# Learning to Rank with Pairwise Regularized Least-Squares

Tapio Pahikkala Evgeni Tsivtsivadze Antti Airola
Jorma Boberg Tapio Salakoski

Turku Centre for Computer Science (TUCS)
Department of Information Technology
University of Turku
Joukahaisenkatu 3-5 B
20520 Turku, Finland
firstname.lastname@utu.fi

## ABSTRACT

Learning preference relations between objects of interest is one of the key problems in machine learning. Our approach for addressing this task is based on pairwise comparisons for estimation of overall ranking. In this paper, we propose a simple preference learning algorithm based on regularized least squares and describe it within the kernel methods framework. Our algorithm, that we call RankRLS, minimizes a regularized least-squares approximation of a ranking error function that counts the number of incorrectly ranked pairs of data points. We consider both primal and dual versions of the algorithm. The primal version is preferable when the dimensionality of the feature space is smaller than the number of training data points and the dual one is preferable in the opposite case. We show that both versions of RankRLS can be trained as efficiently as the corresponding versions of standard regularized least-squares regression, despite the fact that the number of training data point pairs under consideration grows quadratically with respect to the number of individual points. As a representative example of a case where the data points outnumber features we choose the Letor dataset. For the opposite case, we choose the parse ranking task. We show that on the Letor dataset the primal RankRLS performs comparably to RankSVM and RankBoost algorithms that are used as baselines. Moreover, we show that the dual RankRLS notably outperforms the standard regularized least-squares regression in parse ranking. We suggest that the main advantage of RankRLS is the computational efficiency both in the primal and the dual versions, especially since the efficient implementation of the latter is not straightforward, for example, for the support vector machines.

## 1. INTRODUCTION

Ranking can be considered as a task in which the aim is to learn a function capable of arranging data points according to a given preference relation [6]. Tasks of this type are often cast as classification problems, where the training set is composed of data point pairs, in which one point is preferred over the other, and the class label of a pair indicates the direction of the preference [8, 10]. Although the preference learning algorithm usually has smaller computational complexity then that of classification algorithm the major drawback associated with this approach is that the number of data point pairs grows quadratically with respect to the size of the dataset, thus making training of the algorithm on the whole dataset too cumbersome.

Recently, it has been shown that the RLS classifiers (see e.g. [18]), also known as the least-squares support vector machines [21], have a classification performance comparable to the regular support vector machines (SVM). We propose a ranking algorithm which is based on minimizing the regularized least-squares (RLS) error when predicting the output variable differences on the pairs of data points, and hence call it RankRLS. We show that while the number of possible output variable differences grows quadratically with respect to the number of training inputs, the training is as efficient as the training of a standard RLS regressor for the individual output variables. The information regarding the output variable differences, that the algorithm is supposed to learn, is stored in a graph defined for the training set. The direction of preference between some pairs of the data points 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 them 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. Similarly, in the task of parse ranking that we consider in our experiments, we are not interested in the directions of reference between parses generated for different sentences.

We also derive a primal version of our algorithm that is more efficient than the kernel version when the number of training data points is larger than the number of features.

Many other ranking algorithms, such as RankSVM [8] for example, can be efficiently trained with this type of data. The ranking performance difference between RankSVM and our method depends only on the type of the loss function (the hinge loss in RankSVM and the least-squares loss in our case). However, we may suggest that our method is the most applicable when used with kernel functions due to the efficient regression of the pairwise output variable differences. To our knowledge, while this type of SVM algorithms have been proposed for tasks where the number of pairwise differences to be learned is reasonably small, these algorithms are not efficient when the number of the differences is large. A common approach in this case is to reduce the number of pairs to be learned using, for example, clustering techniques (see e.g. [17, 3]).

## 2. PAIRWISE REGRESSION

First, we construct a training set from a given set of $m$ data points. A data point $z = (x, y)$ consist of an input variable $x \in \mathcal{X}$ and an output variable $y \in \mathbb{R}$, where $\mathcal{X}$, called the input space, can be any set. In our web search example, an input variable is a query-document pair and its associated output variable is the relevance of the document to the query, that is, $y > 0$ when the document matches the query and $y = 0$ otherwise. Next, we define an undirected graph whose vertices correspond to the training inputs. Two vertices in the graph are connected with an edge if the corresponding pair of data points is relevant to the task. For example, in a web search task, two query-document pairs are connected with an edge if the query-document pairs are the results of the same query. We determine the graph via $m \times m$ adjacency matrix $W$, that is, $W_{i,j} = 1$ when the vertices indexed by $i$ and $j$ are connected, and $W_{i,j} = 0$ otherwise. Note that we set $W_{i,i} = 1$ for all $i$. Further, let $X = (x_1, \ldots, x_m) \in (\mathcal{X}^m)^{\mathrm{T}}$ be a sequence of inputs, where $(\mathcal{X}^m)^{\mathrm{T}}$ denotes the set of row vectors whose elements belong to $\mathcal{X}$. Correspondingly, we define $Y = (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathbb{R}^m$ be a sequence of the corresponding output variables. Altogether, we define the training set to be the triple $S = (X, Y, W)$.

### 2.1 Regularization Framework

Let us denote $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$, and let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$ be the hypothesis space. In order to construct an algorithm that selects a hypothesis $f$ from $\mathcal{H}$, we have to define an appropriate cost function that measures how well the hypotheses fit to the training data. We would also like to avoid too complex hypotheses that overfit at training phase and are not able to generalize to unseen data. Following [19], we consider the framework of regularized kernel methods in which $\mathcal{H}$ is so-called reproducing kernel Hilbert space (RKHS) defined by a positive definite kernel function $k$. The kernel functions (see e.g. [7, 20]) are defined as follows. Let $\mathcal{F}$ denote the feature vector space. For any mapping

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

the inner product

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

of the mapped data points is called a kernel function. We also denote the sequence of feature mapped inputs as

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

for all $X \in (\mathcal{X}^m)^{\mathrm{T}}$. Further, we define the symmetric kernel matrix $K \in \mathbb{R}^{m \times m}$, where $\mathbb{R}^{m \times m}$ denotes the set of real matrices of type $m \times m$, as

$$K = \Phi(X)^{\mathrm{T}} \Phi(X) = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{pmatrix}$$

for all $X \in (\mathcal{X}^m)^{\mathrm{T}}$. Unless stated otherwise, we assume that the kernel matrix is strictly positive definite, that is, $A^{\mathrm{T}} K A > 0$ for all $A \in \mathbb{R}^m, A \neq 0$. This can be ensured, for example, by performing a small diagonal shift.

Following [19], we define RKHS determined by the input space $\mathcal{X}$ and the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

$$\mathcal{H} = \left\{ f(x) = \sum_{i=1}^{\infty} \beta_i k(x, x_i), \beta_i \in \mathbb{R}, x_i \in \mathcal{X}, \|f\|_k < \infty \right\},$$

where $\|f\|_k$ denotes the norm of the function $f$ in $\mathcal{H}$. Using RKHS as our hypothesis space, we define the learning algorithm as

$$\mathcal{A}(S) = \operatorname*{argmin}_{f \in \mathcal{H}} J(f),$$

where

$$J(f) = c(f(X), Y, W) + \lambda \|f\|_k^2, \tag{1}$$

$f(X) = (f(x_1), \ldots, f(x_m))^{\mathrm{T}}$, $c$ is a real valued cost function, and $\lambda \in \mathbb{R}_+$ is a regularization parameter controlling the tradeoff between the cost on the training set and the complexity of the hypothesis. By the generalized representer theorem ([19]), the minimizer of (1) has the following form:

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

where $a_i \in \mathbb{R}$. Using this notation, we rewrite $f(X) = KA$ and $\|f\|_k^2 = A^{\mathrm{T}} K A$, where $A = (a_1, \ldots, a_m)^{\mathrm{T}}$.

To measure how well a hypothesis $f \in \mathcal{H}$ is able to predict the direction of preference for the data point pairs that are relevant to the task, we consider the following cost function that simply counts the number of incorrectly predicted pairs (see e.g. [4] for a similar type of approach):

$$c(f(X), Y, W) = \frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} d(y_i - y_j, f(x_i) - f(x_j)) \tag{3}$$

where

$$d(\alpha, \beta) = \frac{1}{2} \left| \operatorname{sign}(\alpha) - \operatorname{sign}(\beta) \right|$$

and $\operatorname{sign}(\cdot)$ is the signum function.

$$\operatorname{sign}(r) = \begin{cases} 1 & \text{when} \quad r > 0 \\ 0 & \text{when} \quad r = 0 \\ -1 & \text{when} \quad r < 0 \end{cases}.$$

The direction of preference between data points $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$ is determined by $\operatorname{sign}(y_i - y_j)$. Similarly, the predicted direction of preference is determined by $\operatorname{sign}\big(f(x_i) - f(x_j)\big)$. For example, let $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$ be two data points, where $x_i$ and $x_j$ correspond to document-query pairs resulting from the same query. The document corresponding to $x_i$ is relevant to the query in question and the document corresponding to $x_j$ is not, that is, $y_i > 0$ and $y_j = 0$. Then, the document-query pair $x_i$ is preferred over $x_j$.

It is well-known that the use of cost functions like (3) leads to intractable optimization problems. Therefore, instead of using (3), we use functions approximating it. Namely, we use the following type of least-squares approximation of $d(\alpha, \beta)$ so that we are, in fact, regressing the differences $y_i - y_j$ with $f(x_i) - f(x_j)$:

$$\widetilde{d}(\alpha, \beta) = (\alpha - \beta)^2. \tag{4}$$

Note also that when using (4), not only the sign of $y_i - y_j$ but also its magnitude determine the objective function (1).

Before presenting the solution for the minimization problem using the least-squares approximation, we introduce some notation used. Let $L = D - W$ be the Laplacian matrix (see e.g. [2]) of the graph $W$, where $D$ is a diagonal matrix whose entries are defined as $D_{i,i} = \sum_{j=1}^{m} W_{i,j}$.

## 2.2 Dual RankRLS

The next theorem characterizes a method we call dual RankRLS or simply RankRLS.

THEOREM 1. *Let $S = (X, Y, W)$ be a training set and let*

$$\mathcal{A}(S) = \underset{f \in \mathcal{H}}{\operatorname{argmin}}\, J(f), \tag{5}$$

*where*

$$J(f) = c(f(X), Y, W) + \lambda \|f\|_k^2 \tag{6}$$

*and*

$$c(f(X), Y, W) = \frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} \widetilde{d}(y_i - y_j, f(x_i) - f(x_j)). \tag{7}$$

*be the algorithm under consideration. A coefficient vector $A \in \mathbb{R}^m$ that determines a minimizer of (6) for a training set $S$ is*

$$A = (LK + \lambda I)^{-1} LY, \tag{8}$$

*where $L$ is the Laplacian matrix of the graph $W$.*

PROOF. According to the representer theorem, the minimizer of (6) is of the form (2), that is, the problem of finding the optimal hypothesis can be solved by finding the coefficients $a_i, 1 \leq i \leq m$. We observe that for any vector $r \in \mathbb{R}^m$ and an undirected weighted graph $W$ of $m$ vertices, we can write

$$\begin{aligned}
\frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} (r_i - r_j)^2 &= \sum_{i,j=1}^{m} W_{i,j} r_i^2 - \sum_{i,j=1}^{m} W_{i,j} r_i r_j \\
&= \sum_{i=1}^{m} r_i^2 \sum_{j=1}^{m} W_{i,j} - \sum_{i,j=1}^{m} W_{i,j} r_i r_j \\
&= r^{\mathrm{T}} D r - r^{\mathrm{T}} W r \\
&= r^{\mathrm{T}} L r,
\end{aligned}$$

where $D$ and $L$ are the degree matrix and the Laplacian matrix of the graph determined by $W$. Therefore, by selecting $r = Y - KA$, we rewrite the cost function (7) in a matrix form as

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

and hence the algorithm (5) is rewritten as

$$\mathcal{A}(S) = \underset{A}{\operatorname{argmin}}\, J(A),$$

where

$$J(A) = (Y - KA)^{\mathrm{T}} L (Y - KA) + \lambda A^{\mathrm{T}} K A. \tag{9}$$

We take the derivative of $J(A)$ with respect to $A$:

$$\begin{aligned}
\frac{d}{dA} J(A) &= -2KL(Y - KA) + 2\lambda K A \\
&= -2KLY + (2KLK + 2\lambda K) A
\end{aligned}$$

We set the derivative to zero and solve with respect to $A$:

$$\begin{aligned}
A &= (KLK + \lambda K)^{-1} KLY \\
&= (LK + \lambda I)^{-1} LY,
\end{aligned}$$

where the last equality follows from the strict positive definiteness of $K$. $\square$

The calculation of the solution (8) requires multiplications and inversions of $m \times m$-matrices. Both types of operations are usually performed with methods whose computational complexities are $O(m^3)$, and hence the complexity of RankRLS is equal to the complexity of the RLS regression.

We also observe that in the unregularized case when $\lambda = 0$, the standard least-squares regression solution is also a solution of (5), that is, $A = K^{-1} Y$ is one of the minimizers of (9).

## 2.3 Primal RankRLS

In some cases, the number of training data points is much larger than the number of dimensions $n$ in the feature space, that is, we assume that we can write $\mathcal{F} = \mathbb{R}^n$, where $n < m$. Then, the sequence of mapped inputs is a matrix

$$\Phi(X) = (\Phi(x_1), \ldots, \Phi(x_m)) \in \mathbb{R}^{n \times m}$$

and the function (2) minimizing (1) can be equivalently expressed as

$$f(x) = \Phi(x)^{\mathrm{T}} \Phi(X) A = \Phi(x)^{\mathrm{T}} w, \tag{10}$$

where

$$w = \Phi(X) A$$

denotes the normal vector of the hyperplane corresponding to the RLS solution in the feature space. Output prediction for new data points is, of course, more efficient with (10) than with (2) when $n < m$.

We next show that, when $n < m$, the training process can also be performed in a more efficient way than with dual RankRLS (8). Note, that here we do not have to assume the strict positive definiteness of $K$. We call this method the primal version of RankRLS. With the primal version, the computational complexity of the training process becomes more dependent on $n$ rather than on the training set size $m$.

Now we write the algorithm (5) as

$$\mathcal{A}(S) = \operatorname*{argmin}_{w} J(w),$$

where

$$J(w) = (Y - \Phi(X)^{\mathrm{T}}w)^{\mathrm{T}}L(Y - \Phi(X)^{\mathrm{T}}w) + \lambda w^{\mathrm{T}}w.$$

We take the derivative of $J(w)$ with respect to $w$:

$$\begin{aligned}\frac{d}{dw}J(w) &= -2\Phi(X)L(Y - \Phi(X)^{\mathrm{T}}w) + 2\lambda w \\ &= -2\Phi(X)LY + (2\Phi(X)L\Phi(X)^{\mathrm{T}} + 2\lambda I)w.\end{aligned}$$

We set the derivative to zero and solve with respect to $w$:

$$w = (\Phi(X)L\Phi(X)^{\mathrm{T}} + \lambda I)^{-1}\Phi(X)LY. \qquad (11)$$

The computational complexity of the matrix inversion is in this case $O(n^3)$.

Recall that $L = D - W$. The multiplication of the matrices $L$ and $\Phi(X)$ can be performed efficiently using the following sparse low-rank decomposition of $W$. Let $P \in \mathbb{R}^{n\times q}$ be a matrix whose rows are indexed by the query-document pairs and the columns are indexed by the queries. The value of $P_{i,j}$ is 1 when the $i$th document-query pair is associated with the $j$th query and 0 otherwise. Then, the adjacency matrix can be written as $W = PP^{\mathrm{T}}$.

For example, assume that we have two queries of which the first is associated with two and the second with three documents. Then, the adjacency matrix $W$ and its corresponding $P$ matrix are

$$W = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}. \qquad (12)$$

Using the sparse low-rank decomposition of $W$, the multiplication of $L$ and $\Phi(X)$ can be written as

$$\Phi(X)L\Phi(X)^{\mathrm{T}} = \Phi(X)D\Phi(X)^{\mathrm{T}} - \Phi(X)PP^{\mathrm{T}}\Phi(X)^{\mathrm{T}},$$

where $D$ is the degree matrix of $L$. The multiplication of $\Phi(X)$ with $D$ can be performed in $O(nm)$ time, because $D$ is a diagonal matrix. Moreover, the multiplication of $\Phi(X)$ with $P$ requires also only $O(nm)$ operations, since $\Phi(X)$ has $n$ rows and there are exactly $m$ nonzero elements in $P$. The multiplications that dominate the computational complexity are $\Phi(X)D$ with $\Phi(X)^{\mathrm{T}}$ and $\Phi(X)PP^{\mathrm{T}}$ with $\Phi(X)^{\mathrm{T}}$ which both need $O(n^2m)$ time.

Thus, the overall complexity of the primal RankRLS is $O(n^3 + n^2m)$ of which the first term corresponds to the matrix inversion involved in (11) and the second to the matrix multiplications, that is, the primal RankRLS can be trained as efficiently as the primal RLS regression. Further, when the dimensionality $n$ of the feature space is considered to be a constant, the training time is linear with respect to the number of training examples.

### 2.4 Training via Eigen Decomposition

We also note that by calculating the eigen decomposition of $LK$ in (8), it is possible to obtain the solutions for several values of the regularization parameter in the computational cost of calculating just one solution. Formally, let $V$ be the eigenvector matrix of $LK$ and let $\Lambda$ be a diagonal matrix containing the corresponding eigenvalues. The decomposition can be calculated in $O(m^3)$ time. If we store in memory the precomputed matrix $V$ and the vector $Q = V^{-1}LY$, the solution for a regularization parameter value $\lambda$ can be calculated from

$$A = V(\Lambda + \lambda I)^{-1}Q,$$

in $O(m^2)$ time, since $(\Lambda + \lambda I)$ is a diagonal matrix.

An analogous trick can be used also in the primal form (11) by calculating the eigen decomposition of the matrix $\Phi(X)L\Phi(X)^{\mathrm{T}}$. In that case, the complexity of the decomposition calculation is $O(n^3)$ and the solutions for different values of regularization parameter can be obtained in $O(n^2)$ time.

## 3. EXPERIMENTS

We first present an evaluation of RankRLS on the Letor[1] data set that was provided by the workshop organizers. The amount of training data in Letor is very large compared to the number of features. Therefore, we use the primal version of the RankRLS algorithm whose computational complexity depends on the dimensionality of the feature space instead of the number of training data points. The results of the experiments are presented in Section 3.1. The performance of RankRLS is very similar to that of the provided baseline methods RankSVM [8] and RankBoost [5]. Moreover, the standard RLS regression [15] is found to be inferior to the ranking algorithms in two out of the three Letor datasets.

However, we believe that the main advantage of RankRLS is its efficiency when used in high dimensional feature space. This is the case, for example, when complex kernel functions are used. Therefore, we also conduct experiments with the task of parse ranking using graph kernels. The parse ranking results are presented in Section 3.2.

In both tasks, document ranking and parse ranking, we use an adjacency matrix $W$ that encodes which pairs of document-query pairs or parses are relevant to the task. To keep an account which query or sentence each document-query pair or parse is associated with, we define $U_l \subseteq \{1, \ldots, m\}$, where $1 \leq l \leq q$ and $q$ is the number of queries or sentences used to create the training set and $U_l$ is the index set whose elements refer to the indices of the document-query pairs or parses that are associated with the $l$th query or sentence. Of course, $U_l \cap U_{l'} = \emptyset$ if $l \neq l'$, because each document-query pair or parse is associated with only one query or sentence. Then, the adjacency matrix corresponding to the ranking tasks is

$$W_{i,j} = \begin{cases} 1 & \text{when } i,j \in U_l \\ 0 & \text{otherwise} \end{cases}.$$

We also test whether the tied pairs of data points are beneficial in the training process. With a tie, we refer to a pair consisting of data points $(x, y)$ and $(x', y')$ in which $y = y'$.

[1]http://research.microsoft.com/users/tyliu/LETOR/

In this case, the adjacency matrix is

$$W_{i,j} = \begin{cases} 1 & \text{when } i, j \in U_l \text{ and } y_i \neq y_j \\ 0 & \text{otherwise} \end{cases} .$$

The sparse low-rank decomposition of the Laplacian matrix that corresponds to the setting where such pairs are excluded can be constructed analogously to the one described in Section 2.3.

## 3.1 Document Ranking

We perform experiments to evaluate the capability of the RankRLS to rank query-document pairs, a task common within the field of information retrieval. These experiments are run on the publicly available Letor benchmark dataset. Our results are compared to those of two state-of-the-art ranking algorithms, RankSVM and RankBoost. We also make a comparison to the standard RLS regressor.

The task of ranking query-document pairs is a problem central to document retrieval - given a query some of the available documents are more relevant in regards to it than some others. Because the user will usually be most interested in the top results returned, document retrieval systems are typically evaluated using performance measures such as mean average precision (MAP) [1] or normalized discounted cumulative gain (NDCG) [9] that give more weight to the correctness of those rankings highest in the ranking hierarchy.

Letor (LEarning TO Rank) is a collection of three datasets extracted from three corresponding information retrieval data collections, namely the OHSUMED, TREC2003 and TREC2004 datasets. The whole collection consists of a set of document-query pairs, with 16140 related to OHSUMED, 49171 to TREC2003 and 74170 to TREC2004. Each document-query pair is represented as an example with a quite small number of highly abstract features, the amount of which is 25 in OHSUMED and 44 in the TREC collections. In the TREC collections each example is labeled as 1 (relevant) or 0 (non-relevant). For OHSUMED examples there are three possible labels: 2 (relevant), 1 (possibly relevant) and 0 (non-relevant).

Our experiments are performed on each of the three datasets separately. We preprocess the datasets by normalizing all the feature values in them to values between 0 and 1 on per query basis. Thus, the feature mapping $\Phi(X)$ just transforms the original features to the normalized ones preserving the dimensionality of the input space, since we only use the linear kernel over the normalized features. We evaluate the performance of the algorithms with the precision at position 1 to 10, MAP, and NDCG at position 1 to 10. In the case of OHSUMED dataset, examples with label 2 are considered relevant and the rest non-relevant when calculating the MAP-scores.

We use 5-fold cross-validation for parameter choosing and performance evaluation. In each trial three of the folds are used for training, one for choosing the value of the regularization parameter $\lambda$, and one for testing the performance of the trained model. During the parameter selection phase, we also decide whether the tied pairs of data points should be excluded from the training process, since this turned out to have a noticeable effect on the ranking performance. The fold split used is the same as the one defined in the Letor dataset. We use the MAP performance measure to select the regularization parameter and to decide whether to include the ties.

| Task | RRLS | RLS | RSVM | RBoost |
|---|---|---|---|---|
| OHSUMED | 0.447 | 0.450 | 0.447 | 0.440 |
| TREC2003 | 0.257 | 0.212 | 0.256 | 0.212 |
| TREC2004 | 0.359 | 0.304 | 0.350 | 0.384 |

**Table 1: MAP-performance comparison of RankRLS and the baseline methods on the Letor dataset**

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| NDCG@1 | 0.549 | 0.527 | 0.495 | 0.498 |
| NDCG@2 | 0.492 | 0.499 | 0.476 | 0.483 |
| NDCG@3 | 0.477 | 0.491 | 0.465 | 0.473 |
| NDCG@4 | 0.465 | 0.474 | 0.459 | 0.461 |
| NDCG@5 | 0.453 | 0.461 | 0.458 | 0.450 |
| NDCG@6 | 0.450 | 0.454 | 0.455 | 0.442 |
| NDCG@7 | 0.451 | 0.459 | 0.447 | 0.439 |
| NDCG@8 | 0.447 | 0.460 | 0.445 | 0.436 |
| NDCG@9 | 0.447 | 0.458 | 0.443 | 0.433 |
| NDCG@10 | 0.443 | 0.452 | 0.441 | 0.436 |

**Table 2: OHSUMED: NDCG@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

The results are presented in Tables 1-7. All the values are averaged over the five folds. Table 1 contains the MAP-performance values for the algorithms on each of the three datasets. In addition, we provide NDCG and precision values at levels 1 to 10 for each of the datasets. Results for OHSUMED are presented in tables 2 and 3, for TREC2003 in tables 4 and 5, and for TREC2004 in tables 6 and 7.

We observe the performance of RankRLS to be comparable to the baseline ranking methods. None of the three ranking algorithms clearly outperforms the others in this comparison. Interestingly, the standard RLS regression performs better than all the ranking algorithms on OHSUMED dataset, whereas it is clearly inferior on the TREC2003 and TREC2004 datasets. To conclude, while RankRLS has a

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| P@1 | 0.644 | 0.635 | 0.634 | 0.605 |
| P@2 | 0.614 | 0.615 | 0.619 | 0.595 |
| P@3 | 0.586 | 0.602 | 0.592 | 0.586 |
| P@4 | 0.570 | 0.580 | 0.579 | 0.562 |
| P@5 | 0.557 | 0.560 | 0.577 | 0.545 |
| P@6 | 0.546 | 0.541 | 0.558 | 0.525 |
| P@7 | 0.534 | 0.545 | 0.536 | 0.516 |
| P@8 | 0.524 | 0.539 | 0.525 | 0.505 |
| P@9 | 0.516 | 0.525 | 0.517 | 0.494 |
| P@10 | 0.505 | 0.510 | 0.507 | 0.495 |

**Table 3: OHSUMED: P@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| NDCG@1 | 0.400 | 0.320 | 0.420 | 0.260 |
| NDCG@2 | 0.370 | 0.350 | 0.370 | 0.280 |
| NDCG@3 | 0.355 | 0.315 | 0.379 | 0.270 |
| NDCG@4 | 0.351 | 0.310 | 0.363 | 0.272 |
| NDCG@5 | 0.350 | 0.302 | 0.347 | 0.279 |
| NDCG@6 | 0.341 | 0.301 | 0.341 | 0.280 |
| NDCG@7 | 0.344 | 0.298 | 0.340 | 0.287 |
| NDCG@8 | 0.342 | 0.296 | 0.345 | 0.282 |
| NDCG@9 | 0.344 | 0.301 | 0.342 | 0.282 |
| NDCG@10 | 0.346 | 0.303 | 0.341 | 0.285 |

**Table 4: Trec2003: NDCG@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| P@1 | 0.400 | 0.320 | 0.420 | 0.260 |
| P@2 | 0.350 | 0.340 | 0.350 | 0.270 |
| P@3 | 0.307 | 0.273 | 0.340 | 0.240 |
| P@4 | 0.285 | 0.245 | 0.300 | 0.230 |
| P@5 | 0.272 | 0.220 | 0.264 | 0.220 |
| P@6 | 0.247 | 0.207 | 0.243 | 0.210 |
| P@7 | 0.243 | 0.194 | 0.234 | 0.211 |
| P@8 | 0.230 | 0.182 | 0.233 | 0.193 |
| P@9 | 0.222 | 0.182 | 0.218 | 0.182 |
| P@10 | 0.214 | 0.176 | 0.206 | 0.178 |

**Table 5: Trec2003: P@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| NDCG@1 | 0.480 | 0.400 | 0.440 | 0.480 |
| NDCG@2 | 0.427 | 0.373 | 0.433 | 0.473 |
| NDCG@3 | 0.421 | 0.360 | 0.409 | 0.464 |
| NDCG@4 | 0.419 | 0.364 | 0.406 | 0.439 |
| NDCG@5 | 0.407 | 0.354 | 0.393 | 0.437 |
| NDCG@6 | 0.408 | 0.351 | 0.397 | 0.448 |
| NDCG@7 | 0.410 | 0.355 | 0.406 | 0.457 |
| NDCG@8 | 0.416 | 0.356 | 0.410 | 0.461 |
| NDCG@9 | 0.418 | 0.359 | 0.414 | 0.464 |
| NDCG@10 | 0.433 | 0.370 | 0.42 | 0.472 |

**Table 6: Trec2004: NDCG@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

| | RRLS | RLS | RankSVM | RankBoost |
|---|---|---|---|---|
| P@1 | 0.480 | 0.400 | 0.440 | 0.480 |
| P@2 | 0.400 | 0.347 | 0.407 | 0.447 |
| P@3 | 0.369 | 0.316 | 0.351 | 0.404 |
| P@4 | 0.347 | 0.293 | 0.327 | 0.347 |
| P@5 | 0.309 | 0.261 | 0.291 | 0.323 |
| P@6 | 0.291 | 0.238 | 0.273 | 0.304 |
| P@7 | 0.270 | 0.225 | 0.261 | 0.293 |
| P@8 | 0.253 | 0.210 | 0.247 | 0.277 |
| P@9 | 0.240 | 0.197 | 0.236 | 0.262 |
| P@10 | 0.235 | 0.197 | 0.225 | 0.253 |

**Table 7: Trec2004: P@n-performance comparison for RankRLS, RLS, RankSVM and RankBoost**

| Standard RLS | RankRLS |
|---|---|
| 0.444 | 0.500 |

**Table 8: Comparison of the parse ranking performances of standard RLS and RankRLS.**

performance similar to those of RankSVM and RankBoost, its computational advantages make it an attractive alternative.

## 3.2 Parse Ranking

We also evaluate the performance of RankRLS on the task of ranking of the parses of an unseen sentence. In these experiments, we use the BioInfer corpus [16] which consists of 1100 manually annotated sentences. A detailed description of the parse ranking problem and the data used in the experiments is given in [22]. Each parse is associated with a goodness score that indicates how close to the correct parse it is. As a similarity measure for parses, we use the best performing representation and graph kernel considered in [14]. The dimensionality of the feature space corresponding to the kernel function is large, and hence the dual RankRLS is preferable in this case. The ranking performance is measured with the Kendall's correlation coefficient $\tau_b$ [11]. It is considered a standard measure of rank correlation in cases where one is interested in complete ranking between the objects. Moreover, it takes into account ties when estimating correlation, which is crucial in parse ranking problems where the amount of tied parses can be large. The correlation is calculated separately for each sentence and the overall performance for a set of sentences is averaged.

First, we train a standard RLS regressor that we use as a baseline method. Its performance for correctly predicting the preference relations is evaluated afterwards. Next, we train RankRLS to regress the relevant output variable differences, that is, the differences between the goodness scores of parses that are generated from the same sentence.

Both RankRLS and RLS regressor have a regularization parameter $\lambda$ that controls the trade-off between the minimization of the training error and the complexity of the function. Further, the kernel function has parameters. In order to select the parameter values, we divide the set 1100 annotated sentences into two data sets containing 500 and 600 sentences. The first dataset is used for the parameter estimation and the second one is reserved for the final evaluation. The appropriate values of the regularization and the kernel parameters are determined by grid search with 10-fold cross-validation on the parameter estimation data. The parameter selection is performed separately for each experiment.

Finally, the algorithms are trained on the whole parameter estimation data set with the selected parameter values and tested with the sentences reserved for the final validation. The results of the validation are presented in Table 8. The results show that the RankRLS algorithm outperforms the standard RLS regressor.

## 4. CONCLUSION AND FUTURE WORK

In this paper, we propose a kernel based algorithm for learning preferences. The algorithm minimizes a regularized least-squares error of the output variable differences that are relevant to the task in question. We show that while the number output variable differences to be regressed grows quadratically with respect to the number of training data points, the algorithm can be trained as efficiently as a standard regularized least-squares regression for the individual outputs. Therefore, the dual version of the algorithm is particularly suitable in cases where the number of dimensions in the feature space is much larger than the number of training data points. As a representative example of such case we consider parse ranking task. We demonstrate that dual RankRLS notably outperforms the standard RLS regressor.

We also derive a primal version of the algorithm whose computational complexity mostly depends on the number of dimensions in the feature space rather than the number training data points. Therefore, this version is preferable when there are more data points than features. As a representative example of such case we choose the Letor information retrieval dataset for our experiments. The performance of the primal RankRLS is compared to that of RankSVM and RankBoost, two algorithms which can also be efficiently trained in such setting. We show that RankRLS achieves performance comparable to the 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.

In the future, we plan to test the capability of our method to optimize different types of ranking performance measures such as Wilcoxon-Mann-Whitney statistics and its generalizations. Moreover, we aim to evaluate a transductive version of RankRLS, that we introduced in [13], on information retrieval tasks. In our previous work [12], we have proposed a fast method for computing the hold out performance of an RLS learner which can then be used for computing cross-validation together with searching for the optimal regularization parameter efficiently. We plan to investigate if a similar kind of procedure can also be constructed for RankRLS.

## Acknowledgments

This work has been supported by Tekes, the Finnish Funding Agency for Technology and Innovation. We would like to thank CSC, the Finnish IT center for science, for providing us extensive computing resources and the anonymous reviewers for their insightful comments.

## 5. REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In J. Shawe-Taylor and Y. Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 624–638. Springer, 2004.

[3] U. Brefeld and T. Scheffer. Auc maximizing support vector learning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

[4] O. Dekel, C. Manning, and Y. Singer. Log-linear models for label ranking. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[5] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal Machine Learning Research*, 4:933–969, 2003.

[6] J. Fürnkranz and E. Hüllermeier. Preference learning. *Künstliche Intelligenz*, 19(1):60–61, 2005.

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

[8] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN99*, pages 97–102, London, 1999. Institute of Electrical Engineers.

[9] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, New York, NY, USA, 2000. ACM Press.

[10] T. 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.

[11] M. G. Kendall. *Rank Correlation Methods*. Griffin, 4. edition, 1970.

[12] T. Pahikkala, J. Boberg, and T. Salakoski. Fast n-fold cross-validation for regularized least-squares. In T. Honkela, T. Raiko, J. Kortela, and H. Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, pages 83–90, Espoo, Finland, 2006. Otamedia Oy.

[13] T. Pahikkala, H. Suominen, J. Boberg, and T. Salakoski. Transductive ranking via pairwise regularized least-squares. In P. Frasconi, K. Kersting, and K. Tsuda, editors, *Workshop on Mining and Learning with Graphs (MLG'07)*, 2007. To appear.

[14] T. Pahikkala, E. Tsivtsivadze, J. Boberg, and T. Salakoski. Graph kernels versus graph representations: a case study in parse ranking. In T. Gärtner, G. C. Garriga, and T. Meinl, editors, *Proceedings of the ECML/PKDD'06 workshop on Mining and Learning with Graphs (MLG'06)*, Berlin, Germany, 2006.

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

[16] S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50, 2007.

[17] A. Rakotomamonjy. Optimizing area under roc curve with svms. In *ROCAI*, pages 71–80, 2004.

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

[19] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In D. Helmbold and R. Williamson, editors, *COLT/EuroCOLT*, pages

416–426, Berlin, Germany, 2001. Springer-Verlag.

[20] B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[21] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[22] E. Tsivtsivadze, T. Pahikkala, S. Pyysalo, J. Boberg, A. Mylläri, and T. Salakoski. Regularized least-squares for parse ranking. In A. F. Famili, J. N. Kok, J. M. Peña, A. Siebes, and A. J. Feelders, editors, *IDA'05*, pages 464–474. Springer, 2005.