# Machine learning and performance estimation methods for ranking problems

Antti Airola*
Department of Information Technology
University of Turku

## Abstract

The task of learning to rank refers to the machine learning problem, where the aim is to infer from past observations a ranking model that can order new objects according to how well they match some underlying criterion. Ranking problems are commonly encountered in applications such as document retrieval, game playing, information extraction and recommender systems. While learning to rank has been a topic of active research for more than a decade, developing scalable learning methods, and reliable and efficient validation methods has proven to be challenging.

The doctoral thesis of the author, summarized in this article, provides the following main contributions towards solving these issues. First, novel training algorithms based on optimizing a pairwise criterion in the regularized risk minimization framework are derived. Previously, the most well established method of this type is the ranking support vector machine (RankSVM). The introduced RankRLS method, as well as the proposed improvements to RankSVM, lead to orders of magnitude gains in efficiency, without decrease in predictive performance. Second, novel cross-validation approaches are proposed in order to account for the data dependencies and multivariate performance measures characteristic of ranking tasks. Computational short-cuts allow the efficient computation of these estimates for the RankRLS method. Finally, an application study introducing a novel method for information extraction from biomedical text combines several key ideas of the thesis, resulting in a state-of-the-art solution to the problem.

**Keywords:** cross-validation, information extraction, kernel methods, learning to rank, machine learning, regularized least-squares, regularized risk minimization, support vector machine

## 1 Introduction

In learning to rank, the aim is to infer from previously collected data a ranking function, that is able to order new sets of objects according to how well they match the underlying ranking criterion. Learning is necessary in such applications, where we do not know the true underlying ranking function, but rather have access to previous judgements made by some actor, typically a human being. Two typical examples of such applications are search engines that rank documents according to their match to user queries [Joachims 2002] and recommender systems [Minkov et al. 2010]. For an overview of the problem domain and related work we refer to [Liu 2009; Fürnkranz and Hüllermeier 2010; Airola 2011].

We assume the availability of training data, which contains both feature representations and judgements related to objects from the application domain of interest. These previous judgements may be categorical (e.g. good/bad, 1-5 stars) or supplied as real-valued utility scores, where a higher score indicates higher rank than a lower one. More generally, the information may be provided in terms of pairwise comparisons, indicating that certain objects are preferred over other ones. Based on the training data we learn a scoring function, that maps the feature representation of any given object to a predicted utility score. When ranking a new set of objects, the ranking is constructed by sorting the objects according to predicted scores. An accurate ranking function is such that the produced rankings match well the true rankings also for such sets of objects that were not observed in the training set. Finding such a function requires striking a balance between the phenomena of underfitting and overfitting, as learning may fail either due to considering too simple hypotheses (e.g. linear models for a highly non-linear concept) or due to allowing too rich set of hypotheses and ending up simply modeling the noise in the training data.

The family of regularized kernel methods embodies one of the mainstream approaches to machine learning [Schölkopf and Smola 2002; Shawe-Taylor and Cristianini 2004]. These methods allow the use of structured data and non-linear modeling, and offer principled ways to dealing with both the underfitting and overfitting phenomena, while still leading to convex optimization problems, where globally optimal solutions can be found. Widely used kernel methods include the support vector machine (SVM) [Vapnik 1995] and the regularized least-squares (RLS) [Poggio and Smale 2003] algorithms. The ranking support vector machine (RankSVM) method extends standard SVMs to learning to rank by casting the problem as a binary classification problem over pairs of objects. While the method has been demonstrated to achieve excellent ranking performance, the training methods proposed in [Herbrich et al. 1999; Joachims 2002] unfortunately lead to solving optimization problems whose size may depend quadratically rather than linearly on the size of the training set. For linear RankSVM more efficient training methods are known [Joachims 2006; Chapelle and Keerthi 2010], but these are limited to settings where the number of possible ranks in the data can be assumed a small constant.

Cross-validation is one of the most widely used techniques in machine learning for estimating the predictive power of the learned models. However, standard cross-validation approaches such as the leave-one-out method turn out to be highly unreliable in many ranking settings. The assumption that the training data is sampled independently is routinely broken, leading to biased estimates (see e.g. [Pahikkala et al. 2012b]). The use of multivariate ranking performance measures, such as the area under the ROC curve (AUC) and its generalizations leads to problems when predictions made on different rounds of cross-validation are combined together [Parker et al. 2007; Forman and Scholz 2010]. Further, straightforward implementations of cross-validation procedures also incur high computational costs, due to the necessity to re-train a learning algorithm multiple times.

The thesis [Airola 2011] summarized in this article provides a number of contributions towards solving the aforementioned problems. RankRLS [Pahikkala et al. 2009] is a novel learning to rank method, that combines regularized risk minimization with a pair-

*e-mail:antti.airola@utu.fi

wise least-squares loss function. This choice leads to a closed-from solution expressed as a system of linear equations, that can be solved efficiently even though the loss is implicitly computed over all pairs in the training set. Further, using matrix update formulas, the regularization parameter can be selected, and exact cross-validation estimates can be computed, at the same asymptotic cost as training RankRLS once. [Airola et al. 2011b] introduces an improved version of the linear RankSVM training method of [Joachims 2006] reducing the worst-case quadratic computational cost to $O(m\log(m))$ scaling by applying self-balancing search trees. [Airola et al. 2011a] considers the use of Nyström approximation for generating low-dimensional feature representations for training kernel machines, and how to efficiently re-compute these during cross-validation, in cases when there are dependencies present in the data. In [Airola et al. 2011c] we consider the problem of cross-validation when applying pairwise performance measures, demonstrating that the proposed leave-pair-out procedure provides almost unbiased performance estimates with low variance. In [Airola et al. 2008] an application in protein-protein interaction extraction from scientific articles combines a number of ideas, later improved and refined in the aforementioned works, in order to achieve computational efficiency and reliable performance estimates.

## 2 Learning to rank

### 2.1 Regularized risk minimization

Let the input space $\mathcal{X}$, and output space $\mathcal{Y}$ be sets. We are supplied with a training set $Z$ containing inputs, and associated label information, defined as $Z = (X, Y) \in \mathcal{X}^m \times \mathcal{Y}$. By $X = (x_1, \ldots, x_m) \in \mathcal{X}^m$ we denote the set of $m$ inputs belonging to the training set. By $Y \in \mathcal{Y}$ we denote a structured object containing the label information associated with $X$. In learning to rank, in the simplest case $\mathcal{Y} = \mathbb{R}^m$, meaning that each input is associated with one utility score. More generally, the labels may have dependencies between them, or be associated with pairs of inputs rather than with individual inputs.

The learning algorithm takes as input a finite training set $Z$, and outputs a scoring function $f : \mathcal{X} \to \mathbb{R}$, which aims to model the dependency between the inputs and the labels. Let $X \in \mathcal{X}^m$, $m \in \mathbb{N}$ be a sequence of inputs. Then by $f(X) \in \mathbb{R}^m$ we denote the vector of predictions for this sample. A loss function

$$l : \bigcup_{m\in\mathbb{N}} \mathbb{R}^m \times \mathcal{Y} \mapsto [0, \infty)$$

measures how well the predicted labels and true labels for a data set match. The goal of learning is to find such a scoring function that would incur minimal expected loss on data drawn from the same distribution from which the training data was sampled from. In practice we can never compute the expected loss, but are rather limited to using the estimate known as the empirical risk

$$R(f) = l(f(X), Y),$$

that is simply the loss computed on the training set.

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a a finitely positive semi-definite kernel function. In the kernel methods framework, we consider hypotheses of the type

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

where $\alpha_i \in \mathbb{R}$. In the special case where $\mathcal{X} = \mathbb{R}^n$ and $k = \langle\cdot,\cdot\rangle$ is the standard inner product in $\mathbb{R}^n$ this setting reduces to standard linear models of the type $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}$. We denote the hypothesis space as $\mathcal{H}$.

The regularized risk minimization problem (see e.g. [Evgeniou et al. 2000]) can be defined as

$$\underset{f\in\mathcal{H}}{\operatorname{argmin}}\, R(f) + \lambda\|f\|^2, \qquad (1)$$

where the first term measures how well $f$ fits the training data, and the second term called the regularizer measures the complexity of the hypothesis, and $\lambda > 0$ is the regularization parameter.

### 2.2 Ranking problem

In the following, we assume that the label information in the training set is supplied in terms of pairwise preferences. These may be collected directly from pairwise comparisons. For example, [Joachims 2002], collected such preferences from clickthrough data from search engine, by considering clicked links to be preferred over those that were not chosen by a user. When supplied with scored data, preferences can be constructed by considering objects with higher scores being preferred over those with lower ones. In such cases not all objects may be comparable, for example the standard approach to modeling document retrieval problems [Joachims 2002; Liu 2009] results in a setting where data consists of query-document pairs, and preferences are constructed only between pairs such as are associated with same query. The concept of preference graph allows us to unify all these settings, though for computational reasons we might often in practice want to avoid its explicit construction.

A set of pairwise preferences can be encoded as a directed preference graph, where input points serve as vertices, and the edges encode preferences between the vertices. By an edge $e_i = (h, j)$, where $h \neq j$, we encode that $x_h$ is preferred over $x_j$. We denote a preference graph drawn from the underlying distribution as

$$E = (e_1, \ldots, e_l).$$

In addition to pairwise preferences, we may in some settings have access to preference magnitudes, that denote to which degree an object is preferred over another. For scored data, preference magnitude can be defined as $y - y'$. If such information is not available and magnitudes are required, we may assume that each preference has a magnitude 1. In the following, we use $E_M$ to denote a set of pairwise preferences augmented with preference magnitudes, meaning that each $e_i = (h, j, w_i) \in E_M$ contains a magnitude $w_i$ encoding the degree, to which $x_h$ is preferred over $x_j$.

Following [Herbrich et al. 1999], we measure the discrepancy between predicted and true rankings using the pairwise ranking error, defined as

$$l(f(X), E) = \sum_{(i,j)\in E} H(f(x_j) - f(x_i)).$$

where $H$ is the Heaviside step function defined as

$$H(a) = \begin{cases} 1, & \text{if } a > 0 \\ 1/2, & \text{if } a = 0 \\ 0, & \text{if } a < 0 \end{cases}.$$

Intuitively, the loss can be considered as an estimate of the probability that the function is able to correctly predict, which of two randomly drawn examples is preferred over another. No polynomial time algorithm is known for minimizing this loss, which motivates the convex approximations introduced next.

## 2.3 RankRLS

The magnitude preserving pairwise ranking loss is defined as

$$l(f(X), E_M) = \sum_{(h,j,w_i)\in E} (w_i - f(x_h) + f(x_j))^2.$$

By inserting this loss in to (1), we recover the RankRLS method. The method extends the RLS regression method [Poggio and Smale 2003] by casting the problem of ranking into a pairwise regression framework. In [Pahikkala et al. 2009] we proved that a global minimizer of the RankRLS risk functional can be found by solving a system of linear equations.

Let us denote by $m$ the number of training inputs, by $l$ the number of pairwise preferences in the training set, and by $n$ the dimensionality of the feature space[1]. The complexity of training RankRLS using the algorithm described in [Pahikkala et al. 2009] is $O(m^3)$, which is based on solving a $m \times m$ linear system, using matrix factorization algorithms. This is a significant improvement compared to the straightforward approach of training RankRLS using a black-box RLS solver trained directly on the pairwise preferences, as this would result in highly impractical $O(l^3)$ worst case complexity (note that in many problems $l \approx m^2$). When using the linear kernel, RankRLS can be solved in $O(n^3 + \min(n^2m + m^2n + l, n^2l))$ time. If $n << m$ this can be quite efficient. Thus using basic dense linear algebra techniques based on matrix factorization, RankRLS can be trained in a time that is either cubic in the number of training examples, or cubic in the dimensionality of feature space.

Perhaps the main advantage of the RankRLS approach is the number of computational shortcuts made possible by the closed form solution. First, it can be shown that solutions for different regularization parameter values $\lambda$ can subsequently be computed by re-using computations needed for RankRLS training in quadratic time. This is quite useful, since one rarely knows in advance the suitable value for $\lambda$, rather it is typically chosen by grid searching. Second, based on low-rank matrix update operations, one can develop computationally efficient cross-validation algorithms for RankRLS. These methods in effect allow a trained RankRLS model with a minimal number of operations to "unlearn" the effects of a hold-out set of examples. In [Pahikkala et al. 2009] we introduce exact methods for leave-pair-out cross-validation and leave-query-out cross-validation, and prove that these estimates can be computed with no additional asymptotic cost compared to training RankRLS once.

When using kernels, reduced set approximation can be used to scale RankRLS training beyond a few thousand training examples. This approach is considered in detail in [Pahikkala et al. 2009], and can be seen as a special case of the Nyström approximation scheme studied in [Airola et al. 2011a]. Finally, let us consider application domains where the data is sparse, meaning that the data matrix is filled mostly with zeroes. Using the linear kernel, it is possible to make use of this sparsity, avoiding explicitly constructing dense $m \times m$ or $n \times n$ matrices. Using the conjugate gradient method, the RankRLS optimization can rather be formalized in terms of sparse matrix - vector products. The basic technique is described in [Pahikkala et al. 2009], more detailed analysis and further experimental results are presented in [Airola et al. 2010]. Let $\overline{n}$ be the average number of non-zero features per example, and $t$ the number of iterations that conjugate gradient optimization needs to converge. Then linear RankRLS can be trained with $O(tm\overline{n} + tl)$ cost.

[1]Learning from scored data is more efficient than from pairwise preferences. In this setting, the terms containing $l$ can be removed from all the following RankRLS complexities

| | Running times | | | | | |
|---|---|---|---|---|---|---|
| Inputs | 200 | 500 | 1000 | 2000 | 2500 | 4000 |
| RankRLS | 1 | 3 | 10 | 48 | 83 | 280 |
| RankSVM | 2 | 150 | 1740 | 13707 | 20055 | - |

**Table 1:** *Runtime comparisons of training kernel RankRLS and RankSVM in CPU seconds. The number of inputs ranges from 200 to 4000, the runtimes are measured in seconds.*

## 2.4 RankSVM

The pairwise hinge loss is defined as

$$l(f(X), E) = \sum_{(i,j)\in E} \max(1 - f(x_i) + f(x_j), 0).$$

By inserting this loss in to (1), we recover the RankSVM method. The method extends SVMs [Vapnik 1995] by casting the problem of ranking into a pairwise classification framework. The approach was first considered in [Herbrich et al. 1999].

In theory, any standard SVM solver can be used to solve also the RankSVM problem by training on pairwise preferences directly. This approach was originally adapted in [Herbrich et al. 1999]. Further, the popular kernel RankSVM solver included in the SVM$^{\text{light}}$ software package uses a standard SVM solver trained on pairwise preferences for training the RankSVM [Joachims 2002]. The downside of this approach is that the computational complexity of these solvers becomes dependent not on the number of examples, but on the number of pairwise preferences, leading to $O(m^4)$ or worse scaling. For scored data and linear kernel, [Joachims 2006] introduce a method with $O(m\overline{n} + m\log(m) + rm)$, where $r$ is the number of different utility levels in the data. If the number of allowed scores is not restricted, at worst case $r = m$ with the resulting complexity $O(m\overline{n} + m^2)$, meaning quadratic behavior.

[Airola et al. 2011b] presents a technique for removing this dependence on $r$ from the complexity. The method uses balanced binary search trees to speed up loss and subgradient computations, allowing $O(m\overline{n} + m\log(m))$ worst case behavior for linear RankSVM training. On large enough data sets this can make a substantial difference in training times, reducing days of training time to minutes. Efficient kernel RankSVM training can be achieved using the empirical kernel map corresponding to the Nyström approximation, explored especially in [Airola et al. 2011a], in order to convert the dual RankSVM problem to the primal problem. This idea was introduced already in [Pahikkala et al. 2009]. The use of this approach for kernel RankSVM training has subsequently been independently considered by [Chapelle and Keerthi 2010]. Briefly put, using the efficient linear RankSVM training method this approach leads to $O(mk^2 + m\log m)$ RankSVM training complexity, where $k << m$ is a parameter that controls the amount of approximation.

## 2.5 Experimental results

In [Pahikkala et al. 2009], we report results for an experimental comparison of RankRLS and RankSVM, as well as RLS regression as baseline. Considered problems include collaborative filtering, document retrieval, n-best re-ranking of syntactic parses and text categorization. The main conclusion of the experiments is that there are rarely significant differences in ranking performance between the RankRLS and RankSVM approaches, while both tend to outperform the baseline method. However, as discussed before, there are major differences in computational efficiency.

In Table 1, we re-produce subset of the runtime comparison between RankRLS and RankSVM presented in [Pahikkala et al.

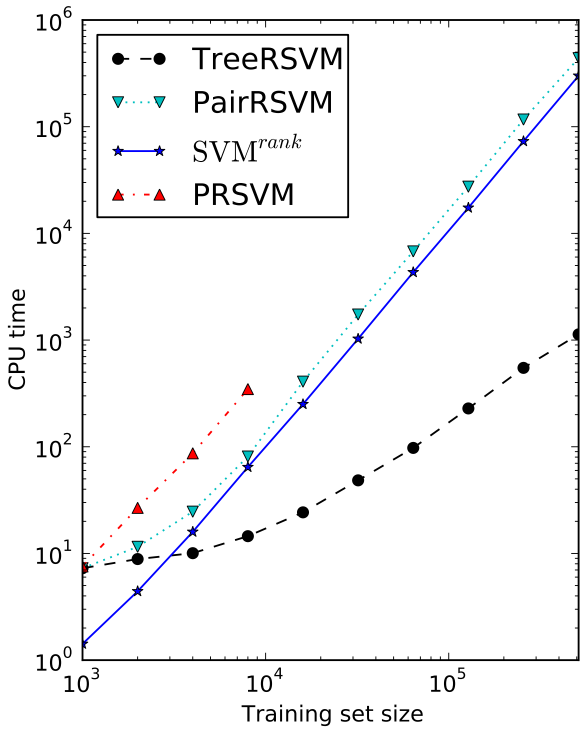

**Figure 1:** *Linear RankSVM runtime comparison.*

2009]. The RankRLS implementation is based on an early version of the RLScore software, whereas the RankSVM implementation is from $\mathrm{SVM}^{\mathrm{light}}$. As can be expected from the computational complexity considerations, RankRLS has much better scalability than the standard RankSVM implementation. When one further considers the costs of performing regularization parameter selection and cross-validation the difference becomes much larger, since these procedures can be performed essentially for free for RankRLS.

For the linear kernel, both RankRLS and RankSVM can be scaled to much larger problem sizes. Next, we consider how the RankSVM training algorithm that was introduced in [Airola et al. 2011b] for linear models and scored data, compares to the best previously known methods. The proposed method (TreeRSVM) is compared to the methods of [Joachims 2006] ($\mathrm{SVM}^{\mathrm{rank}}$ and PairRSVM) as well as to that of [Chapelle and Keerthi 2010] (PRSVM). With 512000 training examples, training $\mathrm{SVM}^{rank}$ took 83 hours, whereas training TreeRSVM took only 18 minutes in the same setting. Both methods reach the same solution. These results also suggest quite simple approach to improving nonlinear RankSVM training methods, since as discussed previously, for regularized kernel methods the kernelized learning problem can always be cast into a linear learning problem. Comparing results in [Airola et al. 2011b] and [Airola et al. 2010], it can be established that the fastest linear RankSVM and RankRLS training methods have quite similar scalability.

## 3 Cross-validation

Cross-validation is one of the most widely used methods in machine learning for model selection and performance evaluation. In cross-validation, one repeatedly splits the data set into two parts, a training set and a holdout set. The model is trained on the training set, after which it is used to make predictions on the holdout set. This procedure is repeated a number of times, after which a final estimate of the performance is computed over all the holdout sets on which predictions were made on. One major challenge in applying cross-validation is the computational cost, due to having to train a learner multiple times; our contributions towards solving this issue were briefly discussed in the previous section. In typical ranking problems further challenges are encountered with regards to reliability of cross-validation results, these are discussed next.

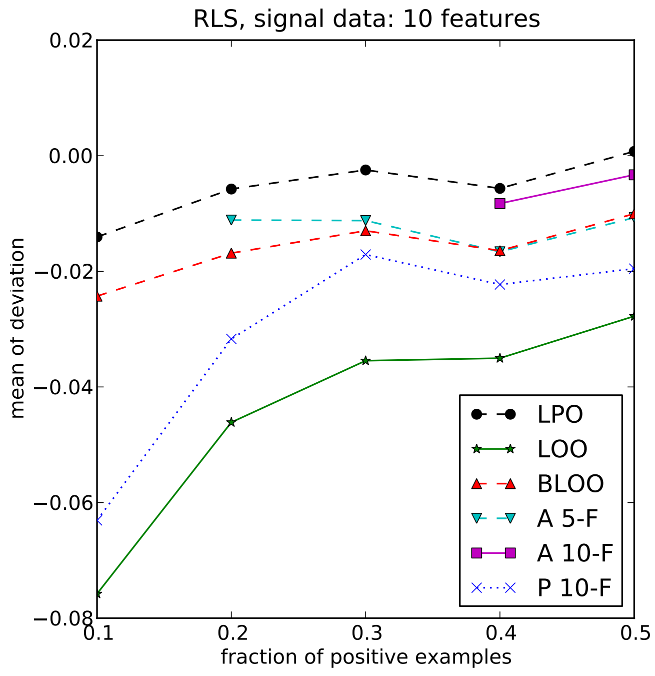

**Figure 2:** *Comparison of different cross-validation strategies.*

First, there exist two general strategies for aggregating cross-validation results together. [Bradley 1997] who considered the specific problem of AUC estimation referred to these alternatives as pooling and averaging. In pooling all the predictions are combined together and the performance measure is then computed over the combined predictions. In averaging, the performance is computed separately for each holdout set, and finally the average over these is computed. For classical univariate performance measures such as classification accuracy, or squared error, it makes little difference which strategy is used. However, for pairwise ranking measures, there is a clear difference. Briefly put, in pooling most of the compared pairs are formed from predictions made on different rounds of cross-validation. It can be shown that this can lead to substantial biases in the results. The averaging strategy avoids comparing predictions from different rounds. However, for typical approaches this leads to increased variability in the estimates, since most of the pairs are in this case ignored during cross-validation. These issues are further discussed in [Airola et al. 2011c].

Next, we consider one of the simulation studies reported in [Airola et al. 2011c], where a number of cross-validation strategies, are applied for AUC estimation. The problem is a bipartite ranking task (or equivalently, binary classification), with 30 training examples and 10 features. Instances from both classes are first drawn from normal distributions with unit variance and no covariance between the features. Nine of the features have mean zero for both classes, the tenth has mean 0.5 for the positive, and $-0.5$ for the negative

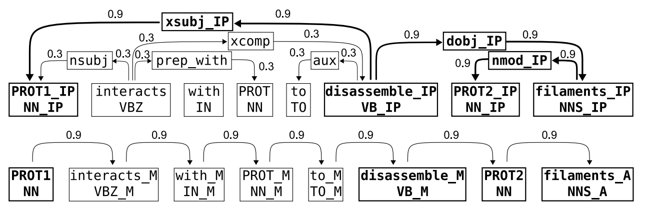

**Figure 3:** *Graph representation of a sentence.*

class. We compute the cross-validation estimates over 10000 repetitions of the experiments, and compare these to the true expected AUC computed on a simulated test set of 10000 examples. The deviation measures the mean difference between estimated and true AUC. For an unbiased estimator this should be 0, positive values indicate optimistic, and negative values pessimistic bias.

The compared standard approaches are averaged 10-fold cross-validation (A 10-F), averaged 5-fold cross-validation (A 5-F), pooled 10-fold cross-validation (P 10-F), and leave-one-out (LOO). Further, we consider the balanced leave-one-out variant proposed by [Parker et al. 2007] (BLOO). Finally, we test the leave-pair-out method (LPO) where pairs of data points are used as holdout sets, which we argue to be the most natural choice for pairwise performance measures such as AUC. The results, presented in Figure 2 demonstrate substantial biases in all the pooled methods. The averaged methods have less bias, with LPO providing the least biased estimates. Further experiments verify this trend over a large number of settings, while experiments that consider the variability of the approaches show LPO to be more reliable than the other averaging methods in this respect. The conclusion of the study is that for AUC estimation LPO should be preferred over other approaches, in cases where it can be computationally afforded.

Further studies in cross-validation can be found in [Airola et al. 2011a] where we consider how to correctly and efficiently deal with hold-out basis vectors when using the Nyström approximation to speed up training of kernel methods. In [Airola et al. 2008], we consider in an information extraction study the substantial biases that can arise due to the fact that training examples generated from the same sentence are much more similar to each other, than those generated from different sentences. The findings support the notion that for data where strong dependencies occur between the examples, dependent data points should never be split between the training and test sets, in order to ensure reliable performance estimation.

## 4 Biomedical information extraction

The task of protein-protein interaction (PPI) extraction from scientific literature is one of the major tasks considered in the field of biomedical natural language processing. Online resources, such as PubMed offer researchers access to millions of research articles in the biomedical domain, making manual search for stated results about interactions impractical. Rather, automated information extraction systems are needed. In the past, both rule-based approaches and methods based on machine learning have been proposed for solving the problem (see references in [Airola et al. 2008; Airola 2011]). In [Airola et al. 2008], we proposed and evaluated a novel approach for PPI extraction. The approach combines a novel graph kernel approach to learning from syntactic parses of sentences with an RLS based learning method, and efficient and reliable cross-validation strategies for model selection.

In Figure 3 we see an example of a sentence talking about PPI-interactions. In addition to the sentence itself, the figure presents a dependency parse for the graph, that was generated using an automated syntactic parser. Based on both the syntactic parse, as well as the linear ordering of the words in the sentence, the system has to decide which of the protein pairs appearing in a sentence are stated to interact, and which are not. Using kernel methods learning from this type of structured data becomes possible once we can define a suitable kernel function between the graph representations. In [Airola et al. 2008] we propose such an approach that is based on random walks in a graph, extending the earlier work of [Gärtner et al. 2003; Pahikkala et al. 2006].

The final method combines a wide range of approaches considered in other articles related to the thesis. Learning and parameter optimization is done by optimizing a RLS based objective function, using the Nyström approximation to scale the method. The fast leave-document-out cross-validation approach is very closely related to the computational shortcuts presented in [Pahikkala et al. 2009; Airola et al. 2011a], while reliable AUC-estimation requires accounting for the issues considered in [Airola et al. 2011c].

The experimental results presented in [Airola et al. 2008] demonstrated, that the method reached state-of-the-art performance compared to methods that had been previously proposed. Since then [Miwa et al. 2009] have proposed an improved solution to the same problem, that incorporates the proposed graph kernel as one of its main components. [Tikk et al. 2010] have further recently conducted a large-scale benchmark study of different kernel-based approaches to PPI-extraction, and reported the graph kernel to be among the most competitive approaches. As a further development, Turku Event Extraction System [Björne et al. 2011] provides a solution to the more challenging task of extracting complex structured interactions, incorporating many of the same syntactic features as used by the graph kernel.

## 5 Open source software

The importance of sharing open source implementations of published methods has recently been advocated in the machine learning community [Sonnenburg et al. 2007]. Accordingly, we are currently working on developing the RLScore machine learning open source software framework, which is made publicly available[2] under the MIT open source license. The package contains the RankRLS algorithms, as well as a wide variety of other learning methods. Further, the All-paths graph kernel -software package[3] [Airola et al. 2008], as well as the TreeRankSVM package [Airola et al. 2011b] are made publicly available under open source license.

## 6 Conclusion

In this article we have summarized the main contributions of the thesis [Airola 2011]. One of the major themes of research in this work was the development of computationally efficient algorithms for training and cross-validation, especially through the use of matrix algebra based techniques. In our related research, not included in the thesis, similar ideas have been applied for example in order to learn preference relations over relational graphs [Pahikkala et al. 2010], extend feature selection methods to genome wide scale [Pahikkala et al. 2012a], and for speeding up unsupervised and semi-supervised RLS training [Gieseke et al. 2012]. Most of the developed methods are or will be made freely available as part of the RLScore open source software.

[2]`http://tucs.fi/rlscore`

[3]`http://mars.cs.utu.fi/PPICorpora/GraphKernel.html`

## Acknowledgements

I would like to thank the supervisors of my thesis, Adjunct Prof. Tapio Pahikkala and Prof. Tapio Salakoski, and the other coauthors. My thanks go also to the reviewers of the thesis, Prof. Eyke Hüllermeier and Associate Prof. Juho Rousu as well as to my opponent Adjunct Prof. Timo Honkela, for their constructive criticisms and encouragement. The thesis work was done at the department of Information Technology at the University of Turku, as a member of the graduate programme of Turku Centre for Computer Science. I would further like to acknowledge the financial support provided by the Academy of Finland, Nokia Foundation, Sofia and Wilhelm Fagerholm Scholarship Fund and the Turku University Foundation for the thesis work. Finally, I would like to thank the Pattern Recognition Society of Finland for the dissertation award and the invitation to present this work.

## References

AIROLA, A., PYYSALO, S., BJÖRNE, J., PAHIKKALA, T., GINTER, F., AND SALAKOSKI, T. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics 9 Suppl 11*.

AIROLA, A., PAHIKKALA, T., AND SALAKOSKI, T. 2010. Large scale training methods for linear RankRLS. In *Proceedings of the ECML/PKDD-Workshop on Preference Learning (PL-10)*, E. Hüllermeier and J. Fürnkranz, Eds.

AIROLA, A., PAHIKKALA, T., AND SALAKOSKI, T. 2011. On learning and cross-validation with decomposed Nyström approximation of kernel matrix. *Neural Processing Letters 33*, 1, 17–30.

AIROLA, A., PAHIKKALA, T., AND SALAKOSKI, T. 2011. Training linear ranking SVMs in linearithmic time using red-black trees. *Pattern Recognition Letters 32*, 9, 1328–1336.

AIROLA, A., PAHIKKALA, T., WAEGEMAN, W., DE BAETS, B., AND SALAKOSKI, T. 2011. An experimental comparison of cross-validation techniques for estimating the area under the ROC curve. *Computational Statistics & Data Analysis 55*, 4, 1828–1844.

AIROLA, A. 2011. *Kernel-Based Ranking: Methods for Learning and Performance Estimation*. PhD thesis, Turku Centre for Computer Science.

BJÖRNE, J., HEIMONEN, J., GINTER, F., AIROLA, A., PAHIKKALA, T., AND SALAKOSKI, T. 2011. Extracting contextualized complex biological events with rich graph-based feature sets. *Computational Intelligence 27*, 4, 541–557.

BRADLEY, A. P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition 30*, 7, 1145–1159.

CHAPELLE, O., AND KEERTHI, S. S. 2010. Efficient algorithms for ranking with SVMs. *Information Retrieval 13*, 3, 201–215.

EVGENIOU, T., PONTIL, M., AND POGGIO, T. 2000. Regularization networks and support vector machines. *Advances in Computational Mathematics 13*, 1–50.

FORMAN, G., AND SCHOLZ, M. 2010. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explorations 12*, 1, 49–57.

FÜRNKRANZ, J., AND HÜLLERMEIER, E. 2010. Preference learning. In *Encyclopedia of Machine Learning*. 789–795.

GÄRTNER, T., FLACH, P. A., AND WROBEL, S. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the Sixteenth Annual Conference on Learning Theory and Seventh Annual Workshop on Kernel Machines (COLT/Kernel 2003)*, Springer, B. Schölkopf and M. K. Warmuth, Eds., vol. 2777 of *Lecture Notes in Artificial Intelligence*, 129–143.

GIESEKE, F., KRAMER, O., AIROLA, A., AND PAHIKKALA, T. 2012. Efficient recurrent local search strategies for semi- and unsupervised regularized least-squares classification. *Evolutionary Intelligence*, 1–17. Accepted for publication.

HERBRICH, R., GRAEPEL, T., AND OBERMAYER, K. 1999. Support vector learning for ordinal regression. In *Proceedings of the Ninth International Conference on Articial Neural Networks (ICANN 1999)*, Institute of Electrical Engineers, London, 97–102.

JOACHIMS, T. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2002)*, ACM Press, New York, NY, USA, D. Hand, D. Keim, and R. Ng, Eds., 133–142.

JOACHIMS, T. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2006)*, ACM Press, New York, NY, USA, T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, Eds., 217–226.

LIU, T.-Y. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval 3*, 3, 225–331.

MINKOV, E., CHARROW, B., LEDLIE, J., TELLER, S., AND JAAKKOLA, T. 2010. Collaborative future event recommendation. In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM 2010)*, ACM, New York, NY, USA, J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, Eds., 819–828.

MIWA, M., SÆTRE, R., MIYAO, Y., AND TSUJII, J. 2009. Protein-protein interaction extraction by leveraging multiple kernels and parsers. *International Journal of Medical Informatics 78*, e39–e46.

PAHIKKALA, T., TSIVTSIVADZE, E., BOBERG, J., AND SALAKOSKI, T. 2006. Graph kernels versus graph representations: a case study in parse ranking. In *Proceedings of the ECML/PKDD 2006 workshop on Mining and Learning with Graphs (MLG 2006)*, T. Gärtner, G. C. Garriga, and T. Meinl, Eds.

PAHIKKALA, T., TSIVTSIVADZE, E., AIROLA, A., JÄRVINEN, J., AND BOBERG, J. 2009. An efficient algorithm for learning to rank from preference graphs. *Machine Learning 75*, 1, 129–165.

PAHIKKALA, T., WAEGEMAN, W., AIROLA, A., SALAKOSKI, T., AND DE BAETS, B. 2010. Conditional ranking on relational data. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2010)*, Springer, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds., vol. 6322 of *Lecture Notes in Computer Science*, 499–514.

PAHIKKALA, T., OKSER, S., AIROLA, A., SALAKOSKI, T., AND AITTOKALLIO, T. 2012. Wrapper-based selection of genetic features in genome-wide association studies through fast matrix operations. *Algorithms for Molecular Biology 7*, 1, 11.

PAHIKKALA, T., SUOMINEN, H., AND BOBERG, J. 2012. Efficient cross-validation for kernelized least-squares regression with sparse basis expansions. *Machine Learning 87*, 3, 381–407.

PARKER, B. J., GUNTER, S., AND BEDO, J. 2007. Stratification bias in low signal microarray studies. *BMC Bioinformatics 8*, 326.

POGGIO, T., AND SMALE, S. 2003. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS) 50*, 5, 537–544.

SCHÖLKOPF, B., AND SMOLA, A. J. 2002. *Learning with kernels*. MIT Press, Cambridge, Massachusetts, USA.

SHAWE-TAYLOR, J., AND CRISTIANINI, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge.

SONNENBURG, S., BRAUN, M. L., ONG, C. S., BENGIO, S., BOTTOU, L., HOLMES, G., LECUN, Y., MÜLLER, K. R., PEREIRA, F., RASMUSSEN, C. E., RÄTSCH, G., SCHÖLKOPF, B., SMOLA, A., VINCENT, P., WESTON, J., AND WILLIAMSON, R. 2007. The need for open source software in machine learning. *Journal of Machine Learning Research 8*, 2443–2466.

TIKK, D., THOMAS, P., PALAGA, P., HAKENBERG, J., AND LESER, U. 2010. A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature. *PLoS Computational Biology 6*, 7, e1000837+.

VAPNIK, V. N. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.