Large scale training methods for linear RankRLS

Antti Airola, Tapio Pahikkala and Tapio Salakoski

University of Turku, Department of Information Technology
Turku Centre for Computer Science (TUCS)
Finland
firstname.lastname@utu.fi

Abstract. RankRLS is a recently proposed state-of-the-art method for learning ranking functions by minimizing a pairwise ranking error. The method can be trained by solving a system of linear equations. In this work, we investigate the use of conjugate gradient and regularization by iteration for linear RankRLS training on very large and high dimensional, but sparse data sets. Such data is typically encountered for example in applications where natural language based data is used. We show that even though a pairwise loss function is optimized when training RankRLS, the computational complexity of the proposed methods, when learning from data with utility scores, is $O(tms)$, where t is the required number of iterations, m the number of training examples and s the average number of non-zero features per example. In addition, the complexity of learning from pairwise preferences is $O(tms+tl)$, where l is the number of observed preferences in the training set. In the experiments, it is further confirmed that restricting the number of conjugate gradient iterations has a regularizing effect and that the number of iterations that provides optimal results is, in practice, a small constant. Thus, the use of regularization by iteration, while providing similar performance as the more well-known Tikhonov regularization, results in a tremendous reduction in the computational cost of training and parameter selection.

1 Introduction

Learning to rank has been a topic of significant interest during the recent years. This development has been largely motivated by applications such as search engines and recommender systems. In these domains there are unprecedented quantities of data available, meaning that scalability is an essential property for machine learning algorithms for them to be useful in this setting.

A popular approach to modeling the ranking problem is to consider pairwise preferences. In this setting, the aim is to minimize the number of pairwise misorderings in the ranking produced when ordering a set of examples according to predicted utility scores. A number of machine learning algorithms optimizing relaxations of this criterion have been proposed [13, 15, 6, 3, 5, 21, 20]. In this work we consider large scale training algorithms for the RankRLS method [21, 20]. RankRLS is based on optimizing a least-squares approximation of the pairwise ranking error. The method can be seen as a modification of the regularized

least-squares method (RLS)[35, 24, 25], also known as the least-squares support vector machine [29]. Further, the minimized objective function is very similar to that of the ranking support vector machine (RankSVM)[13, 15], and the two methods have been experimentally shown to often achieve very similar ranking performance [21, 20].

An unique property of RankRLS, compared to other ranking algorithms, is that its solution can be expressed as a solution to a system of linear equations. As a result highly efficient and simple matrix calculus based training algorithms can be used to train the method as long as either the number of training examples, or dimensionality of the feature space is fairly small (i.e. few thousands or less) [21, 20]. Additionally, efficient matrix algebra based shortcuts can be derived for tasks such as fast regularization, cross-validation and multi-output learning [20], as well as feature selection, [19] as shown in our previous work.

In many central ranking applications, such as web search, the data to be ranked consists of natural language documents. In this domain it is typical to encounter very large and high dimensional, sparse data sets. The number of possible features that a document can have is often related to the number of distinct words in a vocabulary, or in n-gram models some power of this number. However, most features for a given document will be zero-valued, since only a small set of words actually appear in any give document. Most of the previously considered RankRLS training algorithms [21, 20] are not practical in this setting, since they are cubic either in the number of training examples, or in the number of distinct features. On such high dimensional data linear models can be expected to be already quite competitive, and this is to where we restrict our considerations in this article.

We have previously [20] noted that RankRLS can be trained in this setting using iterative conjugate gradient optimization. A similar approach has been proposed for regular RLS regression and classification [28, 25, 4]. The original RankRLS formulation relies on Tikhonov regularization [30], which is perhaps the most commonly used strategy for avoiding overfitting among regularized risk minimization based learners. This approach is compatible with conjugate gradient training. However, selection of the regularization parameter value requires an expensive grid search, and the convergence speed of the conjugate gradient method can be slow for small regularization parameter values.

Regularization has been a topic of intense research within the field of inverse problems, where an efficient alternative to Tikhonov regularization has been proposed for iterative gradient descent type of methods. Regularization by iteration, or early stopping, relies on stopping the optimization process before noise in the data begins to dominate the minimized objective function [11, 10]. In the field of machine learning, these results have recently gained renewed interest. For example, the framework of spectral regularization has been introduced for the purpose of deriving and analyzing efficient learning methods that regularize by early stopping [9].

In this work we introduce and experimentally evaluate different approaches to RankRLS training using large, high dimensional data sets. Both standard

Tikhonov regularization and regularization by iteration, as well as a hybrid approach are considered. Efficient optimization approaches based on sparse matrix operations and conjugate gradient optimization are presented. Experimental results verify that the proposed methods are suited for large-scale learning, and that regularization by iteration allows substantial decrease in computational costs compared to standard Tikhonov regularization by eliminating the need for parameter selection via grid searching.

2 Learning setting

Let D be a probability distribution over a sample space $\mathcal{Z} = \mathbb{R}^n \times \mathbb{R}$. An example $z = (\mathbf{x}, y) \in \mathcal{Z}$ is a pair consisting of an n-dimensional column vector of real-valued features, and an associated real-valued utility score. Let the sequence $Z = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in \mathcal{Z}^m$ drawn according to D be a training set of m training examples. $\mathbf{X} \in \mathbb{R}^{n \times m}$ denotes the $n \times m$ data matrix whose columns contain the feature representations of the training examples, and $\mathbf{y} \in \mathbb{R}^m$ is a column vector containing the utility scores in the training set. The utility scores reflect the "goodness" of the training examples with respect to the ranking criterion.

Let $\mathcal{I} = \{1 \ldots m\}$ denote the index set for the training set. In many application settings instead of having one global ranking over the training examples, the training set consists of several separate rankings. In this setting we assume that the indices are divided into a number of disjoint subsets $Q = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_q\}$ such that $\mathcal{Q}_i \subset \mathcal{I}$, $\bigcup_{i=1}^{q} \mathcal{Q}_i = \mathcal{I}$ and $\mathcal{Q}_i \cap \mathcal{Q}_j = \emptyset$, if $i \neq j$. We shall refer to these subsets as queries, a term that originates from the learning to rank for information retrieval setting, where the data sets consists of query-document pairs. Pairwise preferences exist only between examples from the same subset. In the following, the setting where we have a single global ranking over all the training examples should be interpreted as $Q = \{\mathcal{I}\}$.

Our task is to learn from the training data a ranking function $f : \mathbb{R}^n \to \mathbb{R}$. In the linear case such a function can be represented as $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters. Where the ranking task differs in the scoring setting from that of simple regression is that the actual values taken by the ranking function are typically not of interest. Rather, what is of interest is how well the ordering acquired by sorting a set of new examples according to their predicted scores matches the true underlying ranking. This is a reasonable criterion for example in the web search engines and recommender systems, where the task is to choose a suitable order in which to present web pages or products to the end user. A popular way to model this criterion is by considering the pairwise preferences induced by a ranking (see e.g. [7, 8]). We say that an example z_i is preferred over example z_j, if $y_i > y_j$, and z_i and z_j are comparable meaning that they belong to the same query. In this case one would require from the ranking function that $f(\mathbf{x}_i) > f(\mathbf{x}_j)$.

By restricting the allowed range of utility scores we can recover some popular special cases of the introduced setting. In ordinal regression (see e.g. [13, 34]) it

is assumed that there exists a finite, often quite small set of possible discrete ordinal labels. For example, movie ratings ranging from one star to five stars would constitute such a scale. In the bipartite ranking task where only two possible scores are allowed, the ranking task becomes equivalent to the task of maximizing the area under the ROC curve (AUC) [1, 31], a popular performance measure in binary classification.

The performance of a ranking function can be measured by the pairwise ranking error defined as

$$\frac{1}{|Q|} \sum_{\mathcal{Q}_i \in Q} \frac{1}{N_i} \sum_{j,k \in \mathcal{Q}_i, y_j < y_k} H(f(\mathbf{x}_j) - f(\mathbf{x}_k)), \tag{1}$$

where H is the Heaviside step function defined as

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

and N_i is the number of pairs in $\mathcal{Q}_i$ for which $y_j < y_k$ holds true. The equation (1) simply counts the number of swapped pairs between the true rankings and the one produced by f. For trivial predictors that assign predictions randomly, or give the same prediction to all examples, the resulting error is 0.5.

Additionally, we also consider the case where instead of utility scores, we are supplied only with pairwise preferences between data points. As an example, such data appears when gathering implicit user feedback from search engines [15]. A link that was clicked by a user can be considered to be preferred over those links that were not clicked. In this setting we still assume the transitivity of the underlying preference relation, meaning that any intransitivities occurring in the training data are treated as noise. For discussion on how to model and learn intransitive preference relations, we refer to the work of Pahikkala et al. [18].

Let $E = (e_1, \ldots, e_l)^{\mathrm{T}} \in (\mathcal{I} \times \mathcal{I})^l$ be a sequence of observed preferences between the inputs, that is, for each $e = (i, j)$, we say that x_i is preferred over x_j. Clearly, E can be considered as a preference graph in which the inputs are the vertices and the observed preferences are the edges. In this setting, the pairwise disagreement error may be defined more generally as

$$\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in E} H(f(\mathbf{x}_i) - f(\mathbf{x}_j)), \tag{2}$$

possibly normalized by the total number of considered pairs.

Minimizing (1) or (2) directly is computationally intractable, successful approaches to learning to rank according to the pairwise criterion typically minimize convex relaxations instead. In this work we consider the pairwise least-squares approximation, which leads to the RankRLS algorithm introduced in the next section.

3 RankRLS

In this section we formalize the empirical RankRLS risk, consider different regularization strategies and introduce an efficient learning algorithm for learning linear ranking functions from large sparse data sets. We first present an approach that is compatible with learning from utility scores, and then present an extension to learning directly from pairwise preferences.

3.1 Empirical risk

The empirical risk, which is the normalized pairwise least-squares loss over the training data, is formally defined as

$$\sum_{\mathcal{Q} \in Q} \frac{1}{2|\mathcal{Q}|} \sum_{i,j \in \mathcal{Q}} (y_i - y_j - \mathbf{x}_i^{\mathrm{T}}\mathbf{w} + \mathbf{x}_j^{\mathrm{T}}\mathbf{w})^2. \tag{3}$$

The choice of normalizer differs slightly from the one previously used by us [21, 20], the reason for the modification is that it simplifies considerably the following derivations.

Let

$$\mathbf{L}_h = \mathbf{I}_h - \frac{1}{h}\mathbf{1}_h(\mathbf{1}_h)^{\mathrm{T}}$$

be the $h \times h$-centering matrix with $h \in \mathbb{N}$, where $\mathbf{I}$ is the $h \times h$ identity matrix, and $\mathbf{1}$ a $h \times 1$ column vector of ones. The matrix $\mathbf{L}$ is a symmetric idempotent matrix and multiplying it with a vector removes the mean of the vector entries from all elements of the vector. Moreover, the following equality can be shown

$$\frac{1}{2h} \sum_{i,j=1}^{h} (c_i - c_j) = \mathbf{c}^{\mathrm{T}}\mathbf{L}_h\mathbf{c}\,,$$

where c_i are the entries of any vector $\mathbf{c}$. Without loss of generality, we can assume that the training data is ordered according to the queries, so that first come the examples belonging to the first query, next to the second, etc. Now, let us consider the following quasi-diagonal matrix:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{|\mathcal{Q}_1|} & & \\ & \ddots & \\ & & \mathbf{L}_{|\mathcal{Q}_q|} \end{pmatrix}.$$

The matrix $\mathbf{L}$ is again symmetric and idempotent, and can be interpreted as a query-wise centering matrix, that removes the mean from the prediction errors for each query. It is also a normalized version of the Laplacian matrix encoding the structure of the preference graph induced by the queries, which has been used in previous derivations of RankRLS [21, 20]. The matrix $\mathbf{L}$ can be decomposed as

$$\mathbf{L} = \mathbf{I} - \mathbf{P}\mathbf{P}^{\mathrm{T}} \tag{4}$$

where $\mathbf{P} \in \mathbb{R}^{m \times q}$ is a sparse matrix whose entries are defined as

$$\mathbf{P}_{i,j} = \begin{cases} \frac{1}{\sqrt{|\mathcal{Q}_j|}} & \text{if } i \in \mathcal{Q}_j \\ 0 & \text{otherwise} \end{cases} .$$

$\mathbf{P}$ has exactly m non-zero entries. Using this decomposition, the memory cost of storing $\mathbf{L}$, as well as the computational cost of multiplying a column vector with $\mathbf{L}$ are both $O(m)$. This forms the basis for developing the efficient RankRLS training algorithms presented in the following sections.

The empirical risk (3) can be re-written in matrix notation as

$$(\mathbf{X}^{\mathrm{T}}\mathbf{w} - \mathbf{y})^{\mathrm{T}}\mathbf{L}(\mathbf{X}^{\mathrm{T}}\mathbf{w} - \mathbf{y}) = (\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{w} - \mathbf{L}\mathbf{y})^{\mathrm{T}}(\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{w} - \mathbf{L}\mathbf{y}),$$

where the equality is due to the symmetry and idempotence of $\mathbf{L}$. What can be seen now is that by query-wise centering of the data matrix and label vector, we have transformed the empirical risk term to a standard least-squares form, equivalently expressed as

$$\| \mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{w} - \mathbf{L}\mathbf{y} \|^2, \tag{5}$$

whose minimizer is the best fit, in the least-squares sense, to the equation

$$\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{w} = \mathbf{L}\mathbf{y}.$$

This connection between standard least-squares and RankRLS means that in theory, we could center the data matrix and the label vector, and proceed by using known training algorithms for (regularized) least-squares. However, as we will discuss next in more detail, this would destroy the sparse structure of the data matrix, meaning that methods would not scale to large but sparse data sets.

3.2 Tikhonov regularization

Solving algebraic systems of linear equations has been studied extensively in the field of inverse problems, and it is a well known result that problems of this form are often ill-conditioned. This idea has been directly linked to the phenomenon of over-fitting in machine learning, where it has been noticed that pure empirical risk minimization can lead to over-fitting, where the solution models the noise in the data rather than the true underlying concept. The problem is usually addressed by means of regularization.

By far the most popular approach has been Tikhonov regularization [30], where a regularization term, typically of the form $\lambda \| \mathbf{w} \|^2$, is added to the minimization problem. λ known as the regularization parameter is a positive constant that controls the tradeoff between the data fit and model complexity. A large body of machine learning literature is based on this choice, starting from the works of Vapnik [32, 33], Wahba [35] and Poggio and Girosi [24]. The family of regularized risk minimization based learning algorithms includes methods such as RLS [35, 24, 25], support vector machines [33], RankSVM [13, 15] and the

original RankRLS method [21, 20], among others. For RankRLS, the regularizer guarantees the existence of an unique minimizer, which can be recovered from

$$\mathbf{w} = (\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}} + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{L}\mathbf{y} \tag{6}$$

(for proof, see [21]). Also, based on the matrix equivalences of Henderson and Searle [12] and the idempotence of $\mathbf{L}$, it can be seen that the RankRLS solution can be equivalently obtained from

$$\mathbf{w} = \mathbf{X}\mathbf{L}(\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{L} + \lambda\mathbf{I})^{-1}\mathbf{L}\mathbf{y}. \tag{7}$$

Using the sparse decomposition of $\mathbf{L}$ (4), $\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$ can be calculated in $O(mn^2)$ time and $\mathbf{L}\mathbf{y}$ in $O(m)$ time. The overall complexity of solving (6) is $O(mn^2+n^3)$ and of solving (7) $O(m^2n+m^3)$.

In practice direct matrix inversion is not the recommended strategy for solving RankRLS. Instead algorithms based on matrix decompositions, such as the singular value decomposition of the data matrix, are recommended. These approaches are faster than direct matrix inversion by a constant factor, and allow efficient search for the value of the regularization parameter λ [20]. Additionally, efficient computational shortcuts are known for calculating statistics, such as the leave-query-out, or the leave-pair-out ranking error and parallel learning of multiple models simultaneously [20], as well as for feature selection [19].

These training algorithms that are based on dense linear algebra are practical as long as we can guarantee that either m of n is small. For example the first is often the case in life sciences, where the data may be high dimensional measuring thousands of genes, but the sample size is small due to the costs of administering expensive tests on real human patients. Conversely, in much of the recent learning to rank research in information retrieval the considered data sets have consisted of tens to hundreds of thousands of examples, but only tens of high-level features. In both of these settings the dense linear algebra based training algorithms for RankRLS are highly competitive. However, for large and high dimensional, but sparse data sets both (6) and (7) are impractical, since they require quadratic memory storage and cubic computational time either with respect to m or n.

3.3 The conjugate gradient method

The conjugate gradient method [14, 26] provides a powerful iterative method for solving sparse systems of linear equations. The method is well suited sparse equations, since it relies only on repeated multiplication of a vector with the sparse matrices. Further, it has minimal storage requirements and provably converges much faster than regular gradient descent. Previously, Suykens et al. [28] and Chu et al. [4] have proposed the method for non-linear RLS training with kernels, Rifkin et al. [25] noted that conjugate gradient has many additional benefits when training linear RLS classifiers, and in our previous work [20] we have noted the suitability of conjugate gradient for large scale linear RankRLS training, and performed some runtime experiments.

Conjugate gradient is a method for solving equations of the type $\mathbf{Ab} = \mathbf{c}$, where $\mathbf{A}$ is symmetric and positive definite. The equation

$$(\mathbf{X}\mathbf{X}^{\mathrm{T}} - \mathbf{X}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}(\mathbf{y} - \mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{y}). \tag{8}$$

for finding the RankRLS solution, fulfills this criterion. Note that we have replaced $\mathbf{L}$ with $\mathbf{I} - \mathbf{P}\mathbf{P}^{\mathrm{T}}$ to facilitate efficient computation that preserves the sparsity of the data. None of the matrix products on the left side are explicitly calculated, but rather the presented decomposed form is used as such when multiplying it with a vector.

On each iteration of conjugate gradient training, the most expensive operation is the multiplication of the matrix on the left side of (8) to a column vector $\mathbf{d}_t \in \mathbb{R}^n$, that changes each iteration. This can be calculated as $\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{d}_t - \mathbf{P}(\mathbf{P}^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{d}_t))) + \lambda\mathbf{d}_t$, where the parentheses are used to denote the order in which the multiplications are carried out in. Only sparse matrix - vector multiplications are needed. Since $\mathbf{X}$ has ms, and $\mathbf{P}$ m non-zero entries, the whole sequence of operations can be calculated in $O(ms)$ time. The overall complexity of training RankRLS with conjugate gradient is thus $O(tms)$ where t is the required number of iterations.

A basic result about the convergence of conjugate gradient is that t is bounded by the number of distinct eigenvalues of $\mathbf{A}$ [26]. The maximum rank of $\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$ gives us an upper bound on the number of required iterations, since the matrix has at most as many distinct eigenvalues as its rank is, and the regularization term will at most add one new eigenvalue. Thus $t \leq \min\{m+1, n+1\}$. This means that conjugate gradient method is guaranteed to converge very rapidly if either sample size or dimensionality of the data is low. However, for very large sparse data matrices this does not guarantee efficiency, since both m and n may be very large.

Let $\frac{\gamma_{max}}{\gamma_{min}}$ be the condition number of the matrix $\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$, where γ_{max} and γ_{min} are the largest and smallest eigenvalue of $\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$, respectively. By examining the eigen decomposition of this matrix, we can verify that the condition number of the matrix $\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}} + \lambda\mathbf{I}$, which is equivalent to the one present on left side of (8), is $\kappa = \frac{\gamma_{max}+\lambda}{\gamma_{min}+\lambda}$. The maximum number of iterations required by the conjugate gradient method to reduce the norm of the error by a factor of ϵ is $t \leq \frac{1}{2}\sqrt{\kappa}\ln(\frac{2}{\epsilon})$ [26]. Because κ approaches one from above as regularization is increased, the bound suggests that speed of convergence depends inversely on the size of the regularization parameter. In the experiments we observed this to be the case, with large enough values of λ the conjugate gradient method needed finally only one iteration to converge. On the other hand, for very small values of λ the number of iterations required can be very large. Additionally, a problem is how to select the correct value of λ in the first place. Unlike when using the dense linear algebra based training algorithms, no computational shortcuts exist for fast evaluation of a suitable value of λ. Rather, brute force grid search using cross-validation or a separate validation set is needed.

3.4 Regularization by iteration

In numerical optimization literature, a more efficient alternative to using Tikhonov type of regularization is known. The approach is known as regularization by iteration, or early stopping. The technique originated with Landweber iteration [16], which corresponds to a steepest-descent minimization of least-squares equations, such as (5). The basic intuition of this approach is, that the noise in the training data begins to influence the convergence of gradient descent type of methods only during the later iterations. In the first iterations the phenomenon of semi-convergence is observed, at first the methods converge towards a good solution, but after a certain point begin to diverge, as the noise starts to dominate the objective function.

The regularizing effect of early stopping is also known to apply to the conjugate gradient method. Hanke and Hansen [11] discussed different regularization methods for solving large ill-conditioned algebraic systems of linear equations, and recommended the use of conjugate gradient with regularization by iteration. A clear exposition of many of the considered ideas, together with further empirical results can be found in the work of Hanke [10]. In machine learning, the regularizing effect of early stopping is an old well-known result in neural networks literature (see e.g. [27]). Recently, there has been a renewed interest in the use of early stopping for training kernel based learning algorithms. Yao et al. [36] discuss using gradient descent with early-stopping for regularized least-squares, and also discuss the connections of this approach to boosting. Gerfo et al. [9] introduce the framework of spectral regularization for machine learning, which gives rise to regularized learning algorithms based on gradient descent type of optimization algorithms, that regularize by early stopping. Spectral regularization based classifiers were shown to be efficient, and in terms of performance to compare favorably to SVMs and AdaBoost.

Formally, in regularization by iteration we apply t iterations of conjugate gradient for solving approximately

$$(\mathbf{X}\mathbf{X}^{\mathrm{T}} - \mathbf{X}\mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}})\mathbf{w} = \mathbf{X}(\mathbf{y} - \mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{y}).$$

As previously, the cost of each iteration is $O(ms)$. However, early stopping offers two advantages in terms of efficiency. First, since the optimization is terminated early, the required number of iterations t is typically quite small. Second, one does not have to choose the correct value of λ, since t itself acts as the regularization parameter. A problem is how to correctly choose t, since this has a major impact on performance. The most natural choice is to measure performance on an independent validation set, and terminate optimization once no improvement has been seen for a specified number of iterations, retaining the best solution seen before termination. Results in the literature [11, 10, 9] suggest that regularization by iteration should, in terms of performance, be competitive with Tikhonov regularization.

Finally, Hanke and Hansen [11] and Gerfo et al. [9] discuss also hybrid methods, that combine Tikhonov regularization and early stopping. While one may still have to re-start the optimization many times for different values of λ, this

offers some computational benefits, since running conjugate gradient until it converges can require a substantial number of iterations for small values of λ.

3.5 Learning from pairwise preferences

When supplied with only pairwise comparisons between the data points, the empirical risk based on pairwise least-squares loss can be defined as

$$\sum_{(i,j)\in E} (1 - \mathbf{x}_i^{\mathrm{T}}\mathbf{w} + \mathbf{x}_j^{\mathrm{T}}\mathbf{w})^2. \tag{9}$$

Alternatively, as discussed in [20], if we have also information about the magnitudes of the pairwise preferences, we can replace the constant 1 in (9) with this information. In the following we assume that we do not have access to such information.

Let $\mathbf{M} \in \mathbb{R}^{m\times l}$ be a matrix whose rows and columns are indexed by the vertices and edges of the preference graph for the training set, and its entries are given by

$$\mathbf{M}_{i,h} = \begin{cases} 1 & \text{if } e_h = (i,j), \text{ for some } j \neq i \\ -1 & \text{if } e_h = (j,i), \text{ for some } j \neq i \\ 0 & \text{otherwise} \end{cases}.$$

In the graph theory, this matrix is sometimes called the oriented incidence matrix of a graph (see e.g. [2]).

The empirical risk can be re-written in matrix notation as

$$(\mathbf{1} - \mathbf{M}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{w})^{\mathrm{T}}(\mathbf{1} - \mathbf{M}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{w}),$$

where $\mathbf{1}$ is a length l column vector of ones. This corresponds again to a least-squares minimization problem, which, when combined with Tikhonov regularization, can be solved by applying the conjugate gradient method on the equation

$$(\mathbf{X}\mathbf{M}\mathbf{M}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}\mathbf{M}\mathbf{1}$$

The computational cost of the matrix-vector multiplications required in each iteration is $O(ms + l)$, which is the number of non-zero entries in matrices $\mathbf{X}$ and $\mathbf{M}$. Thus the approach scales linearly in the number of examples, non-zero features and pairwise preferences. Again, we can perform regularization by iteration by setting $\lambda = 0$ and by stopping the conjugate gradient method after a suitable number of iterations t, where the value of t is chosen on independent validation data.

Should we transform scored data into pairwise preferences and use label differences as preference magnitudes, the approach presented here would be equivalent to the approach we use for scored data, ignoring the query-wise normalizations. When learning from a single global ranking l grows quadratically with respect to m, leading to $O(ms + m^2)$ iteration cost, which can be too much on very large data sets. For data with query structure this approach would be more manageable, as l grows quadratically with respect to query size, but only linearly with respect to the number of queries.

4 Experimental results

In the experiments we measure the computational efficiency and performance of different RankRLS training algorithms. We compare a training algorithm that uses Tikhonov regularization, a training algorithm that relies on early stopping, and a hybrid method that combines these approaches. In the hybrid approach we choose the value of the regularization parameter as in regular Tikhonov regularization, but terminate the optimization as soon as validation error has not decreased in ten consecutive iterations In addition to training time we measure the parameter selection time, since in practice one cannot know in advance the correct values of λ and t. We do not consider training algorithms based on dense linear algebra, since they would not scale to the considered data set sizes.

The learning algorithms were implemented as part of the RLScore open source machine learning framework [1]. Scientific Tools for Python (SciPy) sparse matrix and conjugate gradient method implementations were used in the implementations. The experiments were run on a modern desktop computer with 2.4 GHz Intel Core 2 Duo E6600 processor, 8 GB of main memory, and 64-bit Ubuntu Linux 9.10 operating system.

We consider two ranking problems. The characteristics of both the considered data sets are presented in Table 1. Both data sets are divided into three parts, of which the validation set is used for selecting the regularization parameter λ, and in regularization by iteration to select when to stop.

The parse ranking problem comes from the field of natural language processing. Due to the ambiguity of natural language, automatic syntactic parsers often produce a large number of syntactically valid candidate parses, when given a sentence as an input. In parse ranking, the task is given a sentence and a set of candidate parses, to rank them according to how well they match the true (at test time unknown) parse for the sentence. The data set is the same as used in [22], and consists of 12000 examples, which are joint feature representations of sentences and parses. The feature set is a linearization of the graph kernel described in [22], resulting in a sparse but very high dimensional feature representation. The labels in the data are real valued utility scores, and instead of a single global ranking, groups of candidate parses related to the same sentence constitute queries.

The other considered problem is bipartite ranking (or equivalently, AUC-maximization), on a text classification task. The data set is constructed from the Reuters RCV1 collection [17], and the task is to rank higher the documents in the CCAT category, than documents not belonging to this category. The features consist of TF/IDF values. The data set consists of close to eight hundred thousand examples, and is also very high dimensional and sparse.

Finally, we run a third experiment to test the behavior of RankRLS when learning directly from pairwise preferences rather than from data with utility scores. Since we are not aware of any very large datasets consisting of pairwise

[1] Available at http://www.tucs.fi/RLScore

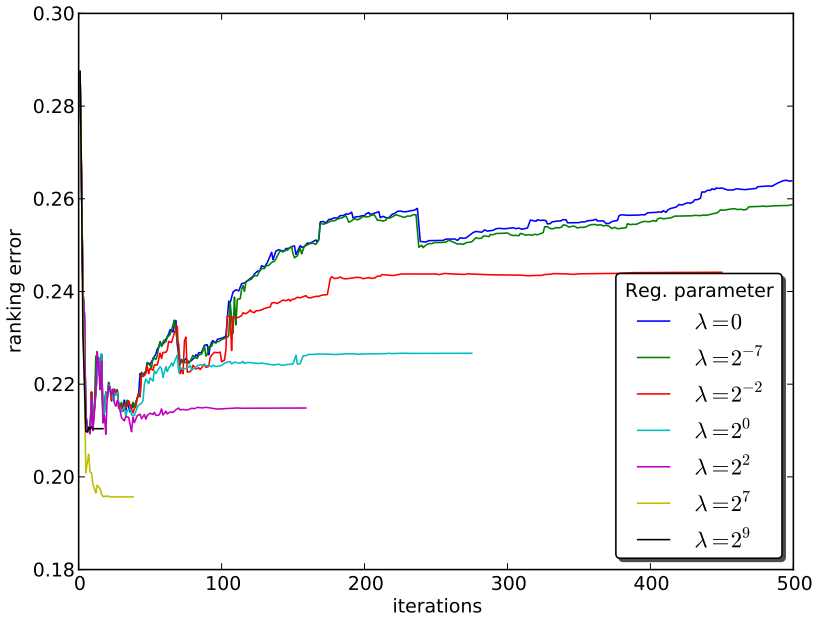

Fig. 1. Ranking error on the parse ranking validation data.

preferences, we create such data by taking a random sample of 10^6 positive-negative example pairs from the Reuters data set.

First, we study the convergence behavior of conjugate gradient trained RankRLS. We plot for different regularization parameter values the pairwise ranking error on the validation set as a function of performed iterations. The termination criterion is $\epsilon < 10^{-5}$. In Figure 1 are the results for the parse ranking task, in Figure 2 on the Reuters data when learning from utility scores. The results for Reuters data when learning from pairwise preferences look essentially the same as when learning from utility scores, so we do not present them separately. The parse ranking and Reuters plots are very similar. Runs with zero or close to zero regularization achieve good performance at first, but begin then to overfit to the noise, confirming the regularizing effect of early stopping. As the value of λ is increased the methods converge faster. If optimization is run until convergence, it is crucial that the value of λ is chosen correctly. Too small values overfit, and too large values underfit, leading to notably decreased performance.

Next we compare three approaches to RankRLS training. Method 1 selects the value of λ with a grid search on the validation set. Optimization is either run until the termination criterion $\epsilon < 10^{-5}$ is fulfilled, or 500 iteration limit reached. Method 2 relies solely on regularization by iteration, with $\lambda = 0$. The method keeps in memory the solution that has had the lowest validation error

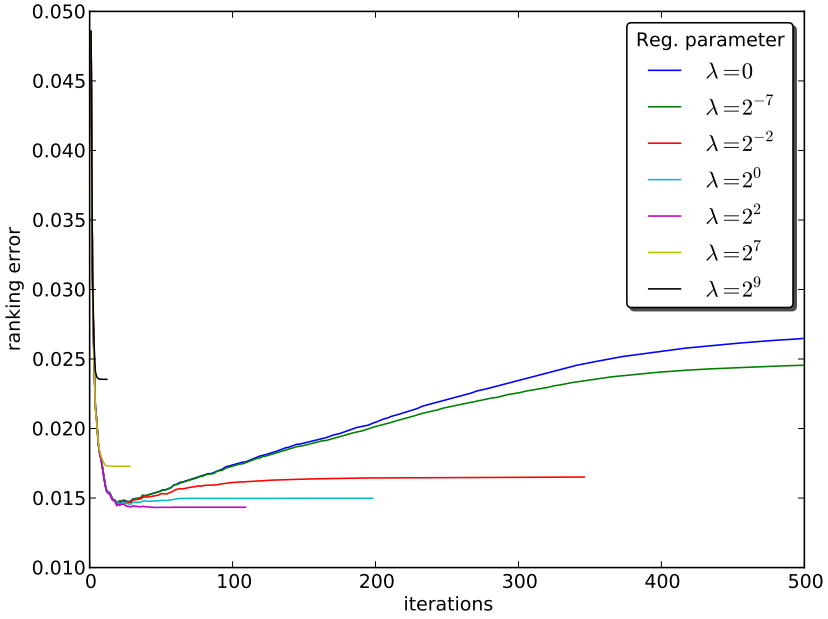

Fig. 2. Ranking error on the Reuters validation data.

this far, and once no improvements have been seen for 10 consecutive iterations, the method terminates. Method 3 is a hybrid approach that combines the first and the second method. Methods 1 and 3 select the regularization parameter from the grid $\{2^{-10} \dots 2^{10}\}$.

In table 1 are the results for all the tasks. The method that runs conjugate gradient optimization with Tikhonov regularization until convergence, and the hybrid method that combines this with early stopping achieve essentially the same performance. Since using early stopping here already results in a tenfold reduction in parameter selection time, we conclude that this approach can yield substantial computational savings when combined with normal Tikhonov regularization. The savings are much more dramatic for method 2, that uses only regularization by iteration, with no separate regularization term. Even on close to million training examples on the Reuters data, the method takes only about one and a half minute to train. The results for regularization by iteration are slightly better than the two other approaches on the parse ranking data, and slightly worse on the Reuters data. On Reuters data the performance is lower when learning from the sample of pairwise preferences than when learning from the utility scores. This is not surprising, since the formulation of RankRLS that learns from utility scores implicitly considers all pairwise preferences in the train-

Name	Train size	Val size	Test size	Features	Sparsity
Parse ranking	7539	2497	2354	195100	0.27%
Reuters	781265	8149	23149	47152	0.16%

Method	Test error	Training time	Param. select. time
1	0.2185	11 s	1743 s
2	0.2180	6 s	9 s
3	0.2185	7 s	137 s

Method	Test error	Training time	Param. select. time
1	0.01472	352 s	17186 s
2	0.01531	65 s	97 s
3	0.01473	155 s	2120 s

Method	Test error	Training time	Param. select. time
1	0.01530	226 s	15295 s
2	0.01621	52 s	78 s
3	0.01530	94 s	1695 s

Table 1. The topmost table contains the data set characteristics. The next two tables contain the results for Parse ranking and Reuters when learning from scored data. The lowermost table contains the results for Reuters, when learning from a sample of 10^6 pairwise preferences. In the three result tables, the methods 1, 2, and 3 refer to conjugate gradient with Tikhonov regularization, regularization by iteration, and combination of Tikhonov regularization and early stopping, respectively.

ing set, whereas only a subsample of 10^6 pairwise preferences is used for training the pairwise formulation of the method.

5 Conclusion

In this article we have proposed large-scale training algorithms for learning linear ranking functions from very large sparse data sets. The minimizer of the RankRLS loss can be found as a solution to a system of linear equations, allowing the use of conjugate gradient optimization. This leads to a training algorithm that is both fast and simple to implement. In addition to using standard approach based on Tikhonov regularization, we have considered the merits of early stopping. Experimental results show that regularization by iteration is a viable alternative to using Tikhonov regularization, as it leads to competitive performance and considerable computational savings.

Though we have in this work limited our considerations to linear RankRLS training, the considered approaches are suitable also for training the kernel version of RankRLS. Explicit construction of the full kernel matrix is not required, as long as an efficient way to calculate the kernel matrix - vector products exist. Such computational shortcuts are available for example for pairwise Kronecker product kernels used in learning intransitive preference relations [18], and more generally in conditional ranking [23].

References

1. Bamber, D.: The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. Journal of Mathematical Psychology 12, 387–415 (1975)
2. Brualdi, R.A., Ryser, H.J.: Combinatorial Matrix Theory. Cambridge University Press (1991)
3. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Raedt, L.D., Wrobel, S. (eds.) Proceedings of the 22nd international conference on Machine learning (ICML 2005). pp. 89–96. ACM, New York, NY, USA (2005)
4. Chu, W., Ong, C.J., Keerthi, S.S.: An improved conjugate gradient scheme to the solution of least squares SVM. IEEE Transactions on Neural Networks 16, 498 – 501 (2005)
5. Cortes, C., Mohri, M., Rastogi, A.: Magnitude-preserving ranking algorithms. In: Ghahramani, Z. (ed.) Proceedings of the 24th Annual International Conference on Machine Learning. ACM International Conference Proceeding Series, vol. 227, pp. 169–176. ACM Press, New York, NY, USA (2007)
6. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4, 933–969 (2003)
7. Fürnkranz, J., Hüllermeier, E.: Pairwise preference learning and ranking. In: Lavrac, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) Proceedings of the 14th European Conference on Machine Learning (ECML 2003). Lecture Notes in Computer Science, vol. 2837, pp. 145–156. Springer (2003)
8. Fürnkranz, J., Hüllermeier, E.: Preference learning. Künstliche Intelligenz 19(1), 60–61 (2005)
9. Gerfo, L.L., Rosasco, L., Odone, F., Vito, E.D., Verri, A.: Spectral algorithms for supervised learning. Neural Computation 20(7), 1873–1897 (2008)
10. Hanke, M.: Iterative regularization techniques in image reconstruction. In: Colton, D., Engl, H.W., Louis, A.K., McLaughlin, J.R., Rundell, W. (eds.) Surveys on Solution Methods for Inverse Problems. pp. 35–52. Springer (2000)
11. Hanke, M., Hansen, P.C.: Regularization methods for large-scale problems. Surveys on Mathematics for Industry 3, 253–315 (1993)
12. Henderson, H.V., Searle, S.R.: On deriving the inverse of a sum of matrices. SIAM Review 23(1), 53–60 (1981)
13. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: Proceedings of the Ninth International Conference on Articial Neural Networks (ICANN 1999). pp. 97–102. Institute of Electrical Engineers, London (1999)
14. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49(6), 409–436 (1952)
15. Joachims, T.: Optimizing search engines using clickthrough data. In: Hand, D., Keim, D., Ng, R. (eds.) Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2002). pp. 133–142. ACM Press, New York, NY, USA (2002)
16. Landweber, L.: An iteration formula for Fredholm integral equations of the first kind. American Journal of Mathematics 73(3), 615–624 (1951)
17. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)

18. Pahikkala, T., Waegeman, W., Tsivtsivadze, E., Salakoski, T., De Baets, B.: Learning intransitive reciprocal relations with kernel methods. European Journal of Operational Research 206, 676–685 (2010)
19. Pahikkala, T., Airola, A., Naula, P., Salakoski, T.: Greedy RankRLS: a linear time algorithm for learning sparse ranking models. In: Gabrilovich, E., Smola, A.J., Tishby, N. (eds.) SIGIR 2010 Workshop on Feature Generation and Selection for Information Retrieval (2010)
20. Pahikkala, T., Tsivtsivadze, E., Airola, A., Boberg, J., Järvinen, J.: An efficient algorithm for learning to rank from preference graphs. Machine Learning 75(1), 129–165 (2009)
21. Pahikkala, T., Tsivtsivadze, E., Airola, A., Boberg, J., Salakoski, T.: Learning to rank with pairwise regularized least-squares. In: Joachims, T., Li, H., Liu, T.Y., Zhai, C. (eds.) SIGIR 2007 Workshop on Learning to Rank for Information Retrieval. pp. 27–33 (2007)
22. Pahikkala, T., Tsivtsivadze, E., Boberg, J., Salakoski, T.: Graph kernels versus graph representations: a case study in parse ranking. In: Gärtner, T., Garriga, G.C., Meinl, T. (eds.) Proceedings of the ECML/PKDD'06 workshop on Mining and Learning with Graphs (MLG'06). Berlin, Germany (2006)
23. Pahikkala, T., Waegeman, W., Airola, A., Salakoski, T., De Baets, B.: Conditional ranking on relational data. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2010) (2010)
24. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9) (1990)
25. Rifkin, R., Yeo, G., Poggio, T.: Regularized least-squares classification. In: Suykens, J., Horvath, G., Basu, S., Micchelli, C., Vandewalle, J. (eds.) Advances in Learning Theory: Methods, Model and Applications, NATO Science Series III: Computer and System Sciences, vol. 190, chap. 7, pp. 131–154. IOS Press, Amsterdam (2003)
26. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. rep. (1994)
27. Sjöberg, J., Ljung, L.: Overtraining, regularization, and searching for minimum in neural networks. In: IFAC Symposium on Adaptive Systems in Control and Signal Processing. pp. 669–674 (1992)
28. Suykens, J.A.K., Lukas, L., Dooren, P.V., Moor, B.D., Vandewalle, J.: Least squares support vector machine classifiers: a large scale algorithm. In: Proceedings of the European Conference on Circuit Theory and Design (1999)
29. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters 9(3), 293–300 (1999)
30. Tikhonov, A.N., Arsenin, V.Y.: Solutions of ill posed problems. V. H. Winston and Sons (1977)
31. Vanderlooy, S., Hüllermeier, E.: A critical analysis of variants of the AUC. Machine Learning 72(3), 247–262 (2008)
32. Vapnik, V.: Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow (1979), (English translation: Springer, New York, 1982)
33. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA (1995)
34. Waegeman, W., De Baets, B., Boullart, L.: ROC analysis in ordinal regression learning. Pattern Recognition Letters 29, 1–9 (2008)
35. Wahba, G.: Spline Models for Observational Data. Society for Industrial and Applied Mathematics (1990)
36. Yao, Y., Rosasco, L., Caponetto, A.: On early stopping in gradient descent learning. Constructive Approximation 6(2), 289–315 (2007)