysis of biological data. In NLP, machine learning can be used to improve the performance of automatic text processing tools and to infer knowledge from text.

Recently, the fields of BioNLP and medical NLP have emerged, because of the huge increase of the biological and medical texts. Usually, the tasks related to this type of texts are more challenging than those related to conventional text found, for example, from newspapers. Therefore, a lot of attention has been paid for tailoring machine learning methods, for example, to infer knowledge from scientific biological texts and clinical narratives.

This thesis considers ways to adapt machine learning algorithms to certain application domains and tasks. The adaptation is done by taking advantage of prior knowledge of the domain and task in question. By prior knowledge, we mean, for example, information about what characteristic of the data may be useful in solving the learning task. Further, prior knowl-

edge about the task may aid us in setting the actual objective of the learning algorithm. For example, it is possible to learn to rank data points using a regression algorithm, but the ranking performance may be improved by using a learning algorithm better suitable to the ranking task. In a sense, the use of prior knowledge together with the machine learning results in "hybrid" algorithms that are partly human designed and partly automatically learned. We focus on two typical ways to do the incorporation in the context of kernel-based methods, namely on designing so-called kernel and cost functions. The kernel functions can be considered as similarity measures between data points. Further, a kernel function acts as an interface between the actual learning algorithm and the data whose representation can depend heavily on the application domain. We can easily use our knowledge of the application domain to design the kernel function so that it helps to solve the learning problem in question. The cost functions are used to determine the actual learning objective. In case of a classification task or example, each misclassification causes a certain cost and the objective of the learning method is to minimize the number of misclassifications.

We also develop computationally efficient algorithms for training the considered learning machines. The training algorithms are designed in such a way that they work efficiently together with the kernel functions proposed in this thesis. Moreover, algorithms for specific learning problems such as ranking are designed. By ranking, we refer to a task in which a set of data points are supposed to be ranked, for example, in order of importance. The proposed ranking method consists of a cost function appropriate for the ranking task and a computationally efficient training algorithm. Further, we consider cross-validation, a widely used method for selecting parameter values for the learning algorithms and performance measurement. We develop fast algorithms for cross-validation and parameter selection for the learning methods considered in this work.

1.1 Aims of the Thesis

The general aims of this thesis are to design machine learning algorithms that are computationally efficient and to incorporate prior knowledge of the learning tasks in question in an advantageous way. We divide this objective into the following two parts.

Incorporating Prior Knowledge We develop kernel functions especially for the natural language disambiguation problems. The proposed kernels take account of the positional information and the mutual similarities of the context words in an advantageous way. We evaluate our kernels with various disambiguation tasks. The tasks are gene versus protein name dis-

ambiguation, word sense disambiguation, and context sensitive spelling error correction. As kernel-based learning algorithms, we use support vector machines (see e.g. [72]), regularized least-squares (RLS) (see e.g. [51]), and flexible Bayes classifiers [31]. For pattern recognition in differential display data, we consider the use of polynomial kernel functions of different degrees. The kernels are used together with the regularized least-squares algorithm.

We also use of our insight of the ranking tasks to develop learning algorithms. The problem of ranking is often cast as a problem of learning pairwise preferences so that the training set is composed of data point pairs, in which one point is preferred over the other. In this case, the aim is to predict the direction of preference for each data point pair. Therefore, we develop a cost function that approximates the number of pairwise misrankings and set the objective of the learning algorithm to be the minimization of this cost. The cost function can be considered as a variation of the cost function used in RLS learners.

Developing Efficient Algorithms Most of the considered kernel functions can be calculated via linear transformations of the data vectors. Therefore, we use standard linear algebra techniques also to train kernel-based learning machines efficiently. Further, we develop a highly efficient prediction algorithm for a learning method trained using the kernel functions.

Moreover, we consider methods based on linear algebra for training RLS learners. In particular, we develop a computationally efficient algorithm for performing cross-validation with RLS learners. This type of algorithms are also designed for variations of RLS, such as the ranking version of the RLS proposed by us.

A drawback of our ranking cost function is that the number of data point pairs grows quadratically with respect to the number of individual data points in the training set, and hence increasing the computational complexity of the traditional algorithms that are used to minimize this type of costs. Therefore, we develop a new algorithm for minimizing the cost whose computational complexity depends on the number of individual data points instead of the number of data point pairs, hence making the use of the ranking method practical.

1.2 Organization of the Thesis

The thesis is organized as follows. Chapter 2 presents a formal introduction to the methods used in this thesis. The introduction is based on machine learning literature and the research papers included in this thesis. The learning tasks are described in Chapter 3. A summary of each paper included in this thesis is presented in Chapter 4, and Chapter 5 concludes the work.

Chapter 2

Regularized kernel methods

In this chapter, we briefly describe machine learning methods relevant for the thesis. We first recall the concept of supervised learning. A *supervised learner* is a machine that is taught with a set of training examples to perform a specific task. By a *task*, we mean the prediction of an output for an unseen data point. Formally, let $X = (x_1, \ldots, x_m) \in (\mathcal{X}^m)^{\mathrm{T}}$ be a sequence of *inputs*, $Y = (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathcal{Y}^m$ a sequence of *outputs*, where $\mathcal{X}$ and $\mathcal{Y}$, called the *input space* and the *output space*, are the sets of possible inputs and outputs, respectively. In this thesis, $\mathcal{Z}^m$ and $(\mathcal{Z}^m)^{\mathrm{T}}$ denote the sets of column and row vectors of size m whose elements belong to the set $\mathcal{Z}$, respectively. Moreover, let $S = ((x_1, y_1), \ldots, (x_m, y_m))^{\mathrm{T}} \in (\mathcal{X} \times \mathcal{Y})^m$ be a *training set* of m training *examples*, where $(\mathcal{X} \times \mathcal{Y})^m$ denotes the set of all possible training sets of size m. Note that while we call S a training set, it is actually an ordered sequence of examples. Further, we use the term *data point* when we refer to input-output pairs $z = (x, y)$ that do not necessarily belong to the training set. Unless stated otherwise, we restrict our considerations to the case in which $\mathcal{Y} \subseteq \mathbb{R}$, and hence the sequence of outputs can be treated as a real column vector $Y \in \mathbb{R}^m$. Training of a supervised learner can be considered as a process of selecting a function among a set of candidates that best performs the task in question. Following the notation of [24], we formalize the training algorithm in the following way. An algorithm $\mathcal{A}$ that selects the function given the training set S can be considered as a mapping

$$\mathcal{A} : \bigcup_{m \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^m \to \mathcal{H}, \; S \mapsto f, \tag{2.1}$$

where $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, called the *hypothesis space*, is a set of functions among which the algorithm selects an appropriate hypothesis $f \in \mathcal{H}$. With $\mathcal{Y}^{\mathcal{X}}$ and $\bigcup_{m \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^m$, we denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, and the set of all possible training sets, respectively.

Typical difficulties of supervised learning are *underfitting* and *overfitting*. By underfitting, we mean that the learning algorithm selects a function that

is not able to correctly predict the outputs of the training examples. This usually happens when the hypothesis space $\mathcal{H}$ does not contain the correctly predicting functions or when the training algorithm does not prefer to select them. For example, if the functional relationship between the inputs and outputs is nonlinear, a learning machine may underfit if the hypothesis space consists of linear functions only. Overfitting, on the other hand, refers to the situation in which the selected function can correctly predict the outputs of the training examples, but fails to do so with unseen data points. This may happen when too complex functions are selected. For example, if we memorize the whole training set into a lookup table and randomly predict the outputs for unseen data points, we have fitted perfectly to the training data but we have not learned to predict correctly.

In this thesis, we use so-called kernel functions to modify the hypothesis space to better fit to the learning task in question and therefore make the learning machines less likely to underfit. We also use so-called regularization techniques and this causes the learning algorithm to prefer simple functions over complex ones in order to avoid the problem of overfitting.

We next discuss about kernel-based learning algorithms. Designing kernel functions suitable for specific learning problems is a practical way to incorporate prior knowledge of the problem into the learning machine. Moreover, with the help of kernel functions, we are usually able to make the hypothesis space $\mathcal{H}$ expressive enough so that the underfitting does not happen. The obtained space $\mathcal{H}$ provides us the tools to measure the complexity of its elements so that the problem of overfitting can be avoided. The use of such tools is often called regularization, and hence we refer to the resulting algorithms as regularized kernel methods.

In order to determine how well a learner is able to predict the outputs of unseen data points, we measure its performance on a test set of data points that the learner has not used in the training phase. How the measurement is done, depends on the learning task in question. In Section 2.4, we give a brief presentation on the performance measures that are relevant in the thesis.

2.1 Kernel Functions

Kernel-based learning algorithms (see e.g. [24, 64, 65]) consist of a learning algorithm and the kernel function. The kernel can be considered as a similarity measure between two inputs which corresponds to their inner product in some feature space into which the original inputs are mapped. This is very useful, for instance, when the concept to be learned depends nonlinearly on the data, but the learning algorithm is able to learn only linear dependencies. For example, we may have a two-class classification problem

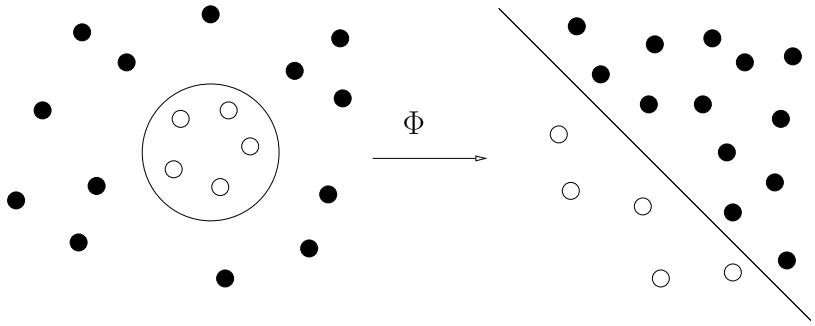


Figure 2.1: If inputs of one class all lie inside a circle in two-dimensional space and inputs of another class are all outside of the same circle, a linear learning machine cannot classify them correctly. We can fix this situation by using a feature mapping Φ that maps the inputs into a new feature space where they are linearly separable.

in which the class boundary is a circle so that the inputs of data points belonging to the positive class lie inside and the inputs of the negative data points outside the circle. This kind of situation is depicted in Figure 2.1 — the mapping Φ transforms the boundary to a form a learner can "comprehend". In order to design a good kernel function for a particular learning problem, we may use prior knowledge of the problem (see e.g. [63] for a typical example of this approach).

Formally, let $\mathcal{X}$ denote the input space, which can be any set, and $\mathcal{F}$ denote the *feature vector space*. For any mapping

$$\Phi : \mathcal{X} \to \mathcal{F}, \tag{2.2}$$

the inner product

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle \tag{2.3}$$

of the mapped inputs is called a *kernel function*.

Note that if we have an efficient way to compute the kernel directly, there is no need to explicitly compute the mapping Φ, because the kernel-based learning algorithms need only the value $k(x, z)$. In fact, we do not even have to care about the mapping Φ or the feature space $\mathcal{F}$. However, in that case we still need to ensure that the mapping exists so that the function $k(x, z)$ really corresponds to an inner product in some feature space. A necessary condition for this is that $k(x, z)$ is symmetric and finitely positive semidefinite (see e.g. [65]). A kernel $k(x, z)$ is said to be *finitely positive semidefinite* if

$$\sum_{i,j=1}^{m} a_i a_j k(x_i, x_j) \geq 0 \tag{2.4}$$

for any $m \in \mathbb{N}$, $x_1, \ldots, x_m \in \mathcal{X}$, and $a_1, \ldots, a_m \in \mathbb{R}$. On the other hand, *symmetry* of a kernel function means that $k(x, z) = k(z, x)$ for all $x, z \in \mathcal{X}$.

Given a sequence $X \in \mathcal{X}^m$ of inputs, the values of the kernel function for each pair of inputs are usually stored in so-called *kernel matrix*. Before presenting the kernel matrix, we define some useful shorthand notations. Firstly, let $\Phi(X)$ be row vector whose elements are the images of the inputs of the training examples, that is,

$$\Phi(X) = (\Phi(x_1), \ldots, \Phi(x_m)) \in (\mathcal{F}^m)^{\mathrm{T}}. \tag{2.5}$$

Secondly, we define

$$k(x, X) = (k(x, x_1), \ldots, k(x, x_m)) \in (\mathbb{R}^m)^{\mathrm{T}} \tag{2.6}$$

and

$$k(X, x) = \begin{pmatrix} k(x_1, x) \\ \vdots \\ k(x_m, x) \end{pmatrix} \in \mathbb{R}^m$$

for all $x \in \mathcal{X}, X \in (\mathcal{X}^m)^{\mathrm{T}}$. Finally, similarly to the above notations, we define the kernel matrix $K \in \mathbb{R}^{m \times m}$, where $\mathbb{R}^{m \times m}$ denotes the set of all $m \times m$ real matrices, as

$$K = k(X, X) = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{pmatrix}. \tag{2.7}$$

For example, if the codomain of the mapping $\Phi : \mathcal{X} \to \mathcal{F}$ is $\mathcal{F} = \mathbb{R}^h$, we can consider $\Phi(X)$ as a real matrix of type $m \times h$ and the kernel matrix is the result of a matrix product of $\Phi(X)^{\mathrm{T}}$ and $\Phi(X)$. We can also express the finitely positive semidefiniteness condition (2.4) equivalently by saying that K is always a positive semidefinite matrix. Formally, this means that if the kernel function is symmetric and a finitely positive semidefinite, then for any $m \in \mathbb{N}$, $X \in (\mathcal{X}^m)^{\mathrm{T}}$, and $A \in \mathbb{R}^m$,

$$A^{\mathrm{T}} K A \geq 0,$$

where K is the kernel matrix $k(X, X)$. This property is very useful for the learning algorithms considered in Section 2.2.

There are various types of kernels that can be found in the literature. Next, we consider so-called *polynomial kernels* (see e.g. [50, 64]). Let $[x]_j$ denote the jth component of an input space vector x. Suppose that the input space is two-dimensional and the learning problem to be solved is not linear, that is, the classes in a binary classification task are not linearly separable. In Figure 2.1, for example, the inputs of one class lie in a circle in

two-dimensional space and inputs of another class are all outside of the same circle. The circle separating the two classes can not be expressed linearly with the two input features. However, the linear separation is possible using the second order product features $[x]_1^2$, $[x]_1[x]_2$, and $[x]_2^2$. Therefore, we transform the input features into their second order product features with a feature map

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3,\ ([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, [x]_1[x]_2). \tag{2.8}$$

The values of Φ are easy to compute and so is the inner product $\langle \Phi(x), \Phi(z) \rangle$. However, the computations are very demanding in higher dimensional input spaces because the number of the product features grows exponentially. In fact, when the number of input features is n, there are altogether

$$\binom{d+n-1}{n} = \frac{(d+n-1)!}{n!(d-1)!}$$

product features of order d.

If we, instead of (2.8), consider the following mapping of the inputs

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3,\ ([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, \sqrt{2}[x]_1[x]_2),$$

we can compute the inner product between the mapped inputs with the following polynomial kernel function of degree 2:

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle = [x]_1^2[z]_1^2 + [x]_2^2[z]_2^2 + 2[x]_1[x]_2[z]_1[z]_2 = \langle x, z \rangle^2,$$

which is simply the square of the inner product taken in the input space. Of course, the same function can be determined for arbitrary n and $d \in \mathbb{N}$, where $d \in \mathbb{N}$ is the degree of the polynomial. Using this kind of shortcut, the expensive computations of values of Φ and their inner products can be avoided. In a general case, the homogenous polynomial kernels are of the form

$$k(x, z) = \langle x, z \rangle^d. \tag{2.9}$$

Sometimes, also so-called inhomogenous polynomial kernels are used, that is,

$$\begin{aligned} k(x, z) &= (\gamma + \langle x, z \rangle)^d \\ &= \sum_{i=0}^{d} \binom{d}{i} \gamma^{d-i} \langle x, z \rangle^i, \end{aligned}$$

where $\gamma \in \mathbb{R}^+$ is a parameter that determines the weights of the homogenous polynomial kernels of different degrees. As an example can be mentioned that we used a homogenous polynomial kernel of degree 8 in paper [VI] when identifying electrophoretic patterns, whereas inhomogenous polynomial kernels were used in paper [I].

2.1.1 Closure Properties of Kernel Functions

In order to construct kernel functions from existing ones, or by using some other functions as building blocks, the following closure properties of kernel functions are essential; they can be found in [12], for example. Let k_1 and k_2 be kernel functions and $K_1, K_2 \in \mathbb{R}^{m\times m}$ be the corresponding kernel matrices. Further, assume $\alpha \in \mathbb{R}^+$ and $A \in \mathbb{R}^m$. Then we may give the following observations.

- Sum of $k_1(x,z)$ and $k_2(x,z)$:

$$k(x,z) = k_1(x,z) + k_2(x,z) \tag{2.10}$$

  is a kernel, because $A^{\mathrm{T}}(K_1 + K_2)A = A^{\mathrm{T}}K_1A + A^{\mathrm{T}}K_2A \geq 0$.

- Multiplication of $k_1(x,z)$ by a positive constant α:

$$k(x,z) = \alpha k_1(x,z) \tag{2.11}$$

  is a kernel, because $A^{\mathrm{T}}\alpha K_1A = \alpha A^{\mathrm{T}}K_1A \geq 0$.

- The product of $k_1(x,z)$ and $k_2(x,z)$:

$$k(x,z) = k_1(x,z)k_2(x,z) \tag{2.12}$$

  is a kernel (see [12] for a proof).

- If $g(\cdot)$ is a real valued function on $\mathcal{X}$, then

$$k(x,z) = g(x)g(z) \tag{2.13}$$

  is a kernel, because

$$\begin{aligned}\sum_{i,j} a_i a_j k(x_i,x_j) &= \sum_{i,j} a_i a_j g(x_i)g(x_j)\\ &= \sum_i a_i g(x_i) \sum_j a_j g(x_j)\\ &= \left(\sum_i a_i g(x_i)\right)^2 \geq 0\end{aligned}$$

  for any set of inputs $x_1, \ldots, x_m$ and any $a_1, \ldots, a_m \in \mathbb{R}$.

- Any positive real constant

$$k(x,z) = \alpha \tag{2.14}$$

  is a kernel. Note that this follows directly from (2.13) by choosing $g(x) = \sqrt{\alpha}$ for all $x \in \mathcal{X}$.

- Any polynomial p with positive coefficients evaluated at k_1:

$$k(x, z) = p(k_1(x, z)) \tag{2.15}$$

  is a kernel.

- The exponential function

$$k(x, z) = e^{k_1(x,z)} \tag{2.16}$$

  is a kernel, because we can approximate it arbitrarily closely using polynomials with positive coefficients.

- If Ψ is a mapping from $\mathcal{X}$ to $\mathbb{R}^n$, where $n \in \mathbb{N}$, and k_3 is a kernel on $\mathbb{R}^n \times \mathbb{R}^n$, then

$$k(x, z) = k_3(\Psi(x), \Psi(z)) \tag{2.17}$$

  is a kernel. This follows from the fact that because each $\Psi(x)$ is already a vector in some feature space, the matrix obtained by using a valid kernel k_3 restricted to those vectors is positive semidefinite as required.

- If $x, z \in \mathbb{R}^n$ and $\widehat{M} \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix, then

$$k(x, z) = x^{\mathrm{T}} \widehat{M} z \tag{2.18}$$

  is a kernel. This can be seen by considering the eigen decomposition $\widehat{M} = V \Lambda V^{\mathrm{T}}$, where V is an orthogonal matrix that contains the eigenvectors of K, and Λ is a diagonal matrix containing the corresponding eigenvalues. Due to the positive semidefiniteness of $\widehat{M}$, the eigenvalues are nonnegative. Therefore, by setting $\Phi(x) = Mx$, where $M = \sqrt{\Lambda} V^{\mathrm{T}}$ and $\sqrt{\Lambda}$ is a diagonal matrix that contains the square roots of the eigenvalues, we can rewrite (2.18) as

$$k(x, z) = x^{\mathrm{T}} \widehat{M} z = x^{\mathrm{T}} V \Lambda V^{\mathrm{T}} z = x^{\mathrm{T}} V \sqrt{\Lambda} \sqrt{\Lambda} V^{\mathrm{T}} z = \langle \Phi(x), \Phi(z) \rangle.$$

  which is of the form (2.3). Note that in this thesis the mappings of the form $x \mapsto Mx$ are generally referred to as linear transformations, or simply as transformations.

2.1.2 Constructing Learners with Linear Kernels

We now consider the special case of the kernel functions being linear, that is, kernels constructed using the closure property (2.18). Suppose that our input space and feature space are $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{F} = \mathbb{R}^h$, respectively. Recall that, in this case, the feature mapping can be expressed as a linear transformation of the inputs

$$\Phi(x) = Mx,$$

where $M \in \mathbb{R}^{h \times n}$ is called the matrix of the transformation. Then, the kernel (2.3) can be written as

$$k(x, z) = \langle Mx, Mz \rangle = (Mx)^{\mathrm{T}} Mz = x^{\mathrm{T}} M^{\mathrm{T}} Mz. \tag{2.19}$$

We can also perform the kernel calculation so that we transform the input x with a matrix

$$\widehat{M} = M^{\mathrm{T}} M.$$

Since M is a real-valued matrix the identity matrix $I \in \mathbb{R}^{h \times h}$ is positive definite, we can write for any $x \in \mathbb{R}^n$

$$x^{\mathrm{T}} M^{\mathrm{T}} I M x = b^{\mathrm{T}} I b \geq 0,$$

where $b = Mx$. Therefore, the matrix $\widehat{M}$ is a symmetric positive semidefinite and it can be used to determine a kernel function for the inputs. Thus, we obtain the value of (2.19) also by computing the inner product of the transformed inputs $\widehat{M}x$ and z

$$k(x, z) = \left\langle \widehat{M}x, z \right\rangle.$$

In Section 2.2, we consider functions that are linear combinations of the inputs x_i of the training examples in the feature space

$$f(\cdot) = \sum_{i=1}^{m} a_i k(\cdot, x_i),$$

where $a_i \in \mathbb{R}$. When the kernel is linear, this type of functions can be expressed as

$$\begin{aligned} f(\cdot) &= \sum_{i=1}^{m} a_i k(\cdot, x_i) \\ &= \sum_{i=1}^{m} a_i \left\langle \cdot, \widehat{M} x_i \right\rangle \\ &= \left\langle \cdot, \widehat{M} \left(\sum_{i=1}^{m} a_i x_i \right) \right\rangle \\ &= \langle \cdot, w \rangle. \end{aligned}$$

If we have w stored in the memory, the evaluation of the function at input $x \in \mathcal{X}$ can be calculated in a linear time with respect to the number of nonzero elements in x. This is, because we do not have to calculate the kernel functions separately for each training example but only the inner product between x and w. This is a clear computational advantage when

compared to the most commonly used nonlinear kernels with which this kind of speedup is not possible. Moreover, the computational benefit is even larger if the input vectors are sparse, that is, if most of the elements of the input vectors are zeros. We further note that there are also nonlinear kernels that take advantage of the sparsity of the feature vectors (see e.g. [61]).

We study kernel functions constructed from linear feature transformations especially in paper [III] but the kernels considered in papers [45] and [II] can also be linearized in a similar way as in the above examples. In addition, the contextual weighting considered in [I] can also be realized with a linear transformation, but in that study, we use nonlinear kernels together with the weighting. Further, many kernel functions taking advantage of external source of information (such as the one proposed in [6]) can be considered as linear feature transformation based kernels.

2.2 Regularization Framework

We now consider an approach that we call here the *regularization framework*. Both the support vector machines and the regularized least-squares learners can be considered as instances of this framework. As a similarity measure of the input space elements, we use a symmetric and positive definite kernel function k. In Section 2.2.1, we consider the hypothesis space determined by the kernel k and the input space $\mathcal{X}$. In Section 2.2.2, we aim to select such a hypothesis f from $\mathcal{H}$ that performs well on the training data set but is not overly complex. We use definitions similar to that used in [62].

2.2.1 Reproducing Kernel Hilbert Space

As discussed above, a learning algorithm $\mathcal{A}$ defined as in (2.1) selects a function f from a hypothesis space $\mathcal{H}$ that fits the training data well but is not too complex. Given a finitely positive semidefinite kernel function k, we can construct a hypothesis space of functions which is called a reproducing kernel Hilbert space (RKHS) [4]. This provides us the machinery

- to measure how well each candidate hypothesis fit to the training data, and
- to measure the complexity of the hypotheses.

Therefore, we are able to construct learning algorithms that make a trade-off between training set performance and hypothesis complexity. These kinds of algorithms are discussed in Section 2.2.

We follow [24, 52, 62, 64], with an exception that, for simplicity, we restrict our consideration to the real-valued functions. Recall the following characteristic properties of any Hilbert space $\mathcal{H}$:

(i) $\mathcal{H}$ is a vector space (over $\mathbb{R}$ in our case),

(ii) $\mathcal{H}$ is endowed with an inner product,

(iii) $\mathcal{H}$ is complete, that is, every Cauchy sequence in $\mathcal{H}$ is a convergent sequence.

Next we consider an interpretation of the mapping Φ that allows us to construct the RKHS corresponding to the kernel k and the input space $\mathcal{X}$.

Let $\mathbb{R}^{\mathcal{X}}$ denote the space of all functions that map from the input space $\mathcal{X}$ to real numbers. Then, the feature mapping can be defined as

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}},\ z \mapsto k(\cdot, z), \tag{2.20}$$

where $k(\cdot, z)\colon \mathcal{X} \to \mathbb{R}$ denotes the function assigning the real number $k(x, z)$ to $x \in \mathcal{X}$, that is, $\Phi(z)(\cdot) = k(\cdot, z)$. Let $\mathcal{H}_0$ be the span of the image of the mapping Φ, that is, the set of all finite linear combinations of the elements of $\{k(\cdot, z) \mid z \in \mathcal{X}\} \subset \mathbb{R}^{\mathcal{X}}$. Formally,

$$\mathcal{H}_0 = \left\{ \sum_{i=1}^{n} \beta_i k(\cdot, z_i) \,\middle|\, \beta_i \in \mathbb{R}, z_i \in \mathcal{X}, n \in \mathbb{N} \right\}.$$

Clearly, $\mathcal{H}_0$ is a vector space over $\mathbb{R}$.

Next, we endow $\mathcal{H}_0$ with an inner product. Let $f, g \in \mathcal{H}_0$ be defined by

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i), \text{ and } \qquad g(\cdot) = \sum_{j=1}^{n'} \beta_j k(\cdot, z_j),$$

where $n, n' \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$, and $x_i, z_j \in \mathcal{X}$. We define the inner product for the elements of $\mathcal{H}_0$ to be the following

$$\begin{aligned} \langle f, g \rangle_k &= \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, z_j) \\ &= \sum_{i=1}^{n} \alpha_i g(x_i) \\ &= \sum_{j=1}^{n'} \beta_j f(z_j). \end{aligned} \tag{2.21}$$

This really is an inner product, because it can be shown (see e.g. [52]) that it satisfies the following properties:

(i) Symmetry: $\langle f, g \rangle_k = \langle g, f \rangle_k$

(ii) Bilinearity: $\langle af + bg, h \rangle_k = a\langle f, h \rangle_k + b\langle g, h \rangle_k$

(iii) Strict positive definiteness:

$$\langle f, f\rangle_k \quad \geq \quad 0, \text{ and } \langle f, f\rangle_k = 0 \text{ implies that } f = 0$$

The term reproducing kernel comes from so-called reproducing kernel property that follows directly from the definition (2.21). Namely,

$$\langle k(\cdot, x), g\rangle_k = g(x), \tag{2.22}$$

which can be seen by choosing $f(\cdot) = k(\cdot, x)$ in (2.21), and in particular

$$\langle k(\cdot, x), k(\cdot, z)\rangle_k = k(x, z).$$

We can also express this by saying that k is the representer of evaluation, that is, a function f is evaluated at the input x by taking the inner product $\langle k(\cdot, x), f\rangle_k$.

By endowing the vector space $\mathcal{H}_0$ with the inner product (2.21), we have obtained a pre-Hilbert space. We also note that the norm induced by the inner product (2.21) is

$$\|f\|_k = \sqrt{\langle f, f\rangle}. \tag{2.23}$$

Following [62], we define the RKHS $\mathcal{H}$ corresponding to the kernel k to be the completion of $\mathcal{H}_0$, that is, in addition of all the finite linear combinations of the elements of $\{k(\cdot, x) \mid x \in \mathcal{X}\} \subset \mathbb{R}^{\mathcal{X}}$, $\mathcal{H}$ also contains all limits of Cauchy sequences that are convergent in the norm:

$$\mathcal{H} = \left\{ f(\cdot) = \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i), \beta_i \in \mathbb{R}, z_i \in \mathcal{X}, \|f\|_k < \infty \right\}. \tag{2.24}$$

The inner product in $\mathcal{H}$ is defined analogously to that in $\mathcal{H}_0$.

2.2.2 The Framework and the Representer Theorem

Let $\mathcal{H}$ be the hypothesis space defined as in (2.24). A suitable measure of the hypothesis complexity can be obtained from the function norm in the RKHS defined by the kernel k. Therefore, we define a function

$$\Omega : \mathbb{R} \to \mathbb{R}_+ \tag{2.25}$$

that we call a *regularizer*. The only requirement for (2.25) is that it is strictly monotonically increasing. The regularizer (2.25) assigns a value

$$\Omega(\|f\|_k) \tag{2.26}$$

for the function f that we use as its complexity measure.

The performance of the hypothesis on the training data set can be measured with various ways which depend on the learning task in question. We start by first defining a general cost function and consider more specific ones in the sequel. Let us denote $f(X) = (f(x_1), \ldots, f(x_m))^{\mathrm{T}}$ for sequence of inputs $X = (x_1, \ldots, x_m)$ and a hypothesis $f \in \mathcal{H}$. We use cost functions of type

$$c : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R} \tag{2.27}$$

to assign a value

$$c(Y, f(X)) \tag{2.28}$$

on the given training set S and a candidate hypotheses $f \in \mathcal{H}$ that measures how well f fits S. Usually, the cost functions have zero as their minimum value. Further, the larger is the value of the cost function, the worse is the fit considered to be. Similar kind of functions are also used to measure the performance of a hypothesis on a given test data set (Section 2.4). The cost functions are often approximations of the performance measures, because searching $\mathcal{H}$ for an optimally performing hypothesis may otherwise be an intractable task. In the sequel, we present some examples of cost functions.

We now consider the following variational problem as a realization of (2.1) that we use to select an appropriate hypothesis f among $\mathcal{H}$ for a given training set S. The problem is formally defined as

$$\mathcal{A}(S) = \operatorname*{argmin}_{f \in \mathcal{H}} (c(Y, f(X)) + \Omega(\|f\|_k)). \tag{2.29}$$

The first term measures the performance of a candidate hypothesis on the training set and the second term, the regularizer, measures the complexity of the hypothesis with the RHKS norm.

Following [62], we present the generalized representer theorem. Any solution $f \in \mathcal{H}$ of the minimization (2.29) admits the representation of the following form:

$$f(\cdot) = \sum_{i=1}^{m} a_i k(\cdot, x_i), \tag{2.30}$$

where $a_i \in \mathbb{R}$ and k is the kernel function associated with the RKHS mentioned above. This can be shown by contradiction. Suppose that the optimal hypothesis f can not be expressed with the images of the inputs and is therefore not of the form (2.30). Because $\mathcal{H}$ is a vector space, f can be written in the following two mutually orthogonal parts

$$f(\cdot) = \sum_{i=1}^{m} a_i k(\cdot, x_i) + v(\cdot), \tag{2.31}$$

where $v(\cdot) \in \mathcal{H}$ is a function with

$$\langle k(\cdot, x_i), v(\cdot) \rangle_k = 0$$

for all $1 \leq i \leq m$. Recall that since of the reproducing property (2.22) holds, we can write $v(z) = \langle k(\cdot, z), v(\cdot)\rangle_k$ for $z \in \mathcal{X}$. Therefore, we observe that, for all inputs $x_j, 1 \leq j \leq m$,

$$\begin{aligned} f(x_j) &= \langle k(\cdot, x_j), \sum_{i=1}^{m} a_i k(\cdot, x_i) + v(\cdot)\rangle_k \\ &= \sum_{i=1}^{m} a_i k(x_j, x_i) + \langle k(\cdot, x_j), v(\cdot)\rangle_k \\ &= \sum_{i=1}^{m} a_i k(x_j, x_i), \end{aligned}$$

that is, the value of (2.28) is independent of v.

Moreover,

$$\begin{aligned} \Omega(\|f_v(\cdot)\|_k) &= \Omega\left(\sqrt{\left\|\sum_{i=1}^{m} a_i k(\cdot, x_i)\right\|_k^2 + \|v(\cdot)\|_k^2}\right) \\ &\geq \Omega\left(\left\|\sum_{i=1}^{m} a_i k(\cdot, x_i)\right\|_k\right), \end{aligned}$$

that is, the smallest value of the regularizer (2.26) is obtained when $\|v(\cdot)\|_k = 0$. Therefore, when we substitute (2.31) in (2.29), we observe that (2.31) can be the solution only if $v = 0$, which proves the representer theorem.

Due to the representer theorem, we can restrict our set of candidate hypotheses to be the set of linear combinations of the training examples mapped with (2.20):

$$\mathcal{H}_X = \left\{ \sum_{i=1}^{m} a_i k(\cdot, x_i) \mid a_i \in \mathbb{R} \right\} \subseteq \mathcal{H}.$$

Therefore, we only need to solve a regularization problem with respect to a finite number of coefficients a_i.

Let $A = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^m$ be a vector consisting of some real values and let

$$f_A(\cdot) = \sum_{i=1}^{m} a_i k(\cdot, x_i) \in \mathcal{H}_X$$

be the function determined by the vector A. According to (2.5), (2.6) and (2.20), we write

$$\Phi(X) = k(\cdot, X) = (k(\cdot, x_1), \ldots, k(\cdot, x_m)) \in (\mathcal{H}^m)^{\mathrm{T}}$$

and, in particular

$$k(z, X) = (k(z, x_1), \ldots, k(z, x_m)) \in (\mathbb{R}^m)^{\mathrm{T}}, \text{ for } z \in \mathcal{X}.$$

Using this type of matrix notation, the image $f_A(z)$ of an input $z \in \mathcal{X}$ can be written as

$$f_A(z) = \sum_{i=1}^{m} a_i k(z, x_i) = k(z, X)A. \tag{2.32}$$

Similarly, the column vector $f_A(X) \in \mathbb{R}^m$, that contains the output predictions for the training examples obtained with the function f_A, is

$$f_A(X) = \begin{pmatrix} k(x_1, X)A \\ \vdots \\ k(x_m, X)A \end{pmatrix} = k(X, X)A = KA. \tag{2.33}$$

Further, the norm of f_A is

$$\|f_A\|_k = \sqrt{\langle f_A, f_A \rangle_k} = \sqrt{\sum_{i,j=1}^{m} a_i a_j k(x_i, x_j)} = \sqrt{A^{\mathrm{T}} K A}. \tag{2.34}$$

We may use these forms in (2.29) and have

$$\mathcal{A}(S) = \underset{A \in \mathbb{R}^m}{\operatorname{argmin}} \left(c(Y, KA) + \Omega \left(\sqrt{A^{\mathrm{T}} K A} \right) \right). \tag{2.35}$$

We have not yet considered the actual algorithms used to search the optimal coefficient vector that minimize the regularization problem. The difficulty of the problem depends crucially on the selection of the suitable cost function and on the regularizer. One of the greatest difficulties related to the classical machine learning algorithms is the problem caused by having several local minima. That problem can be avoided by ensuring that both the cost function and the regularizer are convex. Recall that a function g is convex if $g(zx + (1 - z)y) \leq zg(x) + (1 - z)g(y)$ for any two vectors x and y in the domain of g and any z in $[0, 1]$.

We restrict our considerations here to the following convex regularizer, sometimes called a *quadratic regularizer*, which usually leads to feasible optimization problems:

$$\Omega(\|f\|_k) = \lambda \|f\|_k^2 = \lambda A^{\mathrm{T}} K A. \tag{2.36}$$

Here $\lambda \in \mathbb{R}_+$ is called a *regularization parameter*. The regularization parameter is used to control the tradeoff between the training set performance and the hypothesis complexity. Thus, if we set $\lambda = 0$, we do not perform any regularization and we select the hypothesis that minimizes the cost function only. In this case, the hypothesis may, of course, be too complex and overfit to the training set. On the other hand, if the value of the parameter is too large, the selected hypothesis may be too simple to be able to fit properly to the training set.

2.3 Cost Functions

We can select an appropriate cost function depending on the learning problem in question. For example, if we want to maximize the accuracy when solving a binary classification task in which the class labels y are either $+1$ or -1 (see (2.51) in Section 2.4.1), a natural way would be to minimize the number of misclassifications. Let

$$\text{sign}(r) = \begin{cases} 1, & r > 0 \\ 0, & r = 0 \\ -1, & r < 0 \end{cases}. \tag{2.37}$$

be the signum function. The ith training example is misclassified by a hypothesis f if $y_i \neq \text{sign}(f(x_i))$ and hence the number of misclassification can be calculated as

$$c(Y, f(X)) = \sum_{i=1}^{m} \frac{1}{2}|y_i - \text{sign}(f(x_i))|. \tag{2.38}$$

Here the prediction $f(x) = 0$ can be interpreted, for example, so that the classifier is not taking the risk to predict the wrong class for x, and hence the cost is only $1/2$.

Unfortunately, it is reported that the use of (2.38) as a cost function usually leads to intractable optimization problems (see e.g. [64] where further references can be found). Therefore, instead of using (2.38), we use such cost functions that only approximate (2.38) and thus leads to "easier" optimization problems. Further, we use only convex cost functions to avoid the problem of having multiple local minima.

2.3.1 Support Vector Machines

We get several well-known classification algorithms with different approximations of the misclassification cost function (2.38). For example, support vector machine classifiers [72] are obtained with

$$c(Y, f(X)) = \sum_{i=1}^{m} \max(1 - y_i f(x_i), 0), \tag{2.39}$$

which is called the *hinge loss* or the *linear soft margin* loss function.

Due to the nature of the loss function, the problem to be solved is an instance of quadratic optimization problems (see e.g. [7]). However, instead of using a general purpose quadratic problem solvers, SVMs are usually trained with algorithms that are specially designed for them. One such algorithm is, for example, the sequential minimal optimization algorithm given in [49].

An algorithm that trains SVM for each value of the regularization parameter that make any difference in the form of the minimizer of the regularization problem was proposed in [21]. This type of algorithms are usually called regularization path algorithms (see e.g. [60] for more information on this topic). Briefly, the training algorithm starts from the extreme margin case where the regularization parameter is initialized to

$$\lambda = \max_{j \in \{1,\ldots,m\}} \left| \sum_{i=1}^{m} y_i k(x_i, x_j) \right| . \tag{2.40}$$

It can be shown that regularization parameter values larger than or equal to (2.40) determine a minimizer of (2.35) in which the coefficient vector satisfy $\lambda A = Y$. The algorithm then trawls through the intermediate values of λ and ends at the optimal margin, that is, the linearly separable case when $y_i f(x_i) \geq 1$ for all the training inputs. If the data are not linearly separable, λ goes all the way down to zero, and the algorithm ends at the solution with minimal training error measured by the cost function (2.39). We note that the vector A is piecewise linear with respect to λ, that is, there are only a finite number of different vectors λA along the way from the extreme margin to the minimal training error, and hence it is not necessary to train the SVM classifier with several values of λ that correspond to the same vector. As a rule of thumb, the extreme margin usually underfits, the optimal margin may overfit and the best value of the regularization parameter is somewhere in between these two extremes. However, the algorithm does not tell which value of the regularization parameter is the optimal for the classification accuracy. Therefore, it has to be found out, for example, by cross-validation. Nevertheless, [21] claim that their regularization path algorithm runs as efficiently as training SVM for only a single value of the regularization parameter, and hence it has an important computational advantage over the previously proposed SVM training algorithms.

2.3.2 Regularized Least-Squares

Regularized least-squares learners are obtained with the *least-squares cost* function,

$$c(Y, KA) = \sum_{i=1}^{m} (y_i - f(x_i))^2 = (Y - KA)^{\mathrm{T}}(Y - KA), \tag{2.41}$$

Traditionally, *regularized least-squares* (RLS) type of algorithms have been applied to regression problems [51]. However, lately they have also been used on other machine learning problems, such as classification, and their performance has been reported to be comparable to that of the ordinary support vector machines [57]. Recently, we have successfully applied RLS

to classification of clinical narratives [27, 67] and identification of multi-trace electrophoretic patterns in differential display experiments [VI]. Our another successful application area has been ranking of dependency parses [48, 69, 71].

Using the least-squares cost function (2.41) and the quadratic regularizer (2.36), we may rewrite (2.29) in a matrix form as follows

$$\mathcal{A}(S) = \underset{A \in \mathbb{R}^m}{\operatorname{argmin}} \left((Y - KA)^{\mathrm{T}} (Y - KA) + \lambda A^{\mathrm{T}} K A \right). \tag{2.42}$$

The solution can be found by first taking a gradient of the objective to be minimized with respect to A, that is,

$$\nabla_A = (-K)^{\mathrm{T}} (Y - KA) + \lambda K A.$$

Then, the gradient is set to be zero and solved with respect to A:

$$A = (KK + \lambda K)^{-1} KY. \tag{2.43}$$

If the kernel matrix is positive semidefinite instead of being strictly positive definite, the solution is not unique. However, as discussed above, each solution of (2.43) is a global optimum of (2.42), because both the cost function and the regularizer are convex. Since we only need one solution, we can (see [57]) reduce (2.43) to

$$A = (K + \lambda I)^{-1} Y. \tag{2.44}$$

The matrix $K + \lambda I$ is invertible if $\lambda > 0$, because the kernel matrix is positive semidefinite. We next consider an implementation technique for the RLS algorithm that is in many cases more practical than directly computing the inverse of the diagonally shifted kernel matrix as in (2.44).

We have made a simple implementation of the RLS algorithm which is described in [IV]. The implementation is done via the eigen decomposition of the kernel matrix. Our implementation of the RLS has the following three computational advantages as compared to the ordinary support vector machines.

- It is possible to calculate the N-fold cross-validation (CV) performance of RLS on the training data without retraining in each CV round. The proposed CV method for RLS is a generalization of the fast leave-one-out cross-validation (LOOCV) method for RLS which is widely known in the literature.

- We can compute the RLS solution for several different values of the regularization parameter in parallel.

- Several problems on the same data set can be solved in parallel provided that the same kernel function is used with each problem. We note that in the following considerations, the sequence of the outputs Y can also be defined to be a matrix $Y \in \mathbb{R}^{m \times p}$ whose rows are the vectors of the outputs, that is, it has one column per each of the p problems to be solved in parallel (see e.g. [58] for a more comprehensive discussion of this type of parallelization).

We now introduce the eigen decomposition on which our approach for training RLS learners is based on.

Let $G = (K + \lambda I)^{-1}$ denote the inverse of the diagonally shifted kernel matrix. Teaching several learners in parallel is efficient, because G has to be computed only once and it can then be used in (2.44) for each subproblem. The computational complexity of the matrix inversion is $O(m^3)$ in the worst case. Therefore, the complexity of the matrix multiplication of G with Y is negligible when the number of subproblems p is small compared to the number of training examples m.

We do not usually know in advance the optimal value of the regularization parameter λ for the learning task in question, and therefore several values have to be tested (see Section 2.4.5 for more discussion on selecting the parameters). We can train an RLS learner efficiently with several different values of λ via the eigen decomposition of the kernel matrix

$$K = V \Lambda V^{\mathrm{T}}, \tag{2.45}$$

where V is an orthogonal matrix that contains the eigenvectors of K and Λ is a diagonal matrix that contains the corresponding eigenvalues. The computational complexity of calculating the eigen decomposition of K is also $O(m^3)$ in the worst case. We observe that

$$G = (V \Lambda V^{\mathrm{T}} + \lambda I)^{-1} = V \widetilde{\Lambda}_\lambda V^{\mathrm{T}} \tag{2.46}$$

in which the diagonal matrix $\widetilde{\Lambda}_\lambda = (\Lambda + \lambda I)^{-1}$ contains the eigenvalues of G. To be exact, the element $[\widetilde{\Lambda}_\lambda]_{i,i}$ is $1/(\mu_i + \lambda)$, where μ_i is the ith eigenvalue of K. Note that we do not need to compute the matrix G, because the solution (2.44) can now be efficiently obtained by computing the following matrix products in the order determined by the parentheses

$$A = V\big(\widetilde{\Lambda}_\lambda (V^{\mathrm{T}} Y)\big). \tag{2.47}$$

Since, we can easily calculate the eigenvalue matrix $\widetilde{\Lambda}_\lambda$ for different values of λ, we can easily compute a whole array of the corresponding RLS solutions from (2.47). We can then select the regularization parameter for each subproblem with a cross-validation which we consider below.

The idea of training RLS using the eigen decomposition resembles the training of SVMs with the regularization path algorithms considered in Section 2.3.1 in the sense that both make it possible to train the learners efficiently with several values of the regularization parameter. However, in case of RLS, the coefficient vector A is not piecewise linear with respect to λ, and hence it is not fruitful to consider RLS under the concept of the regularization path.

2.3.3 Preference Learning

We can also define cost functions for learning tasks that are more complex than the binary classification. In the ranking or preference learning type of tasks (see e.g. [16]), the aim is to learn a hypothesis that is able to rank the given data points in the correct order. The order of two data points, $z_1 = (x_1, y_1)$ and $z_2 = (x_2, y_2)$, can be expressed, for example, by the difference of their outputs so that z_1 should be ranked higher than z_2 if $y_1 - y_2 > 0$ and lower if $y_1 - y_2 < 0$. Thus, this type of ranking tasks can be reduced to binary classification tasks in which the inputs to be classified are the pairs of the original inputs and the signs of the corresponding output differences indicate whether the pairs belong to the positive or negative class.

Of course, some of the data point pairs may be irrelevant for the task in question. For example, suppose that we are given a set of web-search results obtained with a set of queries and our aim is to rank the results according to the user preference. In that case, we are not interested in the order of the web documents obtained from different queries. The only relevant pairs of data points are the ones in which both of the web documents are obtained from the same query. To take into account which of the data point pairs are relevant, we define an undirected loopless graph whose vertices are the training examples and two vertices are connected with an edge if the corresponding example pair is relevant. The graph of a training set is determined by an adjacency matrix $W \in \mathbb{R}^{m \times m}$. Using the regularization framework point of view, we aim to learn a hypothesis f that predicts the order of the examples z_1, z_2 via the difference $f(x_1) - f(x_2)$. For this purpose, we have defined in [47] a cost function

$$c_W(Y, f(X)) = \frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} \left| \operatorname{sign}\big(y_i - y_j\big) - \operatorname{sign}\big(f(x_i) - f(x_j)\big) \right|. \quad (2.48)$$

However, like the misclassification cost function, the use of the above cost function leads to intractable optimization problems, and hence we have to use approximations also in this case. In [47], we have proposed the use of

the following least-squares approximation of (2.48)

$$c_W(Y, f(X)) = \frac{1}{2}\sum_{i,j=1}^{m} W_{i,j}\Big(\big(y_i - y_j\big) - \big(f(x_i) - f(x_j)\big)\Big)^2. \qquad (2.49)$$

We note that for any vector $r \in \mathbb{R}^m$ and an undirected weighted graph W of m vertices, we can write

$$\begin{aligned}
\frac{1}{2}\sum_{i,j=1}^{m} W_{i,j}(r_i - r_j)^2 &= \frac{1}{2}\sum_{i,j=1}^{m} W_{i,j}r_i^2 - \sum_{i,j=1}^{m} W_{i,j}r_i r_j + \frac{1}{2}\sum_{i,j=1}^{m} W_{i,j}r_j^2 \\
&= \sum_{i,j=1}^{m} W_{i,j}r_i^2 - \sum_{i,j=1}^{m} W_{i,j}r_i r_j \\
&= \sum_{i=1}^{m} r_i^2\left(\sum_{j=1}^{m} W_{i,j}\right) - \sum_{i,j=1}^{m} W_{i,j}r_i r_j \\
&= r^{\mathrm{T}}Dr - r^{\mathrm{T}}Wr \\
&= r^{\mathrm{T}}Lr,
\end{aligned}$$

where D is a diagonal matrix whose entries are defined as $D_{i,i} = \sum_{j=1}^{m} W_{i,j}$, and $L = D - W$ is so-called Laplacian matrix (see e.g. [8] for a typical use of the Laplacian matrix in machine learning) of the graph determined by W. Therefore, by selecting $r = Y - KA$, we rewrite the cost function (2.49) in a matrix form as

$$c_W(Y, f(X)) = (Y - KA)^{\mathrm{T}}L(Y - KA),$$

and hence the minimizer of (2.35) using this cost function can be obtained similarly as in the case of the ordinary least-squares cost function (2.41). We refer to the method minimizing (2.35) with this cost function as RankRLS.

In addition to our information retrieval experiments reported in [47], we have used RankRLS for maximization of the AUC performance measure [43] (see Section 2.4.1 for the definition of AUC). We have also developed an efficient cross-validation algorithm for RankRLS that works for certain types of ranking tasks [46], and a sparse version of the algorithm that can be used for large scale learning [68].

2.4 Measuring Learner Performance

When studying machine learning methods, we use functions that measure, on a given test data set, how well the learner is able to predict the outputs of the test data points. Before the measurement, a supervised learner is trained with a training set that is independent of the test set. The results of the

performance measurements are also used for parameter selection purposes. Therefore, the estimation usually takes two rounds, namely the validation round with an independent data set for parameter selection and a final test round with another independent data set called a test set. In this section, we give a brief presentation of the performance measures that are relevant for the thesis. The measures, of course, depend on the learning task in question. We first present a general formulation for the performance evaluation functions. Next, in Section 2.4.1, we give examples of different performance measures used in the task of binary classification. Performance measures used in the ranking tasks are considered in Section 2.4.2. Finally, we consider the hold-out and cross-validation techniques in Section 2.4.3

As before, we denote $X = (x_1, \ldots, x_m) \in (\mathcal{X}^m)^{\mathrm{T}}$, $Y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, and $f(X) = (f(x_1), \ldots, f(x_m)) \in \mathbb{R}^m$, for a given set $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathbb{R})^m$ of training examples and a hypothesis f. In the following definitions of measures we use the training data, but the measures can, of course, be analogously defined for any sequence of data points. We consider performance evaluation functions of type

$$p : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}. \tag{2.50}$$

Each function assigns a value

$$p(Y, f(X))$$

which measures how well the hypothesis f fits to S. How the value of the function is calculated, depends on the learning task in question.

2.4.1 Measures for Binary Classification Learning

Classification accuracy, that is, the number of correctly classified examples divided by the number of all examples, is a simple and intuitive way to measure the performance of a binary classifier. In binary classification, we assume that the outputs are either $+1$ or -1. Then, the realization of (2.50) for the performance measure accuracy is the following sum over the data points

$$p(Y, f(X)) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |y_i + \operatorname{sign}(f(x_i))|. \tag{2.51}$$

Note the similarity of the binary classification accuracy (2.51) and the training error cost function (2.38). Thus, if the aim is to learn a classifier having a maximal binary classification accuracy, it is natural to select a hypothesis that minimizes (2.38) on the training data set.

When we have a balanced class distribution, that is, there are approximately equal number of positive and negative examples in the data set,

the accuracy of a classifier that assigns the classes randomly has an accuracy close to 0.5, while a perfect classifier has an accuracy 1. However, the classification tasks have often skewed distributions of positive and negative examples. For example, in our gene versus protein name disambiguation study [I], the number of protein examples is 30269 which is about three times as big as the number of gene examples 9533. Thus, if we select a hypothesis f such that $f(x) = 1$ for all $x \in \mathcal{X}$, that is, a classifier that always predicts the protein class, the performance measure (2.51) gives us a classification accuracy of about 0.75.

To cope with the imbalance in the data, we measure the performance of a classifier as the area under the ROC curve (AUC) (see e.g. [14, 36, 53]). Unlike other popular measures such as accuracy and precision-recall analysis, the AUC measure is invariant to the prior class probabilities. AUC corresponds to the probability that given a randomly chosen positive example and a randomly chosen negative example, the classifier will correctly say which is which. Formally, for a training set S and a classifier f, the AUC measure can be calculated from the following formula which is also called the Wilcoxon-Mann-Whitney statistic (see [11] for a proof):

$$p(Y, f(X)) = \frac{1}{|P||Q|} \sum_{i \in P, j \in Q} \frac{1}{2}(1 + \text{sign}(f(x_i) - f(x_j))), \tag{2.52}$$

where $P \subset \{1, \ldots, m\}$ and $Q \subset \{1, \ldots, m\}$ are index sets containing the indices of the positive and negative examples in S, respectively. We use the AUC performance measure in all our studies that involve binary classification.

2.4.2 Measures for Ranking Performance

We now consider learning problems, where the task is to predict a *ranking* for a given set of data points. Let (x_1, y_1) and (x_2, y_2) be two unseen data points and $f(x_1)$ and $f(x_2)$ their predicted outputs obtained with a learner f. While the learning machine makes an output prediction for each unseen data point, the goal is to ensure that $f(x_1) < f(x_2)$ whenever $y_1 < y_2$. In this case, it is not important how close the predicted outputs are to the true values of the outputs. We present the measure used to evaluate the ranking performance in our experiments with parse ranking [48, 69, 71], and [IV], and ranking of clinical narratives [67], namely the Kendall's τ_b *correlation coefficient*. For more information on this type of ranking measures, we refer to [32].

The vector Y of outputs determines a complete ranking for the data points (x_i, y_i) so that when $\text{sign}(y_i - y_j) = 1$, we say that $s_i = (x_i, y_i)$ is ranked higher than $s_j = (x_j, y_j)$. On the other hand, when $\text{sign}(y_i - y_j) = 0$ or $\text{sign}(y_i - y_j) = -1$, the ranks of s_i and s_j are tied or s_i is ranked lower

that s_j, respectively. Similarly, a complete ranking for the data points is determined by the vector $f(X)$ of predicted outputs. We now define the Kendall's τ_b *correlation coefficient* for the rankings corresponding to Y and $f(X)$ as follows:

$$p(Y, f(X)) = \frac{\sigma(Y, f(X))}{\sqrt{\sigma(Y,Y)\sigma(f(X), f(X))}},$$

where

$$\sigma(a, b) = \sum_{i,j=1}^{m} \text{sign}(a_i - a_j)\text{sign}(b_i - b_j),$$

for $a = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^m$ and $b = (b_1, \ldots, b_m)^{\mathrm{T}} \in \mathbb{R}^m$. We observe that $p(Y, f(X)) = 1$ when the rankings determined by Y and $f(X)$ are completely correlated. On the other hand, $\tau_b = -1$ when the rankings is are the reverse of each other. In all other cases, the value of τ_b lies between -1 and 1.

2.4.3 Hold Out and Cross-Validation

The learning algorithms discussed above have parameters that must be selected before the learners are trained. The regularized kernel methods that use a regularizer of the form (2.36) have a regularization parameter λ. We do not usually know in advance which value of λ provides us the best performance for unseen data. Moreover, the kernel function that determines the hypothesis space has usually also parameters to be selected. The degree of the polynomial kernel (2.9) is one such parameter, for example. One of the most popular ways to obtain empirical estimates of the learner performance on an unseen data is to use so-called hold-out estimation technique. By *hold-out*, we mean that a part of the available data set is set aside to form a hold-out set for performance estimation which is independent of the training set used to train a learner. *Cross-validation* (CV) is a method in which the hold-out technique is repeated with different (usually mutually exclusive) hold-out sets of the data set. We now present a formalization for these techniques.

Let $H \subseteq \{1, \ldots, m\}$ denote an index set in which the indices refer to the examples in the training set that belong to the hold-out set, and let $C = \{1, \ldots, m\} \setminus H$ be the set indexing the rest of the training examples. Let $X_H \in (\mathcal{X}^{|H|})^{\mathrm{T}}$, $Y_H \in \mathbb{R}^{|H|}$, and $S_H \in (\mathcal{X} \times \mathbb{R})^{|H|}$ denote, respectively, the sequence containing only the inputs, outputs, and training examples that are indexed by H. We call S_H a hold-out set. Further, let $f_C = \mathcal{A}(S_C)$ denote a learner that is trained using only the training examples S_C, where S_C is defined analogously to S_H. Then, $f_C(X_H)$ consists of the output values for the hold-out examples X_H that are predicted by f_C. The hold-out performance of a learner, that is, the performance of a learner f_C on the

hold-out set S_H is defined by

$$p(Y_H, f_C(X_H)).$$

To formalize the CV technique, we follow [22] and define an indexing function that determines a partition of the training set

$$\varphi : \{1, \ldots, m\} \to \{1, \ldots, N\},$$

where N is the size of the partition, so that each training example is assigned a fold number among $\{1, \ldots, N\}$. Let $H_j = \{i \mid \varphi(i) = j\}$ for all $1 \leq j \leq N$, that is, H_j is the index set containing the indices of the training examples that belong to the jth cross-validation fold. *CV performance estimate* is defined to be the average of the hold-out estimates obtained with the N mutually exclusive hold-out sets (called cross-validation folds in context of CV) determined by a partition φ. Thus, it can be computed from

$$p_{\mathrm{CV}}(S, \mathcal{A}, \varphi) = \frac{1}{N} \sum_{j=i}^{N} p(Y_{H_j}, f_{C_j}(X_{H_j})).$$

For the statistical properties of the CV estimators, we refer to [34]. The CV performance estimates, of course, depend of the size N of the CV folds. The smaller is the size of a fold, the closer the learner trained with the rest of the data set usually is to the learner trained with the whole data set. The estimates also depend of the selected partition φ. There are, in fact, $\binom{m}{m/N}$ possibilities to choose a hold-out set of size m/N out of m examples and the performance differences between the different possibilities may be large.

An extreme case of the CV, in which $N = m$, is called a leave-one-out cross-validation (LOOCV). LOOCV is known to be an almost unbiased estimator of the learning performance. However, the variance of the LOOCV may be larger than that of the N-fold cross-validation. This was experimentally demonstrated in [34], and the parameter selection method recommended there for classifiers was a stratified ten-fold cross-validation repeated ten times with different fold partitions (by a stratification we mean that the CV fold partition is formed so that the class distribution of examples in them is approximately the same as the distribution in the whole training set). Indeed, in our word sense disambiguation study [45], the size of the training data set was so small that we had to use repeated CV in order to stabilize the parameter estimation procedure.

2.4.4 Computation of Hold Out for RLS Learners

If the learner has to be retrained each time a hold-out estimate is computed, the calculation of the average CV performance may be computationally too

expensive. This is especially true when repetition with different partitions or LOOCV is used. Fortunately, with some learning algorithms, the training examples in the hold-out set can be efficiently "unlearned" and the retraining does not have to be performed. By unlearning, we mean the elimination of the effect of the hold-out set from the learner trained with the whole data set, that is, the learner after the unlearning is equal to the one that would be obtained with a training set from which the examples that belong to the hold-out set are removed.

We now consider an efficient method to calculate the hold out performance of the RLS learners. The formulation can then be used, for example, to perform a cross-validation by holding out a different part of the data set at a time and averaging the results.

Below, with any matrix that has its rows indexed by the training examples $M \in \mathbb{R}^{m \times p}$, where $p \in \mathbb{N}$, we use the subscript H so that a matrix $M_H \in \mathbb{R}^{|H| \times p}$ contains only the rows that are indexed by H. For $M \in \mathbb{R}^{m \times m}$ we also use $M_{HH} \in \mathbb{R}^{|H| \times |H|}$ to denote a matrix that contains only the rows and the columns that are indexed by H.

We can, of course, use the naive approach of training f_C first, and then computing $f_C(X_H)$. According to (2.44) and (2.32), this can be formally written as

$$f_C(X_H) = K_{HC}(K_{CC} + \lambda I_{CC})^{-1} Y_C. \tag{2.53}$$

The computation of (2.53) is, however, too cumbersome for cross-validation purposes, because RLS has to be retrained separately for each different hold-out set. In [IV], we prove that $f_C(X_H)$ can also be obtained from

$$f_C(X_H) = (I_{HH} - B_{HH})^{-1}((BY)_H - B_{HH} Y_H), \tag{2.54}$$

where $B = K(K + \lambda I)^{-1}$. The computation of (2.54) is much less computationally complex than the naive method (2.53), because we only have to invert a matrix $I_{HH} - B_{HH}$ of size $H \times H$ and it can be easily generated when the RLS with the whole data set has already been trained. Namely, the elements of B_{HH} can be computed using the eigenvectors V and the diagonal elements of $\Lambda\widetilde{\Lambda}_\lambda$, since according to (2.45) and (2.46) $B = V\Lambda\widetilde{\Lambda}_\lambda V^{\mathrm{T}}$ and therefore $B_{HH} = V_H \Lambda \widetilde{\Lambda}_\lambda (V_H)^{\mathrm{T}}$. Thus, the computational complexity of calculating the outputs $f_C(X)$ for the hold-out set is dominated by the complexity of calculating $B_{HH} =$ which is $O(|H|^2 m)$. The equation (2.54) was proved using a technique similar to the leave-one-out lemma (see e.g. [73]).

We now introduce even simpler formulation for obtaining the hold-out performance estimate and present a proof that uses only linear algebra techniques. Before we give the formulation, we first present the following lemma which is often called the block inverse (see e.g. [28]).

Lemma 1. *Let $M \in \mathbb{R}^{m\times m}$ be a positive definite matrix, $H \subset \{1,\ldots,m\}$, and $C = \{1,\ldots,m\} \setminus H$. Without losing generality, by reindexing we may write M as a following block matrix*

$$M = \begin{bmatrix} M_{HH} & M_{HC} \\ M_{CH} & M_{CC} \end{bmatrix}.$$

Then the inverse of M is

$$\begin{bmatrix} R^{-1} & -R^{-1}M_{HC}(M_{CC})^{-1} \\ -(M_{CC})^{-1}M_{CH}R^{-1} & (M_{CC})^{-1} + (M_{CC})^{-1}M_{CH}R^{-1}M_{HC}(M_{CC})^{-1} \end{bmatrix},$$

where R is the Schur complement of M_{CC}, that is,

$$R = M_{HH} - M_{HC}(M_{CC})^{-1}M_{CH}. \tag{2.55}$$

Let $M = K + \lambda I$ and $G = M^{-1}$. Using the block inverse, we can derive the following simple formulation for the RLS hold out.

Proposition 1. *The matrix $f_C(X_H)$ consisting of the output values of the hold-out examples predicted with f_C can be obtained from*

$$f_C(X_H) = Y_H - (G_{HH})^{-1}A_H. \tag{2.56}$$

Proof. We start by noting that the matrix G_{HH} is a positive definite, because it is a principal submatrix of a positive definite matrix G, and therefore, it is invertible for any H (see e.g. [28]). By reindexing we may write the kernel matrix K as the following block matrix

$$K = \begin{bmatrix} K_{HH} & K_{HC} \\ K_{CH} & K_{CC} \end{bmatrix}. \tag{2.57}$$

Recall that the output $f_C(X_H)$ can be computed naively by training an RLS without the examples indexed by H as in (2.53). We start from the formulation of the naive method (2.53) and show the equivalence with (2.56).

$$\begin{aligned} f_C(X_H) &= M_{HC}(M_{CC})^{-1}Y_C \\ &= (G_{HH})^{-1}G_{HH}M_{HC}(M_{CC})^{-1}Y_C \\ &= -(G_{HH})^{-1}G_{HC}Y_C \qquad (2.58) \\ &= Y_H - Y_H - (G_{HH})^{-1}G_{HC}Y_C \\ &= Y_H - (G_{HH})^{-1}G_{HH}Y_H - (G_{HH})^{-1}G_{HC}Y_C \\ &= Y_H - (G_{HH})^{-1}(G_{HH}Y_H + G_{HC}Y_C) \\ &= Y_H - (G_{HH})^{-1}G_H Y \\ &= Y_H - (G_{HH})^{-1}A_H \end{aligned}$$

where the tenability of the equality (2.58) can be seen by observing that $-G_{HH}M_{HC}(M_{CC})^{-1} = G_{HC}$ from the formula of the block inverse (Lemma 1). □

If we have already calculated the eigen decomposition of the kernel matrix when training an RLS with the whole data set, the computational complexity of calculating the outputs $f_C(X_H)$ using (2.56) is dominated by the complexity of calculating $G_{HH} = V_H \widetilde{\Lambda}_\lambda (V_H)^{\mathrm{T}}$ which is $O(|H|^2 m)$, because $\widetilde{\Lambda}_\lambda$ is a diagonal matrix. Therefore, performing an N-fold cross-validation with the training set has an overall computational complexity $O(n(m/n)^2 m) = O(m^3/n)$, since the number and the size of the hold-out sets are N and m/n, respectively.

2.4.5 Parameter Selection

The learning algorithms considered in this thesis have parameters that should be selected on the basis of the training data. For example, the regularization parameter λ of the regularized kernel methods and the parameters of the kernel functions are such that we do not usually know their optimal values before seeing the training set. In order to fit the parameters to the training set, we use a grid search over the parameter space. For example, when selecting the regularization parameter together with the width of a Gaussian kernel, we may create the following set of candidate parameter combinations $\mathcal{B} = \{(2^i, 2^j) \mid -5 \leq i, j \leq 5\}$. We select the parameter combination which gives the best cross-validation performance estimate on the training set. Formally, let $\mathcal{B}$ be a finite set of parameter combinations that can be used to train a learner. The optimal parameter combination b^* is

$$b^* = \operatorname*{argmax}_{b \in \mathcal{B}} p_{\mathrm{CV}}(S, \mathcal{A}_b, \varphi), \tag{2.59}$$

where $\mathcal{A}_b$ is a learning algorithm parametrized by $b \in \mathcal{B}$ and φ is an indexing function determining a fold partition for cross-validation.

Chapter 3

Learning Tasks

In this chapter, we briefly discuss the domains with which we have evaluated the methods described in Chapter 2. In Section 3.1, we discuss about a task that we call here as natural language disambiguation. In Section 3.2, we consider ranking tasks in the domain of information retrieval. Then, we discuss about multi-trace electrophoretic patterns in differential display experiments in Section 3.3. The task of dependency parse ranking is discussed in Section 3.4.1 and the classification as well as ranking of the clinical narratives are discussed in Section 3.4.2.

3.1 Natural Language Disambiguation

The amount of ambiguity in natural language is large and it causes different kinds of problems in the field of automatic natural language processing. In this thesis, we consider three kinds of disambiguation tasks, namely the word sense disambiguation in Section 3.1.1, context sensitive spelling error correction in Section 3.1.2, and gene versus protein name disambiguation in Section 3.1.3. On these tasks, we apply the kernel functions and feature transformations that we have proposed. The classifiers that we use together with the proposed kernels or transformations are the support vector machines in [I-III] and [45], and the flexible Bayes kind of classifiers in [45] and [44]. There is also a lot of other research on designing kernel functions for the natural language disambiguation tasks (see e.g. [19]).

3.1.1 Word Sense Disambiguation

The most common disambiguation problem in natural language is the process of resolving the sense of a word or some other linguistical entity, namely the task of word sense disambiguation (WSD) (see e.g. [29, 38]). The word "bank" is an example traditionally used to explain what is WSD. The word can refer to a river bank, a financial institution, or the building where a

financial institution resides, and even subtler nuances can be found. This is not, however, the only possible formulation of the problem, but it can be formulated in a wider way. For example, one needs to decide, whether a bank is situated in some specific area in case banks from several areas are considered in the same text. WSD is an essential part of the NLP and has been a long time under scientific research. It is not of much use for end-users as such, but rather an intermediate task of several other NLP tasks (see e.g. [29]).

We consider WSD as a classification problem in which each sense of a word is considered as a class. The classification performance of the WSD systems are often evaluated and compared with each other using a publicly available annotated data set. A popular choice is the Senseval-3 English lexical sample train and test datasets [39], which we use in [45].

3.1.2 Context-Sensitive Spelling Error Correction

Context-sensitive spelling error correction is a classification task which is closely related to WSD. The task refers to a problem, for example, in which the word "desert" is misspelled as "dessert" (see e.g. [20]). The word "dessert" belongs to the English lexicon and therefore, the mistake will be unnoticed by the lexicon-based spellcheckers. A set of similar words, such as "desert" and "dessert", that belong to the lexicon and that are often confused with the other words in the set is called a *confusion set*.

Similarly to the other WSD problems, we cast context-sensitive spelling error correction as a classification task such that each word of the confusion set is considered as a class. Namely, the task is to select the correct spelling from the confusion set to which the word to be spellchecked belongs. This selection can be performed on the grounds of the context. The context-sensitive spelling error correction task is ideal for evaluating classification systems, because the data can be easily obtained automatically for these problems, whereas data for the other kinds of WSD problems may have to be annotated by hand. As high-quality texts such as newswire articles are widely available, the required examples for the classification experiments can simply be extracted from such resources. For example, in [20] a context-sensitive spelling correction system was tested on data sets of commonly confused words extracted from the Reuters News corpus [59]. In our experiments with binary classifiers, we use the binary confusion sets among the ones created by [20].

3.1.3 Gene versus Protein Name Disambiguation

In order to find the relations between biological or chemical entities, first the names of the entities have to be recognized in a reliable way. There

has been a significant amount of effort to do that automatically (see e.g. [33] for a recent comparison of the state-of-the-art systems for this task). A large standardized domain corpus helps to consolidate the research efforts. The GENIA corpus [9] has been commonly used in biomedical named entity recognition.

The task of named entity recognition can be divided in two subtasks, the identification of entities, that is, determining the boundaries of the named entities, and their classification into proper classes. In our study [I], we have focused on the problem of finding the class the entity belongs to. The entities in biomedical text are highly ambiguous. For example, it is common that a gene has the same name as the protein it codes for. Such an ambiguity is illustrated in the two sentences given by [23]:

- *By UV cross-linking and immunoprecipitation, we show that SBP2 specifically binds selenoprotein mRNAs both in vitro and in vivo.*
- *The SBP2 clone used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3′ UTR truncated at the polyadenylation site).*

The occurrence of SBP2 is a protein in the first sentence, whereas the occurrence of SBP2 in the second sentence is a gene. In the same study, a domain corpus was annotated by three biology experts. The three experts unanimously agreed only in 78% of the cases, each name being classified as either a gene, protein or mRNA. This low rate of inter-annotator agreement suggests that the task is relatively difficult even for human experts. However, the study does not analyse more closely the reasons that lead to annotation disagreements.

In our study [I], we have considered the disambiguation of the sense "gene" or "protein" when the name is not disambiguated explicitly by the author with the word "gene" or "protein" (e.g. "SBP2 gene"). This task is important, because the release of the human genome and large scale functional genomics studies and methods have made it important to be able to find information from literature specifically for proteins and the corresponding genes. For example, in data mining related to proteomics the scientists could save much time if they could direct their literature searches only to proteins, since the searches provide a lot of hits among which the correct and important articles have to be sorted manually.

3.2 Learning to Rank for Information Retrieval

One of the most important tasks in information retrieval is the ranking of documents based on their relevance to a query (for more on information retrieval, we refer to [5]). Much effort has been placed on the development of

ranking functions and applying machine learning techniques to learn them is viewed as a promising approach (see e.g. [30]). The aim is to develop methods to automatically learn a function from training data, such that the function can sort documents according to their degrees of relevance with respect to, for example, a certain query.

Letor (LEarning TO Rank) [37] is a collection of data for evaluation of learning to rank methods that contain features and relevance degrees for training. The collection consists of a set of document-query pairs. Each document-query pair is represented as an input consisting of a small number of features that indicate how well the document matches the corresponding query. The document-query pairs are also associated with a relevance degree indicating whether (or how much) the document is relevant to the query. The machine learning methods are trained to predict the relevance degrees of the document-query pairs that are associated with unseen queries. The predicted degrees should be such that the documents being relevant to the query in question are ranked higher than the less relevant ones. The data sets are, of course, partitioned according to the queries, that is, the rankings are considered for each query separately and we are not interested in the mutual order of the query-document pairs that are associated with different queries.

In paper [V], we proposed the RankRLS algorithm (see Section 2.3.3) and evaluated its ranking performance with the Letor dataset. The performance was compared to that of RankSVM [25] and RankBoost [15], two algorithms which can also be efficiently trained in such setting. It was shown that RankRLS achieved performance comparable to the two baseline methods on the task of ranking query-document pairs. Thus, RankRLS appears to be a simple and efficient alternative to the state of the art rank learning algorithms for document retrieval tasks.

3.3 Pattern Recognition in Differential Display Experiments

So-called mRNA differential display (DD) [35] is a widely used method to identify differentially expressed genes [13] that is based on comparison of labeled DNA fragments separated in electrophoresis. The method enables weekly production rate of digital data from tens of thousands multiplexed electrophoretic samples containing hundreds of thousands fragment peaks. However, visual analysis of these results is usually infeasible, and hence automatic analysis tools are needed for the detection of differential expressions.

To address the need of more detailed analysis in terms of peak pattern comparisons across pairs of electrophoretic traces, a method for computer-assisted ranking of such patterns in DD experiments was developed by [2].

The ranking method aims at identifying the most potential findings to be visually confirmed among the massive number of all peak patterns. Therefore, the researcher will finally evaluate visually only the automatically prescreened findings, drastically reduced in number and enriched in relevance. The ranking method has successfully been used to detect changes in expression patterns, for example, in response to human colonic carcinoma [3]. However, as the trace comparisons were based on a pairwise alignment algorithm, the method is not applicable to designs involving simultaneous comparisons between multiple traces.

With this task, we used an RLS learner with an eight-degree polynomial kernel. The experiments are reported in [VI] in which it is shown that while a prediction function designed by a human expert has the highest performance, a function that is automatically learned by a machine learning algorithm also obtains a competitive performance.

3.4 Other Applications

In addition to the above described applications, we have used the methods described in Chapter 2 also for following tasks in papers not included in this thesis.

3.4.1 Ranking of Dependency Parses

Ranking, or ordinal regression, has many applications in NLP, for example, parse re-ranking (see e.g [10]) and re-ranking of predicate argument structures (see e.g. [42]). In [71], we study the task of parse re-ranking in the domain of biomedical texts. The link grammar (LG) parser [66] used in the study is a full dependency parser based on a broad-coverage hand-written grammar. The LG parser generates all parses allowed by its grammar and applies a set of built-in heuristics to rank the parses. However, the ranking performance of the heuristics has been found to be poor when applied to biomedical text [55, 56]. Therefore, a primary motivation for the work is to present a machine learning approach for the parse re-ranking task in order to improve the applicability of the parser to the domain. A method based on the RLS algorithm is trained to regress the goodness values of the candidate parses obtained from a hand-annotated corpus introduced in [17, 54]. The regressed outcomes are then used to generate a ranking for each sentence.

In [69, 70], we developed a kernel function for sequential data and showed that the parse ranking performance of RLS is further improved when using the new kernel. We continued the seek for good data representations and kernels in [48], and considered how much effect different graph based representations and kernels have on ranking performance (see e.g. [1, 41] and

references therein for more information about kernels for dependency structures). However, it turned out that by using a cost function suitable for ranking problems that we developed in [47] and extended for large scale learning in [68] provided much larger performance gain than engineering the kernel functions.

3.4.2 Classification and Ranking of Clinical Narratives

Health-care providers have recently been changing paper-based patient records to electronic ones, and hence new possibilities to take advantage of the gathered data in intensive care have emerged. For example, tools that classify the nursing narratives automatically make it easier for a nurse to retrieve relevant information in order to build a general picture. To serve as a basis for these kinds of tools, we have applied the RLS algorithm for multi-label classification of pieces of Finnish intensive care nursing narratives according to their content [26, 27]. By multi-label classification, we indicate that each text piece can belong to several classes simultaneously but it does not necessarily belong to any class. The classes used in the studies are breathing, blood circulation and pain.

In some cases, it is more practical to rank the texts in order of importance with respect to the given class than to classify them. In other words, the more strongly the statement relates to the given class, the higher rank it gets. In [67], we have presented an adaptation of the regularized least-squares regression algorithm to ranking pieces of nursing notes according to their relevance to breathing, blood circulation, and pain.

Chapter 4

Summary of publications

The thesis constitutes of six publications. The topics of the articles are summarized in Table 4.

	I	II	III	IV	V	VI
Kernels	x	x	x			x
Algorithm development			x	x	x	
Natural language disambiguation	x	x	x			
Information retrieval					x	
Pattern recognition						x

Table 4.1: Summary of the topics and articles. The columns correspond to the papers included in this thesis and the rows to the considered topics.

Paper I: Contextual Weighting for Support Vector Machines in Literature Mining: an Application to Gene versus Protein Name Disambiguation

In the paper [I], we explore the capability of the support vector machines (SVM) for the gene versus protein name disambiguation task discussed in Section 3.1.3. We incorporate into the conventional SVM a weighting scheme developed in [17, 18]. The weighting is based on distances of context words from the word to be disambiguated. The words that are close to the name to be disambiguated are given larger weights than the words that are far. The weighting scheme increased the performance of SVMs by five percentage points giving performance better than 85% as measured by the area under ROC curve and outperformed the weighted additive classifier developed in [17, 18], which also incorporates the weighting, and the naive Bayes classifier. We also measured the classification performance of the SVMs with the linear, inhomogenous polynomial, and Gaussian kernels with and without

the weighting scheme. We conclude that the performance gain obtained by using the weighting is larger than the performance differences between the different kernels or classification methods.

Although not directly discussed in [I], the weighting can be considered as a linear transformation of word-position feature vectors (see Section 2.1.2 and [III]). Namely, the word-position features are first weighted according to their positions and subsequently transformed to a weighted bag-of-words vectors.

Paper II: Kernels incorporating word positional information in natural language disambiguation tasks

In the paper [II], we introduce a new kernel function designed for the problem of word sense disambiguation. In the article, the proposed kernel is considered as a nonlinear one while it can also be interpreted as a linear feature transformation. This is explicitly shown in [III]. Similarly to the weighting scheme considered in [I], the word-position features are weighted according to their positions. Additionally, the kernel also transforms the word occurrences to the nearby positions instead of identifying every position as it is done when constructing bag-of-words vectors. Unlike the approach proposed in paper [I], the new kernel also takes account of the mutual positional similarities of the context words in an advantageous way when calculating the similarities between the contexts around the ambiguous words. We apply the kernel to context-sensitive spelling correction (see Section 3.1.2) together with SVM classifiers and show that it statistically significantly outperforms the standard bag-of-words kernel and the kernel-based on the contextual weighting scheme.

Paper III: Matrix Representations, Linear Transformations, and Kernels for Disambiguation in Natural Language

In the paper [III], we describe a framework for natural language disambiguation tasks that is based on a word-position matrix representation of text, linear feature transformations of the word-position matrices, and kernel functions constructed from the transformations. The contextual weighting used in [I] and the kernel functions proposed in [II] and in [45] are special instances of this framework. In addition to the positional transformations of word-position features that we have considered in the previous papers, we consider here also the word transformations that can be used to incorporate prior knowledge of the word similarities into the learners.

We introduce several ways to obtain both kinds of transformations. For example, we show how the information of the most common parts-of-speech of the context words can be encoded into a word transformation. We also

introduce a positional transformation that has only one parameter but gives as good or even better results than the positional transformations considered in [I, II], [45], and [44]. Moreover, we introduce a way to construct both types of transformations automatically from a data set. Finally, we present efficient algorithms for calculating kernel functions that correspond to the transformations.

The proposed framework provides an elegant way to use both types of transformations simultaneously, improving further the performance of the kernel-based learning algorithms for certain NLP tasks. We demonstrate the performance gain by experiments of context sensitive spelling error correction with the Reuters News corpus using a regularized least-squares classifier.

Paper IV: Fast n-Fold Cross-Validation for Regularized Least-Squares

The paper [IV] considers our implementation of the regularized least-squares (RLS) algorithm. The implementation takes advantage of the following three properties. Firstly, it is possible to calculate the hold-out performance estimates for a RLS learner efficiently provided that the learner is already trained with the whole training set. In the paper [IV], we give a formal proof for this claim and a more sophisticated version of it is given in Section 2.3.2. The hold-out performance estimates can be used to calculate the cross-validation (CV) performance of RLS on the training data without retraining in each CV round. This decreases the computational complexity of calculating the CV performance, because unlearning the hold-out set is more efficient than retraining the learner. Secondly, it is possible to compute the RLS solution for several values of the regularization parameter at the same time without retraining the learner separately for each different value. This can be combined with the fast hold-out computation so that we can, for example, select the regularization parameter which has the best CV performance. Finally, several problems on the same data set can be solved in parallel provided that the same kernel function is used with each problem. Again, this can be combined with the above two properties so that the regularization parameters can be efficiently selected for each problem separately.

Further, we show that for some tasks, the leave-one-out cross-validation (LOOCV) gives poor performance estimates for the learning machines, because of the dependencies between the training examples. We demonstrate this by experimentally comparing the performance estimates given by LOOCV and CV in a ranking task of dependency parses generated from biomedical texts. The fold partition used in the CV is constructed so that the effect of the dependencies between the training examples is removed. In

summary, our CV algorithm makes it possible to efficiently calculate reliable performance estimates for RLS learners with arbitrary fold partitions.

Paper V: Learning to Rank with Pairwise Regularized Least-Squares

We continue our study of RLS based learning algorithms in paper [V] and develop a simple preference learning algorithm we call RankRLS. As described in Section 2.3.3, RankRLS minimizes a regularized least-squares approximation of a ranking error function that counts the number of incorrectly ranked pairs of training examples. Both primal and dual versions of the algorithm are considered in the paper. By primal version, we refer to the case discussed in Section 2.1.2 in which a linear kernel is used and the learned ranking function can be expressed as a vector in the input space. We show that both versions of RankRLS can be trained as efficiently as the corresponding primal and dual versions of the standard RLS regression, despite the fact that the number of training example pairs under consideration grows quadratically with respect to the number of individual examples. Further, similarly as in paper [IV], we show that it is possible to compute the RankRLS solution for several values of the regularization parameter at the same time without retraining the learner separately for each different value.

Paper VI: Computer-Assisted Identification of Multi-Trace Electrophoretic Patterns in Differential Display Experiments

The paper [VI] proposes two methods for scoring and ranking of multi-trace peak patterns in differential display experiments (see Section 3.3). The peak patterns are obtained by aligning sets of traces with a multiple alignment algorithm that is also described in the paper. The paper focuses on the user defined computer-assisted method that uses strong prior knowledge of the type of the patterns that a human expert prefers. The part which is relevant for this thesis is a data driven method that automatically learns to identify the preferable patterns, thus trading off some of the prior knowledge for automation of the identification process. This method is implemented with a regularized least-squares regressor and a homogenous high-degree polynomial kernel function. It is shown that while scoring function programmed by a human expert has the best performance, the performance of automatically learned method is also surprisingly good.

Chapter 5

Conclusions

5.1 Achievements of the Thesis

In this thesis, we made several adaptations of regularized kernel methods for tasks in natural language processing and bioinformatics. In particular, we aimed to develop kernel functions that incorporate prior knowledge about the learning task under consideration. Another aim was to develop efficient algorithms for training the kernel-based learning algorithms.

Incorporating Prior Knowledge Taking advantage of the positional information of the context words around the term to be disambiguated is a typical use of prior knowledge in the task of natural language disambiguation. We incorporated into the conventional SVM a weighting that is based on distances of the context words from the focus word and introduced kernel functions incorporating the word positional information in even more advantageous way than just taking into account the distances. Further, we described a framework in which the kernels could be considered as linear feature transformations. This allowed the simultaneous incorporation of both word positional and word-word similarity information into kernel-based natural language disambiguation algorithms. Altogether, the proposed framework provides an elegant way to encode prior knowledge into a learning machine, leading to an improved disambiguation performance.

We also used our prior knowledge of the learning task at hand to develop a new cost function. The new ranking cost function was shown to have better performance in information retrieval tasks and in maximization of the AUC classification performance measure than a similar type of regression cost function.

Developing Efficient Algorithms We considered computationally efficient algorithms for calculating the kernels and for training machine learning

algorithms that employ the kernel functions incorporating the word position and word-word similarity information. We also developed a highly efficient algorithm for predicting the outputs of unseen data points when the learning algorithm is already trained.

We also proposed an efficient method of computing a hold-out performance for RLS. This is especially suitable for cases in which there are such dependencies in the training data that the standard leave-one-out cross-validation would give overoptimistic results. In this thesis, we also proposed a refined version of the efficient hold-out method. The method can the be used to perform, for example, N-fold or leave-cluster-out cross-validation efficiently.

The task of ranking or preference learning is often cast to the problem of binary classification of example pairs so that the pair is considered to belong in the positive class if the first example is preferred over the second one and in the negative class in the opposite case. However, since the number of training example pairs grows quadratically with respect to the number of individual examples, the training of a classifier may become too expensive. We proposed RankRLS, a modification of the regularized least-squares (RLS) algorithm that corresponds to a RLS classifier of example pairs. The training of RankRLS is computationally as complex as the training of a RLS classifier or regressor for the individual examples. Hence, the problem of having a quadratic number of training example pairs is avoided.

5.2 Future Work

The use of the linear feature transformations to construct kernel functions is a promising approach in general, because it provides an elegant and efficient way to incorporate external information into learning methods. We will investigate whether the use this technique is advantageous also in other applications than those considered in this thesis.

As considered in this thesis, the standard RLS regression algorithm has many computational advantages such as the efficient cross-validation, regularization, and learning multiple outputs simultaneously. The efficient regularization and multiple output learning can be quite straightforwardly transferred to RankRLS. In [46], we described a cross-validation algorithm for RankRLS that works for certain types of ranking tasks. Since hold-out and cross-validation are very important methods in performance evaluation of the learning methods especially if the amount of labeled data is small, we will continue this investigation and aim to design cross-validation algorithms also for more general ranking and preference learning tasks.

Bibliography

[1] Antti Airola, Sampo Pyysalo, Jari Björne, , Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. Graph kernel for protein-protein interaction extraction. In *BioNLP 2008 Workshop*, 2008. To appear.

[2] Tero Aittokallio, Pekka Ojala, Timo J. Nevalainen, and Olli Nevalainen. Automated detection of differentially expressed fragments in mrna differential display. *Electrophoresis*, 22(10):1935–1945, Jun 2001.

[3] Tero Aittokallio, Tapio Pahikkala, Pekka Ojala, Timo J. Nevalainen, and Olli Nevalainen. Electrophoretic signal comparison applied to mrna differential display analysis. *BioTechniques*, 34(1):116–122, Jan 2003.

[4] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3), 1950.

[5] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN 0-201-39829-X.

[6] Roberto Basili, Marco Cammisa, and Alessandro Moschitti. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 1–8. Association for Computational Linguistics, June 2005.

[7] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 1993.

[8] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 624–638. Springer, 2004.

[9] Nigel Collier, Hyun Seok Park, Norihiro Ogata, Yuka Tateisi, Chikashi Nobata, Takeshi Sekimizu, Hisao Imai, and Jun'ichi Tsujii. The genia

project: corpus-based knowledge acquisition and information extraction from genome research papers. In Henry S. Thompson and Alex Lascarides, editors, *Proceedings of the European Association for Computational Linguistics*, pages 271–272. Association for Computational Linguistics, 1999.

[10] Michael Collins. Discriminative reranking for natural language parsing. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 175–182, San Francisco, CA, 2000. Morgan Kaufmann.

[11] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[12] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[13] Michael Eisenstein. A look back: putting differences aside. *Nature Methods*, 3(4):324, 2006.

[14] Tom Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Labs, Palo Alto, California, 2003.

[15] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal Machine Learning Research*, 4:933–969, 2003.

[16] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. *Künstliche Intelligenz*, 19(1):60–61, 2005.

[17] Filip Ginter. *Information Extraction in the Biomedical Domain: Methods and Resources*. PhD thesis, Turku Centre for Computer Science (TUCS), 2007.

[18] Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. New techniques for disambiguation in natural language and their application to biological text. *Journal of Machine Learning Research*, 5:605–621, 2004.

[19] Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 403–410. Association for Computational Linguistics, June 2005.

[20] Andrew R. Golding and Dan Roth. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999.

[21] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[22] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.

[23] Vasileios Hatzivassiloglou, Pablo A. Duboué, and Andrey Rzhetsky. Disambiguating proteins, genes and rna in text: a machine learning approach. *Bioinformatics*, 17(1):97–106, 2001.

[24] Ralf Herbrich. *Learning kernel classifiers: theory and algorithms*. MIT Press, 2002.

[25] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *ICANN99*, pages 97–102, London, 1999. Institute of Electrical Engineers.

[26] Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. In Arie Hasman, Reinhold Haux, Johan van der Lei, Etienne De Clercq, and Francis Roger-France, editors, *Ubiquity: Technologies for Better Health in Aging Societies. Proceedings of MIE2006. The 20th International Conference of the European federation for Medical Informatics*, Studies in Health Technology and Informatics, pages 789–794, Maastricht, Netherlands, 2006. IOS Press.

[27] Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. *International Journal of Medical Informatics*, 76S3:S362–S368, 2007.

[28] Roger Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

[29] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1):1–40, 1998.

[30] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[31] George H. John and Pat Langley. Estimating continuous distributions in Bayesian classifiers. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann Publishers.

[32] Maurice G. Kendall. *Rank Correlation Methods.* Griffin, London, 4. edition, 1970.

[33] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at JNLPBA. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75, August 28–29 2004.

[34] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Chris Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143. Morgan Kaufmann, 1995.

[35] Peng Liang and Arthur B. Pardee. Differential display of eukaryotic messenger rna by means of the polymerase chain reaction. *Science*, 257 (5072):967–971, August 1992.

[36] Charles X. Ling, Jin Huang, and Harry Zhang. Auc: a statistically consistent and more discriminating measure than accuracy. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 519–526. Morgan Kaufmann, 2003.

[37] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007.

[38] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, 1999.

[39] Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. The senseval-3 english lexical sample task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[40] Tom M. Mitchell. Does machine learning really work? *AI Magazine*, 18(3):11–20, 1997.

[41] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 2006.

[42] Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Semantic role labeling via tree kernel joint inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 61–68. Association for Computational Linguistics, June 2006.

[43] Tapio Pahikkala, Antti Airola, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Efficient auc maximization with regularized least-squares. In *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*. IOS Press, 2008. To appear.

[44] Tapio Pahikkala, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Incorporating external information in Bayesian classifiers via linear feature transformations. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Proceedings of the 5th International Conference on NLP (FinTAL 2006)*, volume 4139 of *Lecture Notes in Computer Science*, pages 399–410, Heidelberg, Germany, 2006. Springer.

[45] Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Improving the performance of Bayesian and support vector classifiers in word sense disambiguation using positional information. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 90–97, Espoo, Finland, 2005. Otamedia.

[46] Tapio Pahikkala, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Transductive ranking via pairwise regularized least-squares. In Paolo Frasconi, Kristian Kersting, and Koji Tsuda, editors, *Workshop on Mining and Learning with Graphs (MLG'07)*, pages 175–178, 2007.

[47] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. Learning to rank with pairwise regularized least-squares. In Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

[48] Tapio Pahikkala, Evgeni Tsivtsivadze, Jorma Boberg, and Tapio Salakoski. Graph kernels versus graph representations: a case study in parse ranking. In Thomas Gärtner, Gemma C. Garriga, and Thorsten Meinl, editors, *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG'06)*, pages 181–188, 2006.

[49] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[50] Tomaso Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19:201–209, 1975.

[51] Tomaso Poggio and Steve Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50 (5):537–544, 2003.

[52] Massimiliano Pontil. Learning in reproducing kernel hilbert spaces: a guide tour. *Bull. of the Italian Artificial Intelligence Association – AI*IA Notizie*, 2003.

[53] Foster Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In Jude W. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, San Francisco, CA, 1998.

[54] Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50, 2007.

[55] Sampo Pyysalo, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Evaluation of two dependency parsers on biomedical corpus targeted at protein-protein interactions. *Recent Advances in Natural Language Processing for Biomedical Applications, special issue of the International Journal of Medical Informatics*, 75(6): 430–442, 2006.

[56] Sampo Pyysalo, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, Tapio Salakoski, and Jeppe Koivula. Analysis of link grammar on biomedical dependency corpus targeted at protein-protein interactions. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *Proceedings of the JNLPBA workshop at COLING'04, Geneva*, pages 15–21, 2004.

[57] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning.* PhD thesis, MIT, 2002.

[58] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[59] Tony G. Rose, Mark Stevenson, and Miles Whitehead. The Reuters Corpus Volume 1: From yesterday's news to tomorrow's language resources. In Manuel Gonzales Rodriguez and Carmen Paz Suarez Araujo, editors, *Proceedings of the Third International Conference on Language Resources and Evaluation.* ELRA, Paris, France, 2002.

[60] Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. *Annals of Statistics*, 35(3):1012–1030, 2007.

[61] Juho Rousu and John Shawe-Taylor. Efficient computation of gapped substring kernels on large alphabets. *Journal Machine Learning Research*, 6:1323–1344, 2005.

[62] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In D. Helmbold and R. Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426, Berlin, Germany, 2001. Springer.

[63] Bernhard Schölkopf, Patrice Simard, Alex Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640–646. The MIT Press, Cambridge, Massachusetts, 1998.

[64] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels.* MIT Press, Cambridge, MA, 2002.

[65] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, Cambridge, 2004.

[66] Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1991.

[67] Hanna Suominen, Tapio Pahikkala, Marketta Hiissa, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Relevance ranking of intensive care nursing narratives. In Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, 10th International Conference, KES 2006, Part I*, Lecture Notes in Computer Science, pages 720–727, Heidelberg, 2006. Springer.

[68] Evgeni Tsivtsivadze, Tapio Pahikkala, Antti Airola, Jorma Boberg, and Tapio Salakoski. A sparse regularized least-squares preference learning algorithm. In *Proceedings of the The 10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*. IOS Press, 2008. To appear.

[69] Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality-convolution kernel and its application to dependency parse ranking. In Moonis Ali and Richard Dapoigny, editors, *Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2006)*, volume 4031 of *Lecture Notes in Computer Science*, pages 610–618, Heidelberg, 2006. Springer.

[70] Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality kernels for sequential data and their applications to parse ranking. *Applied Intelligence*, 2008. To appear.

[71] Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Regularized least-squares for parse ranking. In A. Fazel Famili, Joost N. Kok, José Manuel Peña, Arno Siebes, and A. J. Feelders, editors, *Proceedings of the 6th International Symposium on Intelligent Data Analysis*, volume 3646 of *Lecture Notes in Computer Science*, pages 464–474, Heidelberg, Germany, September 2005. Springer.

[72] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[73] Grace Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

Publication Reprints

Paper I

Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation

Tapio Pahikkala, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. *BMC Bioinformatics*, 6(1):157, 2005.

BMC Bioinformatics

Research article

Open Access

Contextual weighting for Support Vector Machines in literature mining: an application to gene versus protein name disambiguation

Tapio Pahikkala*, Filip Ginter, Jorma Boberg, Jouni Järvinen and Tapio Salakoski

Address: Department of Information Technology, University of Turku and Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, 20520 Turku, Finland

Email: Tapio Pahikkala* - tapio.pahikkala@it.utu.fi; Filip Ginter - filip.ginter@it.utu.fi; Jorma Boberg - jorma.boberg@it.utu.fi; Jouni Järvinen - jouni.jarvinen@it.utu.fi; Tapio Salakoski - tapio.salakoski@it.utu.fi

* Corresponding author

Published: 22 June 2005

BMC Bioinformatics 2005, **6**:157 doi:10.1186/1471-2105-6-157

Received: 12 July 2004
Accepted: 22 June 2005

This article is available from: http://www.biomedcentral.com/1471-2105/6/157

Abstract

Background: The ability to distinguish between genes and proteins is essential for understanding biological text. Support Vector Machines (SVMs) have been proven to be very efficient in general data mining tasks. We explore their capability for the gene versus protein name disambiguation task.

Results: We incorporated into the conventional SVM a weighting scheme based on distances of context words from the word to be disambiguated. This weighting scheme increased the performance of SVMs by five percentage points giving performance better than 85% as measured by the area under ROC curve and outperformed the Weighted Additive Classifier, which also incorporates the weighting, and the Naive Bayes classifier.

Conclusion: We show that the performance of SVMs can be improved by the proposed weighting scheme. Furthermore, our results suggest that in this study the increase of the classification performance due to the weighting is greater than that obtained by selecting the underlying classifier or the kernel part of the SVM.

Background

The amount of scientific biomedical literature readable by computer programs is overwhelming. For example, PubMed [1] contains about 7.5 million article abstracts. Therefore automatic literature-mining methods can be exploited in order to retrieve relevant information (for recent thorough reviews of related work in Bio-NLP, see e.g. [2,3]). For example, several algorithms have been developed for extracting information about protein-protein interactions from the biomedical literature [4-10].

The problem

In order to find the relations between biological or chemical entities, first the names of the entities have to be recognized in a reliable way. There has been a significant amount of effort to do that automatically [11-20]. A large standardized domain corpus helps to consolidate the research efforts. The GENIA corpus [21] has been commonly used in biomedical named entity recognition. The state-of-the-art systems have recently been compared, for example, in Kim et al. [22] using the GENIA corpus.

The task of named entity recognition can be divided in two subtasks, the identification of entities, that is, determining the boundaries of the named entities, and their classification into proper classes. The problem of finding the class the entity belongs to can be treated as a word sense disambiguation (WSD) task, which, on its own, is an essential part of natural language processing (see e.g. Manning and Schütze [23] for more information).

The entities in biomedical text are highly ambiguous. For example, it is common that a gene has the same name as the protein it codes for. In the following three sentences from the GENIA corpus, the occurrences of BZLF1 are a protein, a gene, and an RNA, respectively: (1) *Expression of either BZLF1 or BRLF1 triggers expression of...* (2) *... DNA in lymphoblastoid cell lines induced by transfection with BZLF1*. and (3) *... lysis of certain HLA B8+ LCL targets was associated with the abundance of BZLF1 transcripts*. Similar ambiguity is illustrated in the two sentences given by Hatzivassiloglou et al. [24]: *By UV cross-linking and immunoprecipitation, we show that SBP2 specifically binds selenoprotein mRNAs both in vitro and in vivo. The SBP2 clone used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3' UTR truncated at the polyadenylation site)*. The occurrence of SBP2 is a protein in the first sentence, whereas the occurrence of SBP2 in the second sentence is a gene. In the same study, a domain corpus was annotated by three biology experts. The three experts unanimously agreed only in 78% of the cases, each name being classified as either a gene, protein or mRNA. This low rate of inter-annotator agreement suggests that the task is relatively difficult even for human experts, reflecting the inherent complexity of the domain. However, the study does not analyse more closely the reasons that lead to annotation disagreements.

In this paper, we consider the disambiguation of the sense "gene" or "protein" when the name is not disambiguated explicitly by the author with the word "gene" or "protein" (e.g. "SBP2 gene"). This task is important, because the release of the human genome and large scale functional genomics studies and methods have made it important to be able to find information from literature specifically for proteins and the corresponding genes. However, database searches provide a lot of hits among which the correct and important articles have to be sorted manually. Therefore, for example, in data mining related to proteomics the scientists could save much time if they could direct their search only to proteins.

Much of the ambiguity in biomedical text is caused by inconsistent or non-existent naming conventions. For example, there exist Drosophila gene names such as *ring* and *arc* that can be confused with their ordinary meanings. Manual analysis of a small set of abstracts returned by PubMed for the query *ring and drosophila* shows that the word *ring* appears in its gene/protein sense in about 30% of the cases, both capitalized and non-capitalized. Similarly, the word *arc* is ambiguous and appears in about 75% of the cases in its gene/protein sense, again both capitalized and non-capitalized. In both cases, only abstracts regarding Drosophila were considered, thus the two example words retain their ambiguity even in the sublanguage of articles concerning Drosophila. In contrast, some other gene/protein names, such as, *tax* do not retain their ambiguity in the sublanguage: all occurrences of *tax* in the GENIA corpus refer to its gene/protein sense. Another major source of ambiguity in scientific biomedical text are abbreviations, which are widely used and therefore are very important to be identified correctly in natural language processing applications [25].

There have already been applications of word sense disambiguation methods in the field of scientific biological text processing. Hatzivassiloglou et al. [24] disambiguated names of genes, proteins and RNAs using a Naive Bayes classifier. Previously we have developed a method named here Weighted Additive Classifier (WAC) and applied it to the problem of gene/protein name disambiguation [26]. Liu et al. [27] disambiguated abbreviations from Medline abstracts using a Naive Bayes classifier. Yu et al. [28] achieved better results for the same task using Support Vector Machines (SVMs) with the one sense per discourse hypothesis. Furthermore, a system developed by Podowski et al. [29] assigns gene names to their LocusLink IDs in previously unseen abstracts.

Lee and Ng [30] performed a comparison of several supervised learning algorithms for WSD tasks and in their study, SVMs were confirmed to have the best performance. SVMs have also been applied in biomedical WSD (see e.g. Yu et al. [28]) as well as in biomedical named entity recognition [20,31-35]. Furthermore, in the COLING-2004 JNLPBA shared task of Bio-Entity Recognition [22], five studies [36-40] used SVMs either alone or combined with other algorithms. In this paper, we apply SVMs and as baselines, we consider the Naive Bayes and WAC classifiers. The studies mentioned above use narrow context windows and focus mainly on studying different features such as orthographical, morphological, lexical, contextual, part-of-speech, head-noun, and name-alias features. We, in contrast, focus on context representations that use distance of the words from the ambiguous name in addition to the context words themselves. We evaluate the methods on the GENIA data set (see e.g. Collier et al. [21]), using the area under ROC curve (see e.g. [41]) as a performance measure.

Support Vector Machines

SVMs can be used to classify multidimensional data into two classes. SVMs were introduced by Boser et al. [42]. Thorough presentations of SVMs are given by Burges [43] and Vapnik [44], for example, and in the Methods section we give a concise introduction to SVMs. In a binary classification task, the training set consists of data points which are labeled as positive or negative. In our case, the training data points are the contexts of the ambiguous names. The positive and negative labels denote genes and proteins, respectively. In order to improve linear separability, the data points are mapped from the input space to a new feature space before they are used for training or for classification. The mapping is done implicitly by a so-called kernel function, which computes the similarity of two data points in the feature space. The choice of an appropriate kernel function is a nontrivial problem, but there are certain standard kernel functions which are frequently used. Kernels and the SVM itself also have certain parameters which have to be adjusted in order to make the SVM classifier work in the best possible way.

Contribution of this work

In short, we considered application of SVMs to the gene versus protein name disambiguation problem in abstracts of biomedical articles. While other studies focus mainly on studying different features, our work primarily considers context representations. We resolve the ambiguous names using their context which spans up to the whole abstract, in contrast to other previous applications of SVMs which typically use narrow context windows. To improve the performance of conventional SVMs and accommodate the wide context span, we adopted a weighting scheme introduced by Ginter et al. [26] that exploits the information about the distances of the words from the name to be disambiguated, and adjusted the scheme for the SVM classifier. We carefully searched for the best parameter values of SVMs and kernel functions using grid optimization as suggested by Hsu et al. [45], and we also performed a similar search for the parameters of the proposed weighting scheme. Finally, we measured the performance of both conventional and weighted SVMs together with two baseline methods, and showed that the performance improvement was statistically significant.

Results and discussion

We experimented with the protein versus gene name disambiguation problem using conventional SVMs with linear, Gaussian, as well as second and third degree polynomial kernels. Also, we tested SVMs using different kernels augmented with the proposed weighting scheme. As additional baseline classifiers, we used the Weighted Additive Classifier, which also uses contextual weighting, and the Naive Bayes classifier. These methods and their parameters to which we refer in this section are described in detail in the Methods section.

In the following, we first discuss the weighting scheme and the reasons why its use is beneficial. Then, we present how the data was generated and preprocessed. Finally, we present the performance measure used in the experiments, describe the experimental setting and the results of the parameter estimation and the final validation.

Contextual weighting

The training data points are vectors of word frequencies in the context in which the names to be disambiguated were found. The basic SVMs with any kernel use only the word frequencies and do not take into consideration the distances of the words with respect to the position of the name to be disambiguated. However, the distance information seems intuitively to be important, and therefore we apply a weighting scheme that incorporates this information into the context representation used by SVMs. The weighting scheme models the distances of the words from the ambiguous name, while the information whether the words are before or after the ambiguous name is not considered. The weight of a context word at the distance d is given by $d^{-\lambda} + \beta$, where the parameters λ and β are used to control the effect of the distances of the words from the name to be disambiguated. A more detailed explanation of the weighting scheme is presented in the Methods section.

We now discuss some possible reasons for the weighting scheme achieving a statistically significant gain in classification performance. Yarowsky [46] argues that the effect of context words is strongest for immediately adjacent words, and weakens with distance. This phenomenon is called the *one-sense-per-collocation* principle. Yarowsky also considers the *one-sense-per-discourse* principle, that is, all instances of an ambiguous word tend to have the same sense within one discourse unit, the article abstract in our case. In that case also distant words can help in disambiguation. One-sense-per-discourse is, however, presumed to be a weaker hypotheses, which should be overridden when the local evidence is strong. In order to study the tenability of these hypotheses in our data, we estimated the following conditional probabilities of the name to be disambiguated to have another instance of an ambiguous name in its context. For each distance from the name to be disambiguated, we estimated the conditional probability that there is a word in the context at that distance and the word is another ambiguous name with the same sense, as well as the conditional probability that the word is a name with the opposite sense. These probabilities are illustrated in Figure 1, where the solid line denotes the probability of an occurrence of a name with the same sense and the dashed line denotes the probability of the

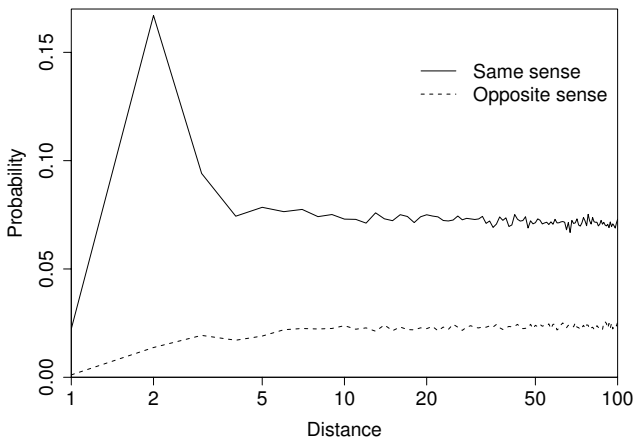


Figure 1
Conditional probabilities of having other instances of ambiguous names in the context of the name to be disambiguated. The X-axis denotes the distance in words from the name to be disambiguated and Y-axis is the probability. The solid line denotes the probability that if there is a word in the context at the given distance, the word would be another ambiguous name having the same sense with the name to be disambiguated and the dashed line denotes the corresponding probability for the opposite sense.

opposite sense. At close distances (<6), the probability of the same sense turned out to be high and decreasing with distance, whereas the probability of the other sense behaved in the opposite way. In the proposed method, the words in the area of influence of the one-sense-per-collocation principle, that is, the words at close distances, have more weight than the long distance words and these weights are controlled by the parameter λ. On the other hand, at long distances, the probabilities of the same and the opposite senses settled down to 0.08 and 0.02, respectively, indicating that mostly the one-sense-per-discourse principle holds. Therefore, when the close context is unable to make a strong decision, the information of the long distance words may be useful. This effect is controlled by the β parameter, which balances the influence of the one-sense-per-discourse principle compared to the one-sense-per-collocation principle. Note that one-sense-per-discourse does not have to hold strictly, because the information can be useful if there are on average more instances of the names with the same sense than with the opposite sense in the far context. Both near and far context words are important when deciding the sense of a name. For example, verbs like "activate" or "phosphorylate" are often found around protein names, whereas verbs like "express" or "transcribe" may be found around gene names. Similarly, head nouns, such as expression, are also highly indicative of the sense. These words may be located near to the name to be disambiguated, being strong indicators of its sense. As shown above, ambiguous names in the abstract are more likely to be of the same sense and therefore the words around the other ambiguous names are partly indicative about the sense of the name to be disambiguated. Since other ambiguous names can occur at any position of the abstract, it is beneficial to use long context. Descriptions of experimental conditions can indicate one sense common to all of the names in the abstract. For example, "yeast two-hybrid" indicates protein-protein interaction finding, while "microarray" relates to gene experiments. Further, the occurrence of a distant coreference, for example, between the full form of an ambiguous name and its abbreviation, in the context of the ambiguous name may provide distant words indicative of the correct sense.

The phenomena discussed above are highly data dependent. However, the proposed weighting scheme models them if the weighting scheme is equipped with the optimal values of the parameters λ and β found in the training phase.

Data and its preprocessing

The data set considered in this paper is constructed as follows. We obtained the evaluation data set for the COLING-2004 JNLPBA shared task of Bio-Entity Recognition [22], which is derived from the GENIA corpus [21], a standard corpus for biomedical named entity recognition, by conflating the original 36 classes into five classes (DNA, RNA, protein, cell line and cell type) of which we only use two classes, namely, DNA and protein. The data set consists of 2000 hand-annotated abstracts and contains 30269 protein examples and 9533 DNA examples. Average number of words in the abstracts is 246.

Naturally, not all names are truly ambiguous in all domains and corpora. In these cases a classifier could gain by simply memorizing the names. We made an experiment in which we used only the ambiguous gene and protein names as data points and the performance obtained was over 95%. Using context words as extra features improved the performance only by 0.3% percentage points, with optimal context span 1 found experimentally. The difference was not statistically significant. To assess the performance in a more general setting and to avoid the overfitting effect of memorizing, we do not include the instance of the term to be disambiguated into its context, as the purpose of this paper is to study context-based name disambiguation. Note also that memorizing the names does not help the classifier to disambiguate names that do not exist in training data, for example, names introduced only recently.

The text was further preprocessed by removing stop words and stemming the words with the Porter stemming algorithm [47] (stemming and stop-word removal are discussed in the Methods section).

Measure of performance

The number of protein examples (30269) in our corpus is about three times greater than the number of gene examples (9533). Thus, we could achieve a classification accuracy of about 75% by always predicting the protein class. To cope with the imbalance in the data, we measure the performance of each classifier as the area under ROC curve (AUC). ROC curve is a relation between the true-positive rate (TPR) and the false-positive rate (FPR) at various classification thresholds:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

where *TP*, *FN*, *FP*, and *TN* are true positives, false negatives, false positives, and true negatives, respectively. Unlike other popular measures such as accuracy and precision-recall analysis, the AUC measure is invariant to the prior class probabilities. AUC corresponds to the probability that given a randomly chosen positive example and a randomly chosen negative example, the classifier will correctly say which is which. For a thorough discussion of ROC curves and the AUC measure, see, for example, Fawcett [41], Maloof [48], and Bradley [49].

We cross-validate all AUC measurements using the 5 × 2 cross-validation scheme, which is an ordinary 2-fold cross-validation performed five times. To obtain a 2-fold cross-validated performance estimate, we randomly divide a set of abstracts into two equally-sized sets and average the two performance measurements obtained by training the classifier on one set and testing the classifier on the other set. To obtain a 5 × 2 cross-validated performance estimate, a 2-fold cross-validation is performed five times and the estimates are then averaged. To avoid indirect overlap between test and training sets, we form the sets so that examples originating in the same abstract always remain in the same set.

To test for statistical significance, we use the robust 5 × 2-cv test [50]. The test avoids the problem of dependence between folds in *N*-fold cross-validation schemes and results in a more realistic estimate than, for example, the *t*-test.

Experimental setup

We randomly divided the preprocessed set of 2000 abstracts into two equal-sized sets; 1000 abstracts for parameter estimation and 1000 abstracts for final validation of the methods. The 5 × 2 cross-validated AUC was used as the measure of performance in both parameter estimation and final validation. In all the experiments with SVMs, we normalized the word frequency vectors to unit length, because the sizes of the contexts varied considerably. We carried out the SVM experiments using the LIBSVM 2.6 software [51] and the Naive Bayes experiments using the Bow toolkit [52].

In the experiments, optimal parameter values for SVMs with different kernels and for the proposed weighting scheme must be searched (the exact definitions of the parameters and kernel functions are presented in the Methods section). Every SVM itself has always a penalty parameter *C*, linear kernel has no other parameters than *C*, and both Gaussian and polynomial kernels have an additional parameter λ. Adopting a contextual representation yields a context span parameter *s*. The SVM equipped with the weighting scheme has two additional parameters λ and β by which we may control the effect of the distances of the words from the name to be disambiguated when weighting the context words. The performance of a classifier may be strongly influenced by the choice of the values for its parameters. For example, from Figure 3 (discussed in more detail later) it can be observed that a wrong choice of the kernel parameters as well as the SVM penalty parameter *C* can lead to a severe loss in performance. Particularly when comparing the methods, the correct parameter setting for each of the compared methods is crucial, as only then a reliable estimate of the performance is obtained for each of the methods. The correct parameter values cannot be known in advance and the use of the default values may result in sub-optimal classification performance. Therefore, the parameter values are most commonly estimated from the data. Hsu et al. [45] recommend a grid-search on *C* and γ parameters using cross-validation and exponentially growing sequences of *C* and γ. Since an exhaustive search for the parameters can not be done in the continuous space of SVM and kernel parameters, we performed a coarse preliminary search in order to find an auspicious region, and subsequently conducted a finer grid search. As can be observed from Figure 2, the choice of values of the weighting parameters λ and β is as important for the classification performance as the choice of the other parameters. Moreover, as noticed by Ginter et al. [26], the optimal values depend on the task and are not known beforehand. Therefore, we use a grid-search also for finding the optimal values of the parameters λ and β.

In short, the parameter estimation for SVM classifiers was performed as follows. First we estimated the context span *s* for the conventional SVM and the values of λ and β for the weighted SVM, using a grid search with the linear

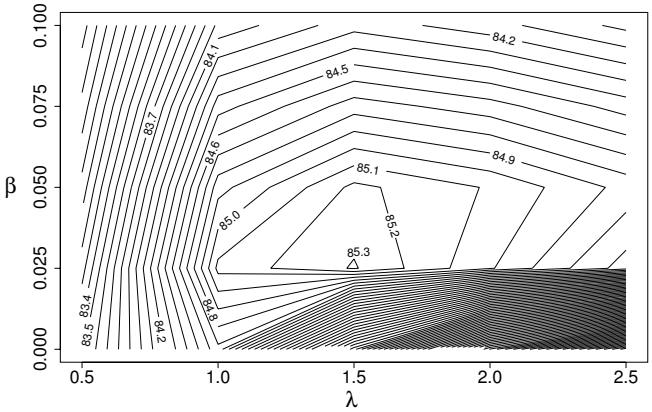


Figure 2
Parameter estimation: The performance of the weighted SVM with the linear kernel as a function of the weighting parameters λ and β. The best performance is reached at λ = 1.5 and β = 0.025.

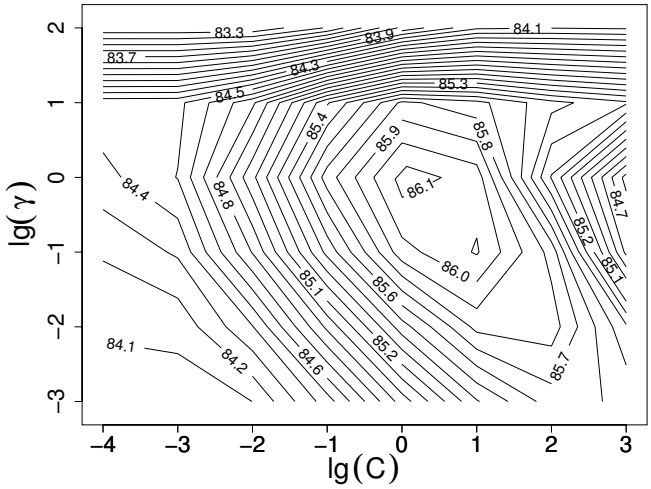


Figure 3
Parameter estimation: The performance of the weighted SVM with the Gaussian kernel as a function of the SVM penalty parameter *C* and the kernel parameter γ. Both parameters are in a logarithmic scale. The best performance is reached at C = 1 and γ = 1.

kernel function. We used the whole abstract to form the examples for the weighted SVM. With the s, λ and β parameters fixed, we evaluated different types of SVM kernels, estimating the kernel parameters with a grid search. The SVM penalty parameter C is optimized separately at each point of the context span, weighting and kernel parameter grids.

Table 1: Parameter estimation: The performance of conventional SVMs with different kernel functions, context span *s* = 60.

Kernel	AUC	Parameters
Linear	80.47%	$C = 2^{-3}$
Gaussian	80.62%	$C = 2^{-2}$, $\gamma = 2$
Polynomial (d = 2)	80.49%	$C = 2^{-9}$, $\gamma = 8$
Polynomial (d = 3)	80.53%	$C = 2^{-7}$, $\gamma = 2$

In a similar manner, the optimal combination of λ, β and s for the WAC classifier was found by performing a 3-dimensional grid search. For the Naive Bayes classifier, only the optimal value for s must be searched. For final validation, we chose the best performing kernel function for conventional and weighted SVMs. Using the parameter values found in the parameter estimation phase, we then measured the performance of each of the compared classifiers on the validation set, again using 5 × 2 cross-validation.

A detailed explanation of the grid search for the parameter estimation described above is presented in the following sections.

Parameter estimation for conventional SVM

First, we searched for the optimal context span s that we will use in our experiments with the conventional SVM. We experimented with different context spans using the conventional SVM with the linear kernel, namely spans of 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 words to both directions from the term to be disambiguated. The parameter C for the SVM with the linear kernel was searched with values 2^{-5}, 2^{-4},..., 2^{3} for each of the different context spans. In these experiments, the context span of 60 words to both directions from the term to be disambiguated resulted in the highest performance for the conventional SVM. We used this context span when experimenting with Gaussian and polynomial (d = 2, d = 3) kernels, because simultaneous searching for optimal context span and kernel parameters C and γ for the Gaussian kernel and polynomial kernels with degrees d = 2 and d = 3 would have been computationally impractical. The values of the C and γ parameters of the Gaussian kernel were 2^{-5}, 2^{-4},..., 2^{2} and 2^{-3}, 2^{-2},..., 2^{2}, respectively, and the values for the C and γ parameters of the polynomial kernels were 2^{-10}, 2^{-9},..., 2^{-2} and 2^{-2}, 2^{-1},..., 2^{5}, respectively. The results obtained with different kernel functions using the optimal context span s = 60 found with the linear kernel are shown in Table 1.

Parameter estimation for weighted SVM

With the weighted SVM, we always used the whole abstract as a context. For the weighted SVM, we estimated

Table 2: Parameter estimation: The performance of weighted SVMs with different kernel functions, λ = 1.5 and β = 0.025.

Kernel	AUC	Parameters
Linear	85.31%	C = 1
Gaussian	86.15%	C = 1, γ = 1
Polynomial (d = 2)	86.09%	C = 2^{-3}, γ = 2
Polynomial (d = 3)	86.13%	C = 2^{-3}, γ = 1

the best combination of the weighting parameters λ and β with the linear kernel. The parameter C was also separately searched with values 2^{-5}, 2^{-4},..., 2^{3} for each of the different weightings. The comparison of the performance with different weightings using the linear kernel is illustrated in Figure 2. At this point, we found that the values λ = 1.5 and β = 0.025 performed best for the linear kernel (for illustration, see Figure 6). We used these parameters when experimenting with Gaussian and polynomial (d = 2, d = 3) kernels. The values of the C and γ parameters of the Gaussian kernel were 2^{-4}, 2^{-3},..., $2^{3}$and 2^{-3}, 2^{-2},..., 2^{2}, respectively, and the values for the C and γ parameters of the polynomial kernels were 2^{-7}, 2^{-6},..., 2^{-2} and 2^{-2}, 2^{-1},..., 2^{2}, respectively. The performance of the weighted SVM with different C and parameters of the Gaussian kernel is illustrated in Figure 3. The figure illustrates the importance of correct parameter selection. The weighted SVM performance with different combinations of γ and the penalty parameter C follows the behavior described by Keerthi and Lin [53]: Areas of underfitting can be seen at the left, where the value of the C parameter is low, and at bottom left where the values of both C and γ are low. On the other hand, SVM with Gaussian kernel overfits heavily if the value of γ is too large, as can be seen at the top of the figure. Overfitting happens also with noisy data at the right part of the figure, where the value of C is too large. The results obtained with different kernel functions using the best weighting parameters λ = 1.5 and β = 0.025 found with the linear kernel are shown in Table 2.

Parameter estimation for baseline methods

The only parameter of the Naive Bayes classifier is the context span s. We performed a search for $s \in [5, 30]$ with step 5. The performance reached maximum for s = 15. The WAC incorporates an identical weighting scheme as the weighted SVM. We found the optimal parameters by performing a 3-dimensional grid search for $\lambda \in [0,3]$ with step 0.1, β || [0, 0.25] with step 0.025 and context span in the interval [5,70] with step 5. The maximum performance was obtained for λ = 1, β = 0.025 and context span s = 55 (for illustration, see Figure 4).

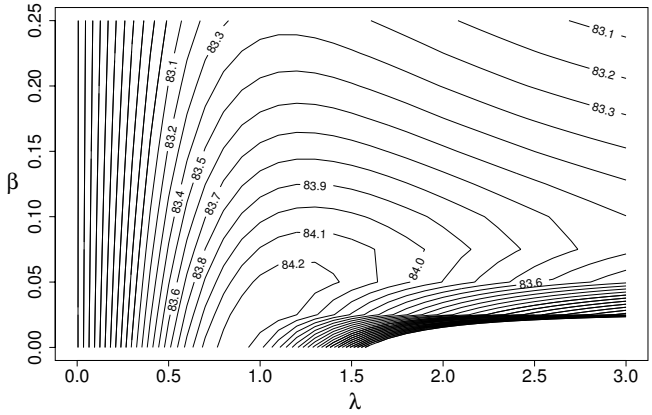


Figure 4
Parameter estimation: The performance of WAC with s = 55 as a function of the weighting parameters λ and β. The best performance is reached at λ = 1.0 and β = 0.025.

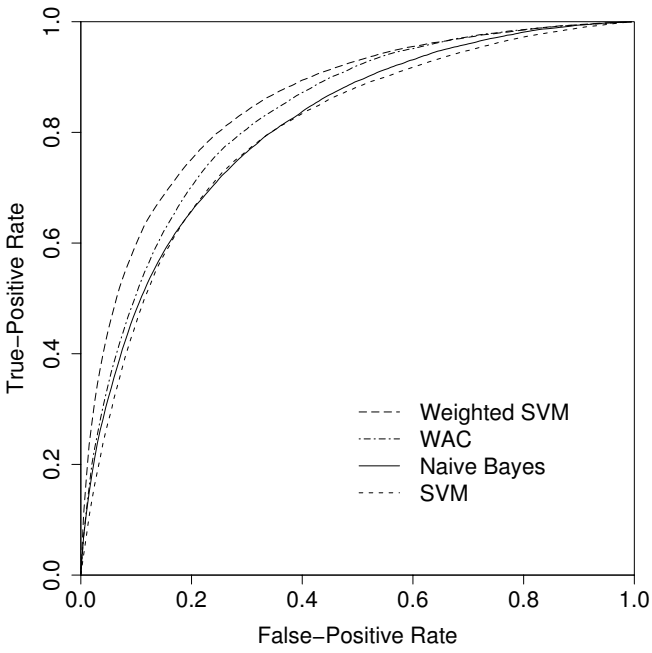


Figure 5
Final validation: Averaged ROC curves of the classifiers. The averaged ROC curves were obtained from the folds of the 5 × 2 cross-validation using the vertical averaging method described by Fawcett [41].

Final validation

The Gaussian kernel was found to be the best with both the conventional and weighted SVMs when tested in the

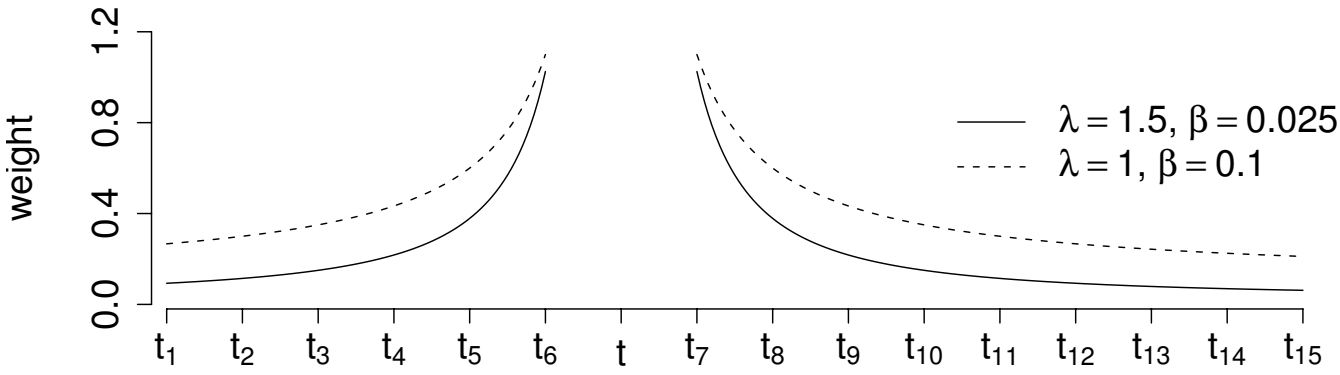


Figure 6

A weighting example. Here is an example of a context (t_1,..., t_{16}). The term t is the name to be disambiguated. Figure illustrates the weights of the context words t_1,..., t_{16} with two different parameter value pairs of λ and β. The weight values are represented as a continuous function, although they take discrete values. The parameter combination λ = 1.5 and β = 0.025 yields the best performance when using the linear kernel.

Table 3: Final validation: The performance of the classifiers.

Method	AUC
Conventional SVM	79.85%
Weighted SVM	85.48%
WAC	83.05%
Naive Bayes	80.81%

parameter estimation. The best parameters for the Gaussian kernel were C = 0.25 and λ = 2 with the conventional SVM, and C = 1 and λ = 1 with the weighted SVM. The AUC results of the final validation are presented in Table 3 and the ROC curves are given in Figure 5. To test the statistical significance of AUC differences between the weighted SVM, the conventional SVM, WAC and Naive Bayes, we performed the robust 5 × 2-cv test on the validation data. Each of the pairwise differences were strongly significant with p-values below 0.01, except for the difference between the conventional SVM and the Naive Bayes classifier (p-value of about 0.1). The conventional SVMs performed poorly compared to the baselines, especially to the WAC classifier that takes advantage of the contextual weighting. Incorporation of the weighting scheme into SVMs, however, improved their performance by five percentage points.

Conclusion

In this paper, we show that SVMs can be successfully applied to gene versus protein name disambiguation. We demonstrate how their performance can be further improved by incorporating a weighting scheme based on the intuition that the words near the name to be disambiguated are more important than the other words. The weighting scheme results in a notable performance gain of five percentage points. We also study carefully the effects of different kernel functions and parameters and show that the proposed weighting scheme influences the performance even more than the selection of the kernel part of SVMs. The weighted methods statistically significantly outperformed their unweighted counterparts, the difference being particularly notable for SVMs.

In Ginter et al. [26], we have shown that the optimal values for λ and β are non-zero and differ substantially depending on the classification task at hand. This suggests that the extent to which the long distance words contribute to the classification is task-dependent and could reflect differing properties of the tasks. While finding correct values of these parameters is clearly important as shown by the experiments, an exact interpretation of the values remains speculative. However, we discuss several reasons why the use of the proposed weighting scheme is beneficial. Further study could bring a better insight into the underlying phenomena.

The performance of the weighted SVM might be further improved, for example, by using collocations in order to capture the local syntax around the term to be disambiguated. However, the proposed weighting scheme uses the local information, and therefore it already captures the information represented by collocations to some extent. In addition, several special text kernels have successfully been applied to text classification as reported by Lodhi et al. [54] and by Cancedda et al. [55]. These kernels and different weighting methods based on the distances

and also, for example, on the biological relevance of the words in the context, are still to be studied.

Methods

In this section, we describe the concepts necessary to understand the application of a SVM classifier to gene versus protein name disambiguation. We start by giving a short introduction to SVMs. The training data points of the classifier are vectors describing the word frequencies in the context in which the names to be disambiguated were found. Then, we explain the general bag of words (BoW) approach that uses only the word frequencies. However, the distances of the words with respect to the word to be disambiguated seem intuitively to be important. We describe a weighting scheme (the weighted BoW) based on distances of the context words from the ambiguous name. Finally, we briefly introduce the two baseline methods, the Naive Bayes classifier and the Weighted Additive Classifier.

Support Vector Machines

Here, we give a brief description of SVMs. A more comprehensive treatment can be found, for example, in Burges [43] and Vapnik [44]. In a binary classification task m labeled examples $(x_1, y_1),..., (x_m, y_m)$, where $x_i \varepsilon X$ are training data points and $y_i \varepsilon \{-1, +1\}$ are the corresponding class labels, form the training set. In order to make the data linearly separable, data points are mapped from the input space X to a feature space F with a mapping

$$\Phi : X \to F$$

before they are used for training or for classification.

SVMs can be considered as a special case of the following regularization problem:

$$\min_f C\sum_i l(f(x_i), y_i) + \frac{1}{2} \| f \|_k^2, \qquad (1)$$

where i ranges from 1 to m, l is the loss function used by the learning machine, $f : X \to Y$ is a function which maps the input vectors $x \in X$ to the output labels $y \in Y$, $C \in \Re_+$ is a regularization/penalty parameter, and $|| \cdot ||_k$ is a norm in a Reproducing Kernel Hilbert Space defined by a positive definite kernelfunction k. The second term is called a regularizer. The loss function used by SVMs for binary classification problems is called linear soft margin loss or hinge loss and is defined as

$$l(f(x), y) = \max(1 - yf(x), 0).$$

By the Representer Theorem, the minimizer of (1) has the following form:

$$f(x) = \sum_i \alpha_i k(x, x_i),$$

where $a_i \varepsilon \Re$ and k is the kernel function associated with the Reproducing Kernel Hilbert Space mentioned above.

The penalty parameter C controls the trade-o3 between the complexity of the decision function and the number of wrongly classified training points the model will tolerate in the feature space. Minimizing number of training errors by selecting an appropriate parameter can sometimes lead to overfitting due outliers. On the other hand, too strong regularization (low penalization) underfits. A good insight of trade-o3 can be found, for example, in Hastie et al. [56].

There are several commonly used kernels (see Vapnik [44]). The ordinary inner product is called the *linear kernel*

$$k(u, v) = \langle u, v \rangle$$

and the *polynomial kernel* is defined as

$$k(u, v) = (\gamma \langle u, v \rangle + 1)^d$$

where $d \varepsilon \mathbb{N}$ is the degree of the polynomial and $\gamma \varepsilon \Re_+$. When the polynomial kernel is used, the datapoints are mapped into a feature space which contains all products of input vector elements up to d (see e.g. Vapnik [57]). The γ parameter of the polynomial kernel controls the weight differences of the product features of different orders. Another widely used kernel function is the *Gaussian kernel*

$$k(u, v) = e^{-\gamma \|u-v\|^2},$$

whose width is determined by the γ parameter. We refer to Keerthi and Lin [53] for more information of the behavior of the SVM with the Gaussian kernel with different combinations of and the penalty parameter C. Hsu et al. [45] suggested to use a cross-validation and a grid search in order to estimate the best combination of and the penalty parameter C for the Gaussian kernel. We adopt this procedure and also perform a similar search for the linear and polynomial kernels. A detailed description of the parameter selection is given in the Results and Discussion section.

Representation of contexts

In our experiments, we trained the SVM classifier to disambiguate the sense of a term between two possible senses based on its context. Let us denote by s the *context span* parameter controlling the lengths of the contexts. For a fixed s, we take such a context $\bar{t} = (t_1,..., t_l)$ that both the number of words preceding and following t is maximal

but at most s. The words which precede t are $t_1,\ldots, t_k$, $0 \le k \le s$, in the order they appear in the text, and correspondingly $t_{k+1},\ldots, t_l$, $0 \le l - k \le s$ are the words which follow t in the text.

Hence, if there exist s words preceding and following the word to be disambiguated, then $k = l - k = s$. The contexts may be of different lengths, since the number of words from t to the beginning or end of the abstract may be smaller than s.

The BoW approach

Let C be a set of all possible contexts and let $V = \{v_1,\ldots, v_n\}$ be an ordered set of all distinct words of the contexts of C. We formed the set V of all distinct words separately for each training-testing experiment from the words found in the contexts of abstracts of the training set. Let be the mapping, which maps contexts to BoW vectors, defined by

$$\Psi : C \to \mathbb{N}^n, \bar{t} \mapsto (\psi_1(\bar{t}),\ldots,\psi_n(\bar{t})), \qquad (2)$$

where $\psi_i(\bar{t})$, $1 \le i \le n$, is the number of occurrences of the word $v_i \in V$ in the context $\bar{t}$. Thus, only the frequency of a word in the context is recorded, but the information about the distances of the words from the ambiguous name is ignored. The BoW vectors can now be used as input space data points for SVM classifiers.

The number of words in V is usually large, and therefore the dimension of the input space is also high. Customary means to reduce the dimensionality are, for example, stop-word removal and stemming. Stop-words are words that occur very often in all documents, for instance, 'is', 'the', 'are', 'a'. Stemming combines words that only differ in suffix. For example, the stemming algorithm of Porter [47] removes all suffixes it recognizes. We have applied these dimensionality reduction techniques in our main experiments. However, the number of words still often remains in tens of thousands.

In a separate experiment, we have estimated the effect of stop word removal and stemming using theconventional SVM with the linear kernel and grid search optimization of C and context span parameters. We found that stemming results in a low increase in performance (0.59%), stop-words removal haspractically no effect on the performance (an increase of 0.01%). Neither of the differences were statistically significant.

The weighted BoW approach

The BoW approach does not preserve any information about the positions of the words in the context. Therefore, we use a particular weighting scheme based on the distances of the words from the term t to be disambiguated. The idea is that the words near t are more likely to be important than other words, and therefore they are given larger weights.

The weighted vector space model of contexts can be formalized as follows. Let dist(j) denote the distance of the word t_j from the term t, that is, the number of words between the word t_j and t including the word t_j itself. Let further $\mathrm{Pos}(v, \bar{t}) = \{j \mid v = t_j \varepsilon\ \bar{t}\}$ denote a set of positions j for each word $v \varepsilon V$ in a particular context $\bar{t}$. The weight for the word t_j is defined as

$$\frac{1}{\mathrm{dist}(j)^{\lambda}} + \beta,$$

where $\lambda, \beta \ge 0$ are the parameters for the weighting. If $\lambda > 0$, the weights get a hyperbolic shape with highest values immediately around the term to be disambiguated (see Figure 6). The bigger λ is, the steeper the weight values grow towards the term t, and β is an offset of the values. The role of β is to reduce the ratio between the weights of the words that are near to the term t and the weights which are far from t.

Let Ψ now be the function which maps contexts to weighted BoWs given by

$$\Psi : C \to \mathbb{R}^n, \bar{t} \mapsto (\psi_1(\bar{t}),\ldots,\psi_n(\bar{t})),$$

Where

$$\psi_i(\bar{t}) = \sum_{j \in \mathrm{Pos}(v_i, \bar{t})} \left(\frac{1}{\mathrm{dist}(j)^{\lambda}} + \beta \right) \qquad (3)$$

Note that the setting $\lambda = \beta = 0$ corresponds to the ordinary BoW approach (2).

Baseline methods

The two baselines, the Naive Bayes classifier and the Weighted Additive Classifier, represent a family of linear classifiers based on aggregating the class-wise co-occurrence statistics of the words in the context. The Naive Bayes classifier (see, for example, Manning and Schütze [23]) evaluates the a posteriori conditional probability of a class by computing a product of the corresponding conditional probabilities of the context words obtained from their class-wise co-occurrence statistics.

The Weighted Additive Classifier [26] considers co-occurrence statistics similar to that used in the Naive Bayes classifier. However, the decision rule is additive and incorporates a weighting scheme. The weighting can be

defined in a manner identical to that described in the section on weighted BoW.

Authors' contributions

TP carried out the experiments with SVMs and the weighting scheme as well as the sense distribution analyses. He also drafted the manuscript. FG designed and carried out all the experiments with Naive Bayes and WAC. JB, JJ and TS conceived the original design of the work, participated in the analysis and interpretation of data, and provided scientific guidance. All authors critically revised the manuscript and approved the final version.

Acknowledgements

We thank Professor Mauno Vihinen, Institute of Medical Technology, University of Tampere, Finland, for the discussions and reasonings of the biological importance of the gene versus protein name disambiguation task. We also thank the anonymous reviewers for their insightful comments. This work was supported by Tekes, the National Technology Agency of Finland.

References

1. **Pubmed database** [http://www.ncbi.nlm.nih.gov/PubMed/]
2. Shatkay H, Feldman R: **Mining the Biomedical Literature in the Genomic Era: An Overview.** *Journal of Computational Biology* 2003, **10:**821-855.
3. Cohen KB, Hunter L: **Natural language processing and systems biology.** In *Artificial intelligence and systems biology* Edited by: Dubitzky W, Pereira F. Kluwer Academic Publishers; 2004.
4. Blaschke C, Andrade MA, Ouzounis C, Valencia A: **Automatic Extraction of Biological Information from Scientific Text: Protein-Protein Interactions.** In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology* Edited by: Lengauer T, Schneider R, Bork P, Brutlag D, Glasgow J, Mewes HW, Zimmer R. AAAI Press; 1999:60-67.
5. Ono T, Hishigaki H, Tanigami A, Takagi T: **Automated extraction of information on protein-protein interactions from the biological literature.** *Bioinformatics* 2001, **17:**155-161.
6. Marcotte EM, Xenarios I, Eisenberg D: **Mining literature for protein-protein interactions.** *Bioinformatics* 2001, **17:**359-363.
7. Temkin JM, Gilder MR: **Extraction of protein interaction information from unstructured text using a context-free grammar.** *Bioinformatics* 2003, **19:**2046-2053.
8. Donaldson I, Martin J, de Bruijn B, Wolting C, Lay V, Tuekam B, Zhang S, Baskin B, Bader G, Michalickova K, Pawson T, Hogue C: **PreBIND and Textomy – mining the biomedical literature for protein-protein interactions using a support vector machine.** *BMC Bioinformatics* 2003, **4:**11.
9. Daraselia N, Yuryev A, Egorov S, Novichkova S, Nikitin A, Mazo I: **Extracting human protein interactions from MEDLINE using a full-sentence parser.** *Bioinformatics* 2004, **20:**604-611.
10. Ginter F, Pahikkala T, Pyysalo S, Boberg J, Järvinen J, Salakoski T: **Extracting protein-protein interaction sentences by applying rough set data analysis.** In *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science 3066* Edited by: Tsumoto H, Slowinski R, Komorowski J, Grzymala-Busse JW. Springer-Verlag; 2004:780-785.
11. Fukuda K, Tsunoda T, Tamura A, Takagi T: **Toward information extraction: Identifying protein names from biological papers.** In *Proceedings of the Pacific Symposium on Biocomputing* Edited by: Altman R, Dunker A, Hunter L, Klein T. Singapore: World Scientific Press; 1998:707-718.
12. Nobata C, Collier N, Tsujii J: **Automatic Term Identification and Classification in Biology Texts.** *Proceedings of the fifth Natural Language Processing Pacific Rim Symposium* 1999:369-374.
13. Collier N, Nobata C, Tsujii J: **Extracting the Names of Genes and Gene Products with a Hidden Markov Model.** In *Proceedings of the Eighteenth International Conference on Computational Linguistics* Association for Computational Linguistics; 2000:201-207.
14. Tanabe L, Wilbur WJ: **Tagging gene and protein names in biomedical text.** *Bioinformatics* 2002, **18:**1124-1132.
15. Yu H, Hatzivassiloglou V, Rzhetsky A, Wilbur WJ: **Automatically identifying gene/protein terms in MEDLINE abstracts.** *Journal of Biomedical Informatics* 2002, **35:**322-330.
16. Franzén K, Eriksson G, Olsson F, Asker L, Lidén P, Cöster J: **Protein Names And How To Find Them.** *International Journal of Medical Informatics* 2002, **67:**49-61.
17. Yu H, Agichtein E: **Extracting synonymous gene and protein terms from biological literature.** *Bioinformatics* 2003, **19(Suppl 1):**i340-i349.
18. Chang JT, Schütze H, Altman RB: **GAPSCORE: finding gene and protein names one word at a time.** *Bioinformatics* 2004, **20:**216-225.
19. Zhou G, Zhang J, Su J, Shen D, Tan CL: **Recognizing names in biomedical texts: a machine learning approach.** *Bioinformatics* 2004, **20:**1178-1190.
20. Lee KJ, Hwang YS, Kim S, Rim HC: **Biomedical named entity recognition using two-phase model based on SVMs.** *Journal of Biomedical Informatics* 2004, **37:**436-447.
21. Collier N, Park HS, Ogata N, Tateisi Y, Nobata C, Sekimizu T, Imai H, Tsujii J: **The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers.** In *Proceedings of the European Association for Computational Linguistics* Edited by: Thompson HS, Lascarides A. Association for Computational Linguistics; 1999:271-272.
22. Kim J, Ohta T, Tsuruoka Y, Tateisi Y, Collier N: **Introduction to the bio-entity recognition task at JNLPBA.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:70-75.
23. Manning CD, Schütze H: *Foundations of Statistical Natural Language Processing* Cambridge, Massachusetts: The MIT Press; 1999.
24. Hatzivassiloglou V, Duboué AP, Rzhetsky A: **Disambiguating proteins, genes and RNA in text: a machine learning approach.** *Bioinformatics* 2001, **17:**97-106.
25. Liu H, Aronson A, Friedman C: **A Study of Abbreviations in MEDLINE Abstracts.** In *Proceedings of the 2002 AMIA Annual Symposium* Edited by: Kohane IS. Hanley and Belfus; 2002:464-468.
26. Ginter F, Boberg J, Järvinen J, Salakoski T: **New Techniques for Disambiguation in Natural Language and Their Application to Biological Text.** *Journal of Machine Learning Research* 2004, **5:**605-621.
27. Liu H, Johnson SB, Friedman C: **Automatic Resolution of Ambiguous Terms Based on Machine Learning and Conceptual Relations in the UMLS.** *Journal of the American Medical Informatics Association* 2002, **9:**621-636.
28. Yu Z, Tsuruoka Y, Tsujii J: **Automatic Resolution of Ambiguous Abbreviations in Biomedical Texts using Support Vector Machines and One Sense Per Discourse Hypothesis.** In *Proceedings of the SIGIR'03 Workshop on Text Analysis and Search for Bioinformatics* Edited by: Brown E, Hersh W, Valencia A. ACM Press; 2003:57-62.
29. Podowski RM, Cleary JG, Goncharoff NT, Amoutzias G, Hayes WS: **AZuRE, a Scalable System for Automated Term Disambiguation of Gene and Protein Names.** In *3rd International IEEE Computer Society Computational Systems Bioinformatics Conference* IEEE Computer Society; 2004:415-424.
30. Lee YK, Ng HT: **An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation.** In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* Edited by: Hajič J, Matsumoto Y. Philadelphia: Association for Computational Linguistics; 2002:41-48.
31. Kazama J, Makino T, Ohta Y, Tsujii J: **Tuning Support Vector Machines for Biomedical Named Entity Recognition.** In *ACL Workshop on Natural Language Processing in the Biomedical Domain* Association for Computational Linguistics; 2002:1-8.
32. Takeuchi K, Collier N: **Bio-Medical Entity Extraction using Support Vector Machines.** *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine* 2003:57-64.
33. Lee KJ, Hwang YS, Rim HC: **Two-Phase Biomedical NE Recognition based on SVMs.** *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine* 2003:33-40.
34. Collier N, Takeuchi K: **Comparison of character-level and part of speech features for name recognition in biomedical texts.** *Journal of Biomedical Informatics* 2004, **37:**423-435.

35. Zhou G: **Recognizing Names in Biomedical Texts using Hidden Markov Model and SVM plus Sigmoid.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:1-7.
36. Park KM, Kim SH, Lee DG, Rim HC: **Incorporating Lexical Knowledge into Biomedical NE Recognition.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:76-79.
37. Lee C, Hou WJ, Chen HH: **Annotating Multiple Types of Biomedical Entities: A Single Word Classification Approach.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:80-83.
38. Rössler M: **Adapting an NER-System for German to the Biomedical Domain.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:92-95.
39. Zhou G, Su J: **Exploring deep knowledge resources in biomedical name recognition.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:96-99.
40. Song Y, Kim E, Lee GG, Yi BK: **POSBIOTM-NER in the Shared Task of BioNLP/NLPBA2004.** *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004:100-103.
41. Fawcett T: **Roc graphs: Notes and practical considerations for data mining researchers.** *Tech Rep HPL-2003-4, HP Labs, Palo Alto, Ca* 2003.
42. Boser BE, Guyon I, Vapnik V: **A Training Algorithm for Optimal Margin Classifiers.** In *Proceedings of the Fifth Annual ACM Workshop on Computational Learing Theory* Edited by: Haussler D. New York: ACM Press; 1992:144-152.
43. Burges CJC: **A Tutorial on Support Vector Machines for Pattern Recognition.** *Data Mining and Knowledge Discovery* 1998, **2:**121-167.
44. Vapnik VN: *Statistical Learning Theory* New York: Wiley; 1998.
45. Hsu CW, Chang CC, Lin CJ: **A practical guide to support vector classification.** Tech. rep., Department of Computer Science and Information Engineering, National Taiwan University, Taipei; 2003.
46. Yarowsky D: **Unsupervised word sense disambiguation rivaling supervised methods.** In *Proceedings of the Thirty-Third conference on Association for Computational Linguistics* Edited by: Uszkoreit H. Association for Computational Linguistics; 1995:189-196.
47. Porter MF: **An algorithm for suffix stripping.** *Program* 1980, **14:**130-137.
48. Maloof M: **Learning when data sets are imbalanced and when costs are unequal and unknown.** *ICML-2003 Workshop on Learning from Imbalanced Data Sets II* 2003.
49. Bradley AP: **The use of the area under the ROC curve in the evaluation of machine learning algorithms.** *Pattern Recognition* 1997, **30:**1145-1159.
50. Alpaydin E: **Combined 5 × 2 cv F Test for Comparing Supervised Classification Learning Algorithms.** *Neural Computation* 1999, **11:**1885-1892.
51. Chang CC, Lin CJ: **LIBSVM: a library for support vector machines.** [http://www.csie.ntu.edu.tw/~cjlin/libsvm].
52. McCallum AK: **Bow: a toolkit for statistical language modeling, text retrieval, classification and clustering.** 1996 [http://www-2.cs.cmu.edu/~mccallum/bow/].
53. Keerthi SS, Lin CJ: **Asymptotic behaviors of support vector machines with Gaussian kernel.** *Neural Computation* 2003, **15:**1667-1689.
54. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C: **Text Classification using String Kernels.** *Journal of Machine Learning Research* 2002, **2:**419-444.
55. Cancedda N, Gaussier E, Goutte C, Renders JM: **Word-Sequence Kernels.** *Journal of Machine Learning Research* 2003, **3:**1059-1082.
56. Hastie T, Rosset S, Tibshirani R, Zhu J: **The Entire Regularization Path for the Support Vector Machine.** *Journal of Machine Learning Research* 2004, **5:**1391-1415.
57. Vapnik VN: *The nature of statistical learning theory* Springer-Verlag New York, Inc; 1995.

Paper II

Kernels incorporating word positional information in natural language disambiguation tasks

Tapio Pahikkala, Sampo Pyysalo, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 442–447, Menlo Park, Ca, 2005. AAAI Press.

Paper III

Matrix Representations, Linear Transformations, and Kernels for Disambiguation in Natural Language

Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. TUCS Techical Report 890, 2008. Submitted to a journal.

Paper IV

Fast n-fold cross-validation for regularized least-squares

Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, pages 83–90, Espoo, Finland, 2006. Otamedia.

Fast n-Fold Cross-Validation for Regularized Least-Squares

Tapio Pahikkala* Jorma Boberg*
Tapio Salakoski*
*Turku Centre for Computer Science (TUCS), Department of Information Technology, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
firstname.lastname@it.utu.fi

Abstract

Kernel-based learning algorithms have recently become the state-of-the-art machine learning methods of which the support vector machines are the most popular ones. Regularized least-squares (RLS), another kernel-based learning algorithm that is also known as the least-squares support vector machine, is shown to have a performance comparable to that of the support vector machines in several machine learning tasks. In small scale problems, RLS have several computational advantages as compared to the support vector machines. Firstly, it is possible to calculate the cross-validation (CV) performance of RLS on the training data without retraining in each CV round. We give a formal proof for this claim. Secondly, we can compute the RLS solution for several different values of the regularization parameter in parallel. Finally, several problems on the same data set can be solved in parallel provided that the same kernel function is used with each problem. We consider a simple implementation of the RLS algorithm for the small scale machine learning problems that takes advantage of all the above properties. The implementation is done via the eigen decomposition of the kernel matrix. The proposed CV method for RLS is a generalization of the fast leave-one-out cross-validation (LOOCV) method for RLS which is widely known in the literature. For some tasks, the LOOCV gives a poor performance estimate for the learning machines, because of the dependencies between the training data points. We demonstrate this by experimentally comparing the performance estimates given by LOOCV and CV in a ranking task of dependency parses generated from biomedical texts.

1 Introduction

Kernel-based learning algorithms (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) have recently become the state-of-the art machine learning methods of which the support vector machines are the most popular ones. In this paper, we consider Regularized least-squares (RLS) algorithm (see e.g. Rifkin (2002); Poggio and Smale (2003)), another kernel-based learning algorithm that is also known as the least-squares support vector machine (Suykens and Vandewalle, 1999). They were shown to have a performance comparable to that of the support vector machines in several machine learning tasks. Traditionally, RLS type of algorithms have been applied to regression problems but lately they have also been used on other machine learning problems. Recent classification tasks in which RLS have been successfully applied are, for example, disambiguation problems in natural language (Popescu, 2004), DNA classification (Ancona et al., 2005), and classification of intensive care nursing narratives (Hirissa et al., 2006). Another successful application area has been ranking or ordinal regression (Tsivtsivadze et al., 2005, 2006; Suominen et al., 2006; Pahikkala et al., 2006c).

Cross-validation (CV) is a commonly used method for the performance estimation and model selection for the learning algorithms. For RLS, it is widely known that the leave-one-out cross-validation (LOOCV) has a closed form whose computational complexity is quadratic with respect to the number of training examples. In reality, however, the training set may contain data points that are highly dependent with each other. In text categorization tasks, for example, it may happen that the training set contains several documents written by the same author while that author may not have written the new examples to be predicted with the learning machine. In this kind of situation, the LOOCV performance estimate becomes unreliable, because it may be much easier to predict the labels of the documents when the machine is trained with documents written by the same author. Fortunately, we often have a priori knowledge

of such clustering in the training data set and we can perform the CV so that we leave out the whole cluster of data points in each CV round. In this paper, we show that also the leave-cluster-out cross-validation can be performed for RLS with much smaller computational complexity than the naive approach in which the RLS would be retrained in each CV round. To our knowledge, this has not been considered in the literature.

2 Regularization Framework

Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (X \times \mathbb{R})^m$ be a training set of m training examples, where $x_i \in X$ are the training data points, $y_i \in \mathbb{R}$ are their labels, and X can be any set. We consider the Regularized Least-Squares (RLS) algorithm as a special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive introduction, see e.g. Poggio and Smale (2003)):

$$\min_f \sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \|f\|_k^2, \tag{1}$$

where l is the loss function used by the learning machine, $f : X \to Y$ is a function which maps the inputs $x \in X$ to the outputs $y \in Y$, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is a norm in a Reproducing Kernel Hilbert Space defined by a positive definite kernel function k. The second term is called a regularizer. The loss function used with RLS for regression problems is called least squares loss and is defined as

$$l(f(x), y) = (y - f(x))^2.$$

Note that if we use $l(f(x), y) = \max(y - f(x), 0)$, we obtain the support vector machines (SVM) for classification. Other choices of the loss function lead to other popular classifiers, for example, the SVM regression and kernel logistic regression. By the Representer Theorem (see e.g. Schölkopf et al. (2001)), the minimizer of equation (1) with the least-squares loss function has the following form:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i), \tag{2}$$

where $a_i \in \mathbb{R}$ and k is a kernel function.

3 Implementation

We now state the solution of RLS in the case where there are several output labels for each data point (see e.g. Rifkin and Klautau (2004) for a more comprehensive consideration). Below, $\mathcal{M}_{i\times j}(\mathbb{R})$ denotes the set of real valued matrices of dimension $i \times j$. Suppose we have a set of m training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (X \times \mathbb{R}^p)^m$, where $x_i \in X$ are the input variables, X can be any set, and y_i are the corresponding vectors of p output variables. Let $K \in \mathcal{M}_{m\times m}(\mathbb{R})$ be the kernel matrix generated from the training data points using the kernel function $k(x, z)$, that is, $K_{ij} = k(x_i, x_j)$. Let $Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ be a matrix whose rows are the vectors of the output variables, that is, it has one column per each subproblem. Further, let $G = (K + \lambda I)^{-1}$, where $I \in \mathcal{M}_{m\times m}(\mathbb{R})$ is the identity matrix. Because the kernel function from which the kernel matrix is generated is positive definite, the matrix $K+\lambda I$ is invertible when $\lambda > 0$. Given a regularization parameter λ, the coefficient matrix is obtained as follows

$$A = GY \in \mathcal{M}_{m\times p}(\mathbb{R}) \tag{3}$$

whose columns are the coefficient vectors of the RLS solution (2) for each p output (for a proof, see e.g. Rifkin (2002)). Because the kernel matrix K is the same for all of the p problems, we obtain all solutions from (3) with approximately the cost of solving only one problem.

3.1 Solving RLS via eigen decomposition of kernel matrix

Solution (3) can be obtained simply by calculating the inverse G of the matrix $K+\lambda I$. Instead of calculating the inverse, we compute the solution by first calculating the eigen decomposition of the kernel matrix

$$K = V \Lambda V^{\mathrm{T}}, \tag{4}$$

where V is an orthogonal matrix that contains the eigenvectors of K and Λ is a diagonal matrix that contains the corresponding eigenvalues. Then, $G = (V\Lambda V^{\mathrm{T}}+\lambda I)^{-1} = V\widetilde{\Lambda}_\lambda V^{\mathrm{T}}$, where $\widetilde{\Lambda}_\lambda = (\Lambda+\lambda I)^{-1}$ is a diagonal matrix that contains the eigenvalues of G. The ith eigenvalue of G is $1/(\mu_i + \lambda)$, where μ_i is the ith eigenvalue of K. Note that we do not need to compute the matrix G, because the solution (3) can now be obtained as

$$A = V\widetilde{\Lambda}_\lambda V^{\mathrm{T}} Y. \tag{5}$$

From (5) we observe that after we have calculated the eigen decomposition of K, we can easily compute a whole array of RLS solutions for different values of λ. To compute the solution (5) for a certain value of λ, we need to calculate the product of the matrices

$V\widetilde{\Lambda}_\lambda \in \mathcal{M}_{m\times m}(\mathbb{R})$ and $V^{\mathrm{T}}Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ which is fast when the number of subproblems p is small compared to the number of training examples m. We can then select the regularization parameter for each subproblem with a cross-validation which we consider below. Of course, different subproblems may prefer different values of λ. In that case, we do not obtain the column vectors of A from (5), but one by one from $A_p = V\widetilde{\Lambda}_{\lambda_p}V^{\mathrm{T}}Y$, where a_p is the pth column of A and λ_p is the value of the regularization parameter preferred by the pth subproblem.

3.2 Efficient Computation of Cross-Validation

We now consider an efficient computation of cross-validation (CV) for the RLS algorithm. By CV we indicate the method that is used to estimate the performance of the learning algorithm with a given data set. The outline of the method is the following. First, the data set is partitioned into subsets called CV folds. Next, the learning machine is trained with the whole data set except one of the folds that is used to measure the performance of the machine. Each of the CV folds is held out from the training set at a time and the CV performance of the machine is obtained by averaging over the performances measured with the different folds.

In order to calculate the CV performance of a learning machine explicitly, we need to train the machine n times, where n is the number of the CV folds. This is, in many cases, computationally cumbersome, especially if the number of the folds is large. Fortunately, it is possible to obtain the CV performance of RLS with a smaller computational cost than in the naive approach.

We now prove a lemma that we use below to derive a faster method for the computation of CV. The proof is similar to the proof of the leave-one-out lemma (see e.g. Wahba (1990)). For simplicity, we prove only the case in which the number of output variables is one. However, it is easy to extend the lemma for p output variables.

Lemma 1. *Let L be a set of indices of the data points that are held out from the training set and let f_L be the function obtained by training the RLS algorithm with the whole data set except the set of data points indexed by L. By definition, f_L is the solution to the following variational problem*

$$\min_f \sum_{i\notin L}(f(x_i)-y_i)^2 + \lambda\|f\|_k^2. \tag{6}$$

The function f_L is also the solution to the following variational problem

$$\min_f \sum_{i\notin L}(f(x_i)-y_i)^2 + \sum_{i\in L}(f(x_i)-f_L(x_i))^2 + \lambda\|f\|_k^2$$

Proof. We observe that for any function f

$$\begin{aligned}\sum_{i\notin L}(f(x_i)-y_i)^2 + \sum_{i\in L}(f(x_i)-f_L(x_i))^2 + \lambda\|f\|_k^2 &\geq \sum_{i\notin L}(f(x_i)-y_i)^2 + \lambda\|f\|_k^2 \\ &\geq \sum_{i\notin L}(f_L(x_i)-y_i)^2 + \lambda\|f_L\|_k^2 \\ = \sum_{i\notin L}(f_L(x_i)-y_i)^2 + \sum_{i\in L}(f_L(x_i)-f_L(x_i))^2 &+ \lambda\|f_L\|_k^2\end{aligned}$$

□

Using the above lemma, we are able to state the result that allows us to calculate the values of the output variables of the data points held out from the training set without explicitly retraining the algorithm with the rest of the training examples.

Let $I \in \mathcal{M}_{m\times m}(\mathbb{R})$ denote an identity matrix and let $I_L \in \mathcal{M}_{|L|\times m}(\mathbb{R})$ be a matrix that contains the rows of I indexed by L. We use the matrix I_L to "cut" out rows from other matrices, or alternatively to "add" rows consisting of zeros. Below, with any matrix $M \in \mathcal{M}_{m\times n}(\mathbb{R})$, where $n \in \mathbb{N}$, we use the subscript L to denote the left multiplication by I_L, that is, we denote $M_L = I_L M$. Further, we denote $M_{LL} = I_L M I_L^{\mathrm{T}}$ for $M \in \mathcal{M}_{m\times m}(\mathbb{R})$. Let $Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ be the label matrix corresponding to the training data and let us denote $B = KG$. Then

$$B = V\Lambda V^{\mathrm{T}}V\widetilde{\Lambda}_\lambda V^{\mathrm{T}} = V\Lambda\widetilde{\Lambda}_\lambda V^{\mathrm{T}}. \tag{7}$$

By the equation (2), the predicted output of the RLS algorithm for its training data is

$$\widehat{Y} = KA = KGY = BY. \tag{8}$$

Finally, let $Y' \in \mathcal{M}_{m\times p}(\mathbb{R})$ denote the matrix consisting of the output values of the training data points obtained using the function f_L.

Theorem 1. *The matrix Y'_L consisting of the output values of the held out data points predicted with f_L can be obtained from*

$$Y'_L = (I_{LL} - B_{LL})^{-1}(\widehat{Y}_L - B_{LL}Y_L). \tag{9}$$

Proof. According to the Lemma 1, the function f_L is obtained by training the RLS using the whole data set with a label matrix $Y - I_L^{\mathrm{T}} Y_L + I_L^{\mathrm{T}} Y'_L$. Knowing the label matrix, we now use the equation (8) to compute the output matrix Y'.

$$\begin{aligned} Y' &= B(Y - I_L^{\mathrm{T}} Y_L + I_L^{\mathrm{T}} Y'_L) \\ &= \widehat{Y} - B I_L^{\mathrm{T}} Y_L + B I_L^{\mathrm{T}} Y'_L. \end{aligned}$$

By multiplying with I_L from left we get

$$\begin{aligned} Y'_L &= \widehat{Y}_L - B_{LL} Y_L + B_{LL} Y'_L \\ \Leftrightarrow (I_{LL} - B_{LL}) Y'_L &= \widehat{Y}_L - B_{LL} Y_L \\ \Leftrightarrow Y'_L &= (I_{LL} - B_{LL})^{-1} (\widehat{Y}_L - B_{LL} Y_L). \end{aligned}$$

In order to the last equivalence to hold, we have to ensure the invertibility of the matrix $I_{LL} - B_{LL}$. Let γ_i be the ith eigenvalue of B. From (7) we observe that $\gamma_i = (\Lambda \widetilde{\Lambda}_\lambda)_{i,i} = \frac{\mu_i}{\mu_i + \lambda}$. Because $0 \leq \gamma_i < 1$, the matrix $I - B$ is a positive definite. From the positive definiteness of $I - B$, it follows that all its principal submatrices $I_{LL} - B_{LL}$ have strictly positive determinants (see e.g. Meyer (2000)) from which the invertibility follows. □

For a held out set of of size $|L|$, the time consuming part in the computation of (9) is the calculation of the matrix B_{LL}. When we have solved the RLS problem via the eigen decomposition of the kernel matrix, the elements of B_{LL} can be computed using the eigenvectors V and the diagonal elements of $\Lambda \widetilde{\Lambda}_\lambda$, since $B = V \Lambda \widetilde{\Lambda}_\lambda V^{\mathrm{T}}$. Thus, the computational complexity of calculating the outputs Y'_L for the held out set is $O(|L|^2 m)$. If we perform an n-fold cross-validation with the training set, the number and the size of the held out sets are n and m/n, respectively, and the overall complexity of the cross-validation is $O(n(m/n)^2 m) = O(m^3/n)$.

The larger the number of folds in the cross-validation is, the faster is its computation. In the extreme case where the size of the held out set is 1, the computational complexity is $O(m^2)$, since we only have to calculate the diagonal elements of B in the whole cross-validation process. Indeed, from Theorem 1, we obtain as a special case the known result of the leave-one-out cross-validation (LOOCV) for RLS (this case has been proved, for example, by Vapnik (1979); Wahba (1990); Green and Silverman (1994)).

Corollary 1. *Let $f(x_j)$ and $f_j(x_j)$ denote the output of the RLS algorithm for the training example x_j, when the algorithm is trained with all examples and all examples except x_j, respectively. We can calculate the value of the output $f_j(x_j)$ as follows*

$$f_j(x_j) = \frac{f(x_j) - B_{j,j} Y_j}{1 - B_{j,j}}. \tag{10}$$

From the computational complexity perspective, the LOOCV should be preferred with the RLS. However, in practice, there are often cases where the LOOCV should not be used to estimate the performance of the learning algorithm because of dependencies between the training data points. Note also that the sizes of the cross-validation folds do not have to be equal. Therefore, we can divide the training set into folds of different sizes according to the dependencies between the training points. Below, we discuss those cases in more detail.

4 Experiments

With real world data, it is often the case that the data points used to train a machine learning method are not completely independent. For example, we may have several text documents written by the same author in text categorization tasks. A document may be very similar to other the documents written by the same author but very different compared to the documents written by an other author. In these cases, the leave-one-out cross-validation (LOOCV) performance of a learning machine may not be a good estimate of the true performance of the learning machine. This is, because on the contrary to the case with the training set, it is rarely the case that a document to be classified with a trained learning machine is written by the same author as some of the documents in the training set. Generally, we say that the training set consists of clusters of data points. By a cluster we indicate a subset of training examples that are mutually dependent.

Fortunately, we often have a priori knowledge of such clustering in the training data set and we can perform the CV so that we leave out the whole cluster of data points in each CV round. For example, in our earlier experiments using support vector machines and Bayesian classifiers on natural language disambiguation tasks (Pahikkala et al., 2005a,b,c, 2006a,b), we often extracted several examples of the words to be disambiguated from a single text document. When we used bag-of-words kind of features extracted from the whole text, the examples originating from the same text document formed a cluster in the training set. Clearly, it does not happen in practice that a learning machine trained to detect context sensitive spelling errors, is trained with examples originating from the same document the examples to be predicted

are originated from. Thus, we had to perform the cross-validation on the document level so that no two examples from the same document end up in different cross-validation folds.

Here we demonstrate the "clustered training set effect" by comparing the LOOCV performance of a trained RLS to a leave-cluster-out cross-validation (LCOCV) performance so that each fold in the LCOCV consists of the training examples that form a cluster in the training set. The demonstration is done with the problem of dependency parse ranking of sentences extracted from biomedical texts. Here we give a brief introduction to the problem, the data, and the solution approach. For a detailed description, see the paper by Tsivtsivadze et al. (2005).

To generate the training data, we took one hundred sentences from the BioInfer corpus (Pyysalo et al., 2006). Each sentence in the corpus has a manually tagged linkage that corresponds to the "correct" parse that a parser is supposed to output for the sentence. For each sentence, we use link grammar parser (Sleator and Temperley, 1991) to generate a set of candidate parses. The number of candidate parses depends of the sentence. If the parser generates more than 20 parses for a sentence, we randomly select 20 of them and discard the rest. Otherwise, we keep all candidate parses. For each candidate parse we calculate a score value that indicates how close to the correct parse it is. The score value is the F-score calculated from the link differences between the candidate parse and the correct parse as follows (Tsivtsivadze et al., 2005):

$$F = \frac{2TP}{2TP + FP + FN},$$

where TP, FP, and FN are the numbers of true positives (the links present in both the candidate and the correct parse), false positives (links present in the candidate parse but not in the correct one), and false negatives (links present in the correct parse but not in the candidate), respectively. The training set is constructed from the candidate parses and their score values. The task of the learning machine is, for a sentence, to rank its candidate parses in the order of their score values. The RLS algorithm is, in fact, trained to regress the score values of the parses but in this paper, we are only interested of the ranking of the candidate parses for each sentence.

We measure the ranking performance by calculating Kendalls τ_b correlation coefficient (Kendall, 1970) for the parse candidate set of each sentence. Note that the coefficient is calculated for each sentence separately, since we are not interested of the mutual order of the parses originating from different sentences. The overall performance for the whole data set is obtained by taking the average of the correlation coefficients of the sentences in the data set.

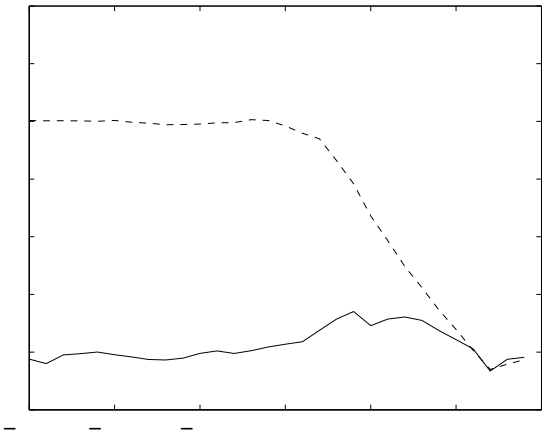


Figure 1: The ranking performance of the RLS algorithm computed with LOOCV (dashed line) and LCOCV (solid line). The x-axis denotes the value of the regularization parameter in a logarithmic scale. The y-axis is the ranking performance measured with τ_b correlation coefficient.

The kernel function, that we use as a similarity measure of the parses, is described in detail by Tsivtsivadze et al. (2005). The kernel function has the drawback that two parses originating from a same sentence have almost always larger mutual similarity than two parses originating from different sentences. Therefore, the data set consisting of the parses is heavily clustered in the feature space determined by the kernel function. The clustered structure of the data can have a strong effect on the performance estimates obtained by cross-validation, because data points that are in the same cluster as a held out point have a dominant effect on the predicted output of the held out point. This does not, however, model the real world, since a parse ranker is usually not trained with parses originating from the sentence from which the new parse with an unknown F-score is originated. The problem can be solved by performing the cross-validation on the sentence level so that all the parses generated from a sentence would always be either in the training set or in the test set.

In order to compare the performance estimates given by the LOOCV and LCOCV, we train an RLS algorithm with the data set described above. Recall that after we have a trained RLS, we obtain the LOOCV output for each parse in the data using (10). From the LOOCV output, we calculate the

F-scores for each sentence and compute their average. We calculate the LCOCV performance by leaving each sentence out from the training set at a time and computing the F-scores for their parses with (9). We make a grid search for the regularization parameter λ of the RLS algorithm with the grid points $2^{-15}, 2^{-14}, \ldots, 2^{14}$. With the grid search, we can test the performances of LOOCV and LCOCV in the model selection of RLS.

The results of the comparison are illustrated in Figure 1. From the figure, we observe that the performance difference between the LOOCV and the LCOCV is, with some values of the regularization parameter, over 0.3 correlation points. Thus, LOOCV clearly overestimates the ranking performance. Moreover, the LOOCV prefers small values of the regularization parameter (the most preferable value of the regularization parameter is 2^{-2}) while the LCOCV prefers a more regularized solution (the most preferable value of the regularization parameter is 2^4). Therefore, we can also conclude that the LOOCV may not be a good method for the model selection when the training set is clustered. Indeed, when we test the ranking performance of the RLS with one hundred test sentences unseen to the RLS, we obtain correlations 0.37 and 0.38 with the machines trained with the whole training set and with the regularization parameters preferred by the LOOCV and LCOCV, respectively.

5 Conclusion

The regularized least-squares (RLS) algorithm has been shown to be a competitive alternative to the standard support vector machines in several machine learning tasks. It also have several computational advantages, such as solving several tasks in parallel, fast tuning of the regularization parameter, and fast leave-one-out cross-validation (LOOCV).

The LOOCV method, however, is not always a suitable method for performance estimation or model selection because of dependencies between the training data points. In several learning problems, the training data set may be clustered so that the prediction of a held out data point will be unrealistically easy if the data points that belong in the same cluster with the held out data point are kept in the training set. Since it may not happen in the real world that a data point, whose output is to be predicted with a learning machine, belongs in the same cluster with some of the training data points, the LOOCV estimate does not reflect the reality.

We introduce and prove a closed form of an n-fold cross-validation performance estimate for the RLS algorithm. The closed form can be used to calculate a leave-cluster-out cross-validation (LCOCV), in which a whole cluster of training examples is held out from the training set in each cross-validation round avoiding the harmful effect of the clustered training set.

We experimentally demonstrate the effect of the clustered data set on the LOOCV performance estimate with a ranking task of dependency parses generated from biomedical texts. With the same data and task, it is also demonstrated that the LCOCV is better than LOOCV as a model selection tool.

Acknowledgments

This work has been supported by Tekes, the Finnish Funding Agency for Technology and Innovation.

References

Nicola Ancona, Rosalia Maglietta, Annarita D'Addabbo, Sabino Liuni, and Graziano Pesole. Regularized least squares cancer classifiers from dna microarray data. *BMC Bioinformatics*, 6 (Suppl 4):S2, 2005.

P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models, A Roughness Penalty Approach*. Chapman and Hall, London, 1994.

Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. In *The 20th International Congress of the European Federation for Medical Informatics (MIE 2006)*, Maastricht, Netherlands, 2006. To appear.

Maurice G. Kendall. *Rank Correlation Methods*. Griffin, London, 4. edition, 1970.

Carl D. Meyer. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

Tapio Pahikkala, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Incorporating external information in bayesian classifiers via linear feature transformations. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Proceedings of the 5th International Conference on*

NLP (FinTAL 2006), volume 4139 of *Lecture Notes in Computer Science*, pages 399–410, Heidelberg, Germany, 2006a. Springer-Verlag.

Tapio Pahikkala, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation. *BMC Bioinformatics*, 6(1):157, 2005a.

Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Matrix representations, linear transformations, and kernels for natural language processing, 2006b. Submitted.

Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Improving the performance of bayesian and support vector classifiers in word sense disambiguation using positional information. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 90–97, Espoo, Finland, 2005b. Otamedia OY.

Tapio Pahikkala, Sampo Pyysalo, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Kernels incorporating word positional information in natural language disambiguation tasks. In Ingrid Russell and Zdravko Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 442–447, Menlo Park, Ca, 2005c. AAAI Press.

Tapio Pahikkala, Evgeni Tsivtsivadze, Jorma Boberg, and Tapio Salakoski. Graph kernels versus graph representations: a case study in parse ranking. In *ECML/PKDD'06 workshop on Mining and Learning with Graphs (MLG'06)*, 2006c. To appear.

Tomaso Poggio and Steve Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544, 2003.

Marius Popescu. Regularized least-squares classification for word sense disambiguation. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 209–212, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: A corpus for information extraction in the biomedical domain, 2006. Submitted.

Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning.* PhD thesis, MIT, 2002.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In D. Helmbold and R. Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426, Berlin, Germany, 2001. Springer-Verlag. ISBN 3-540-42343-5.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1991.

Hanna Suominen, Tapio Pahikkala, Marketta Hiissa, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Relevance ranking of intensive care nursing narratives. In *proceedings of KES2006 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems*, 2006. to appear.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, 1999.

Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality-convolution kernel and its application to dependency parse ranking. In Moonis Ali and Richard Dapoigny, editors, *Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2006)*, volume 4031 of *Lecture Notes in Computer Science*, pages 610–618, Heidelberg, Germany, 2006. Springer-Verlag.

Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Regularized least-squares for parse ranking. In A. Fazel Famili, Joost N. Kok, José Manuel Peña, Arno Siebes, and A. J. Feelders, editors, *Proceedings of the 6th International Symposium on Intelligent Data Analysis*, volume 3646 of *Lecture Notes in Computer Science*, pages 464–474, Heidelberg, Germany, September 2005. Springer-Verlag.

V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

Grace Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

Paper V

Learning to rank with pairwise regularized least-squares

Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. In Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

Paper VI

Computer-assisted identification of multi-trace electrophoretic patterns in differential display experiments

Heidi Vähämaa, Pekka Ojala, Tapio Pahikkala, Olli S. Nevalainen, Riitta Lahesmaa, and Tero Aittokallio. *Electrophoresis*, 28(6):879–893, 2007.

Turku Centre for Computer Science
TUCS Dissertations

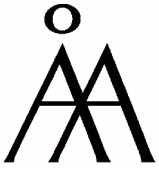
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory
86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku

- Department of Information Technology
- Department of Mathematics



Åbo Akademi University

- Department of Information Technologies

Turku School of Economics

- Institute of Information Systems Sciences

ISBN 978-952-12-2091-3
ISSN 1239-1883

Tapio Pahikkala

New Kernel Functions and Learning Methods for Text and Data Mining

Tapio Pahikkala

New Kernel Functions and Learning Methods for Text and Data Mining