Transductive Ranking via Pairwise Regularized Least-Squares

Tapio Pahikkala Hanna Suominen Jorma Boberg Tapio Salakoski
Turku Centre for Computer Science (TUCS)
University of Turku, Department of Information Technology
Joukahaisenkatu 3-5 B, 20520 Turku, Finland
firstname.lastname@utu.fi

1. INTRODUCTION

Ranking data points with respect to a given preference criterion is an example of a preference learning task. Tasks of this kind are often considered 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 (see, e.g., [5, 6]). Following the terminology of [4], our focus is on label ranking, that is, each data instance is associated with a set of labels that we aim to rank in order of a certain utility scoring. The data instance can be, for example, a query of a web search engine, and the label set consists of the documents obtained by the query. The utility score of a document would then its relevance to the query.

The paper is organized as follows. In Section 2, we propose a ranking algorithm which is based on minimizing the regularized least-squares (RLS) error (see, e.g., [9]) of the score differences. The information regarding the score differences that the algorithm is supposed to learn is stored in a graph defined for the training set. In Section 3, we first introduce an efficient method for computing hold-out estimates for the proposed algorithm. Finally, using the hold-out method, we propose a transductive version of the algorithm.

2. PAIRWISE LEAST-SQUARES

In [8], we proposed the RankRLS learner for a general purpose preference learning. The learner minimizes the RLS error of the output variable differences. We proved that training of the learner has the same computational complexity as training of the standard RLS regressor for the same data set. We compared the RankRLS learner with the standard RLS regressor in the task of dependency parse ranking and found that for this task, the ranking performance of RankRLS is significantly better than that of the standard RLS regressor.

In this section, we continue the study by specifying the RankRLS for label ranking. We use the term input variable, or simply input, to refer to a pair comprised of an instance and one of its associated labels. At first, we construct a training set from input variables and scores corresponding to a given set of data instances. Then, we define an undirected graph whose vertices are the training inputs. Two vertices are connected with an edge if they correspond to the same data instance, and hence the graph consists of a set of isolated complete subgraphs. For example, in the graph generated for the web search queries and documents, two vertices are connected if their corresponding documents are obtained by the same query. The Laplacian matrix of the graph is used to encode the connection information into the algorithm. Finally, we show that while the number of possible score differences grows quadratically with respect to the number of training inputs, RankRLS is as efficiently trained as the standard RLS regressor for the individual score values.

Let the input space $\mathcal{X}$ be the set of possible input variables and let $\mathbb{R}$ be the set of scores. We call the set of possible input-score pairs $\mathcal{Z} = \mathcal{X} \times \mathbb{R}$ the example space. Further, the term instance refers to a set of examples in this context and hence the instance space $\mathcal{V}$ is defined to be the family of all finite subsets of $\mathcal{Z}$.

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. To measure how well a hypothesis $f \in \mathcal{H}$ is able to rank the inputs of the instances, we consider the following definition of the disagreement error which is similar to the one defined in [3]:

$$\mathcal{E}(v, f) = \frac{1}{2}\binom{|v|}{2}^{-1} \sum_{z,z' \in v} d(y - y', f(x) - f(x')) \qquad (1)$$

where $v \in \mathcal{V}$; $x, x' \in \mathcal{X}$; $y, y' \in \mathbb{R}$; $z = (x, y), z' = (x', y')$;

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

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 constant $\frac{1}{2}\binom{|v|}{2}^{-1}$ ensures that the disagreement error is between 0 and 1. The direction of preference of a data point pair (z, z') is determined by $\operatorname{sign}(y - y')$. Similarly, the predicted direction of preference is determined by $\operatorname{sign}\big(f(x) - f(x')\big)$.

Our formulation of this type of a preference learning task is: find a function $f \in \mathcal{H}$ that minimizes the expectation of the disagreement error. Of course, we do not usually know the distribution of the instances. Instead, we are given a finite number q of instances generated by some unknown distribution. From the given instances, we take every input variable, say altogether m inputs, and define $X = (x_1, \ldots, x_m) \in (\mathcal{X}^m)^{\mathrm{T}}$ to be a sequence of inputs, where $(\mathcal{X}^m)^{\mathrm{T}}$ denotes the set of row vectors whose elements belong to $\mathcal{X}$. Analogously, we define $Y = (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathbb{R}^m$ be a sequence of the corresponding score values. We also denote $z_i = (x_i, y_i)$, $1 \leq i \leq m$. Thus, $m = \sum_{l=1}^{q} |v_l|$, where $|v_l|$ is the number of examples in the lth instance. To keep an account to which instance each example belongs, we define $U_l \subseteq \{1, \ldots, m\}$, where $1 \leq l \leq q$, to be the index set whose elements refer to the indices of the examples that belong to the lth instance v_l. Of course, $U_l \cap U_{l'} = \emptyset$ if $l \neq l'$. Next, we define an undirected weighted graph for the training data whose vertices are the indices of all the inputs of all the instances given for the training examples $z_i, 1 \leq i \leq m$. The graph is determined by the adjacency matrix W whose elements are

$$W_{i,j} = \begin{cases} \binom{|v|}{2}^{-1} & \text{when } i,j \in U_l \wedge i \neq j \\ 0 & \text{otherwise} \end{cases}.$$

We observe that the graph W consists of a set of isolated complete subgraphs corresponding to the different instances. Altogether, we define the training set to be the triple $S = (X, Y, W)$.

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 the training phase and are not able to generalize to unseen data. Following [10], we consider the framework of regularized kernel methods in which $\mathcal{H}$ is so-called reproducing kernel Hilbert space defined by a positive definite kernel function k. Then, the learning algorithm that selects the hypothesis f from $\mathcal{H}$ is defined as

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

where

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

$f(X) = (f(x_1), \ldots, f(x_m))^{\mathrm{T}}$, c is a real valued cost function, $\lambda \in \mathbb{R}_+$ is the regularization parameter, and $\|\cdot\|_k$ is the norm in $\mathcal{H}$. By the generalized representer theorem [10], the minimizer of (2) has the following form:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i), \tag{3}$$

where $a_i \in \mathbb{R}$ and k is the kernel function associated with the reproducing kernel Hilbert space mentioned above. For the training set, we define the symmetric $m \times m$ kernel matrix K to be a matrix whose elements are $K_{i,j} = k(x_i, x_j)$. For simplicity, we also assume that K is a strictly positive definite matrix. This can be ensured, for example, by performing a small diagonal shift. Using this notation, we rewrite $f(X) = KA$ and $\|f\|_k^2 = A^{\mathrm{T}}KA$, where $A = (a_1, \ldots, a_m)^{\mathrm{T}}$.

A natural way to encode the preference information into a cost function is to use the disagreement error (1) for each pair of training examples. Formally,

$$c(f(X), Y, W) = \frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} d(y_i - y_j, f(x_i) - f(x_j)). \tag{4}$$

The weighting with W ensures that each instance has an equal weight in the cost funtion. It is well-known that this type of cost functions lead to intractable optimization problems. Therefore, instead of using (4), we use functions approximating it. Namely, we adopt 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 $f(x_i) - f(x_j)$:

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

Before presenting the solution for the minimization problem using the least-squares approximation, we introduce some notation used. Let L be the Laplacian matrix (see, e.g., [1]) of the graph W. Its entries are defined by

$$L_{i,j} = \begin{cases} \sum_{j=1}^{m} W_{i,j} & \text{when } i = j \\ -W_{i,j} & \text{otherwise} \end{cases}.$$

The next theorem characterizes a method we call RankRLS.

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

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

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 (3), 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 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 is a diagonal matrix whose entries are defined as $D_{i,i} = \sum_{j=1}^{m} W_{i,j}$, and $L = D - W$ is 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

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

and hence the algorithm (5) is rewritten as

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

where

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

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 standard RLS regression.

3. TRANSDUCTIVE LEARNING

In this section, we first introduce an efficient way to compute hold-out estimates for RankRLS for label ranking. Then, we show how the hold-out can be used to derive a transductive version of the algoritm.

In [7], we described an efficient method for calculating hold-out estimates for the standard RLS algorithm in which several examples were held out simultaneously. The hold-out computations can be performed for RankRLS in a similar way. Here, we consider the case where the lth instance will be left out from the training set and used as a test instance for a learning machine trained with the rest of the training instances. Recall that the term instance refers to a set of input-score pairs, that is, not to an individual example only, and the input variables that correspond to the lth instance are indexed by $U_l \subset \{1, \ldots, m\}$. Below, we use a shorthand notation U instead of U_l. Leaving more than one instance out can be defined analogously. In that case, U would refer to a union of U_l, where l goes through every hold-out instance.

With any matrix (or a column vector) M that has its rows indexed by a superset of U, we use the subscript U so that the matrix M_U contains only the rows that are indexed by U. Similarly, for any matrix M that has its rows indexed by a superset of U and columns indexed by a superset of V, we use M_{UV} to denote a matrix that contains only the rows indexed by U and the columns indexed by V. Further, we denote $\overline{U} = \{1, \ldots, m\} \setminus U$, and $f_{\overline{U}} = \mathcal{A}(X_{\overline{U}}, Y_{\overline{U}}, W_{\overline{U}\overline{U}})$.

Let $Q = L_{\overline{U}\overline{U}} K_{\overline{U}\overline{U}} + \lambda I_{\overline{U}\overline{U}}$. Then, the predicted scores for the inputs of the held out instance can be obtained, by definition, from

$$f_{\overline{U}}(X_U) = K_{U\overline{U}} Q^{-1} L_{\overline{U}\overline{U}} Y_{\overline{U}}. \tag{9}$$

However, having already calculated the solution with the whole training set, the predictions for the held out instance can be performed more efficiently than with the naive method.

Let $R = LK + \frac{1}{2}\lambda I$ and $P = R^{-1}$. By definition, $L_{U_l \overline{U}_l} = 0$ for all $1 \leq l \leq q$, and hence we can write $Q^{-1} = (R_{\overline{U}\overline{U}})^{-1}$. Further, due to the matrix inversion lemma,

$$(R_{\overline{U}\overline{U}})^{-1} = P_{\overline{U}\overline{U}} - P_{\overline{U}U}(P_{UU})^{-1} P_{U\overline{U}}.$$

When (5) has been solved with the whole training set, we already have the matrix P stored in memory, and hence the computational complexity of calculating the matrix products and inversions (in the optimal order) involved in (9) is $O(m^2 + |U|^3)$. This is more efficient than the naive method of calculating the inverse of Q with complexity $O(m^3)$. The hold out method can be used to calculate a cross-validation efficiently.

Next, we consider a transductive version of the RankRLS algorithm. We assume that one (or several) of the q instances is given without the scoring information. In contrast to the hold-out procedure discussed above, the learning algorithm is given a chance to exploit the information in test instances, that is, the instances without the scoring information: the input of the algorithm consists of the instances with the scoring information, the respective score values, and the instances without scoring information. As previously, the aim is to predict the score values for the test instances. We do this by selecting such score values that minimize the least-squares approximation of the cross-validated ranking error on the whole data set. Our idea is inspired by the method proposed in [2], where output variables of the unlabeled instances were selected so that the leave-one-out cross-validated regression error on the whole data set was minimized. In contrast, we perform a more general cross-validation procedure, because holding out an instance means excluding every example associated with it. In addition, we aim to minimize the ranking error instead on the regression error.

Formally, the transductive RankRLS algorithm is constructed as follows. Let V be the index set of the inputs that belong to instances without the score values. Then, the score values Y_V are obtained from

$$Y_V = \operatorname*{argmin}_{\widehat{Y} \in \mathbb{R}^{|V|}} J, \tag{10}$$

where

$$J = \Big(\sum_U c(f^V_{\overline{U}}(X_U), Y^V_U, W_{UU}) \Big) + \gamma \|\widehat{Y} - f_{\overline{V}}(X_V)\|^2, \tag{11}$$

U goes through every cross-validation fold, c is the cost function defined in (7), $\gamma > 0$, $f^V_{\overline{U}} = \mathcal{A}(X_{\overline{U}}, Y^V_{\overline{U}}, W_{\overline{U}\overline{U}})$, and Y^V is a sequence of score values in which the elements indexed by V are set to $\widehat{Y}$. The second term of (11) penalizes the divergence of $\widehat{Y}$ from $f_{\overline{V}}(X_V)$.

The minimizer of (11) is characterized by the following theorem.

THEOREM 2. *The solution of (10) is*

$$\widehat{Y} = (\sum_U E + \gamma I_{VV})^{-1} (\sum_U F + \gamma f_{\overline{V}}(X_V)), \tag{12}$$

where

$$
\begin{aligned}
E &= \begin{pmatrix} L_{SS} & -L_{SS}G_{SH} \\ -G_{SH}^{T}L_{SS} & G_{UH}^{T}L_{UU}G_{UH} \end{pmatrix}, \\
F &= \begin{pmatrix} L_{SS}G_{SB}Y_B \\ G_{ZH}^{T}L_{ZZ}Y_Z - G_{UH}^{T}L_{UU}G_{UB}Y_B \end{pmatrix}, \\
G &= K_{U\overline{U}}(P_{\overline{UU}} - P_{\overline{U}U}(P_{UU})^{-1}P_{U\overline{U}})L_{\overline{UU}}, \\
S &= V \cap U, \\
H &= V \cap \overline{U}, \\
Z &= \overline{V} \cap U, \text{ and} \\
B &= \overline{V} \cap \overline{U}.
\end{aligned}
$$

Proof. We write (11) in a matrix form:

$$
\begin{aligned}
J &= \sum_U (Y_U^V - f_{\overline{U}}^V(X_U))^{\mathrm{T}} L_{UU}(Y_U^V - f_{\overline{U}}^V(X_U)) \\
&\quad + \gamma(\widehat{Y} - f_{\overline{V}}(X_V))^{\mathrm{T}}(\widehat{Y} - f_{\overline{V}}(X_V)).
\end{aligned}
$$

We observe that, when the rows and the columns of the matrices and vectors are rearranged appropriately,

$$
\begin{aligned}
f_{\overline{U}}^V(X_U) &= \begin{pmatrix} G_{SH} & G_{SB} \\ G_{ZH} & G_{ZB} \end{pmatrix} \begin{pmatrix} \widehat{Y}_H \\ Y_B \end{pmatrix}, \\
L_{UU} &= \begin{pmatrix} L_{SS} & 0 \\ 0 & L_{ZZ} \end{pmatrix}, \\
Y_U^V &= \begin{pmatrix} \widehat{Y}_S \\ Y_Z \end{pmatrix}, \text{ and} \\
\widehat{Y} &= \begin{pmatrix} \widehat{Y}_S \\ \widehat{Y}_H \end{pmatrix}.
\end{aligned}
$$

Hence,

$$
\begin{aligned}
J &= \sum_U \Big(\widehat{Y}_S^{\mathrm{T}} L_{SS} \widehat{Y}_S \\
&\quad -2\widehat{Y}_S^{\mathrm{T}} L_{SS} G_{SH} \widehat{Y}_H \\
&\quad -2\widehat{Y}_S^{\mathrm{T}} L_{SS} G_{SB} Y_B \\
&\quad -2\widehat{Y}_H^{\mathrm{T}} G_{ZH}^{\mathrm{T}} L_{ZZ} Y_Z \\
&\quad +2\widehat{Y}_H^{\mathrm{T}} G_{UH}^{\mathrm{T}} L_{UU} G_{UB} Y_B \\
&\quad +\widehat{Y}_H^{\mathrm{T}} G_{UH}^{\mathrm{T}} L_{UU} G_{UH} \widehat{Y}_H \Big) \\
&\quad +\gamma(\widehat{Y}^{\mathrm{T}}\widehat{Y} - 2\widehat{Y}^{\mathrm{T}} f_{\overline{V}}(X_V)) + C \\
&= \widehat{Y}^{\mathrm{T}} \left(\sum_U E \right) \widehat{Y} - 2\widehat{Y}^{\mathrm{T}} \left(\sum_U F \right) \\
&\quad +\gamma(\widehat{Y}^{\mathrm{T}}\widehat{Y} - 2\widehat{Y}^{\mathrm{T}} f_{\overline{V}}(X_V)) + C,
\end{aligned}
$$

where C is a constant that does not depend on $\widehat{Y}$. We take the derivative of J with respect to $\widehat{Y}$:

$$
\frac{d}{d\widehat{Y}} J = 2\left(\sum_U E\right)\widehat{Y} - 2\left(\sum_U F\right) + \gamma(2\widehat{Y} - 2f_{\overline{V}}(X_V)).
$$

Finally, we set the derivative to zero and solve with respect to $\widehat{Y}$:

$$
\widehat{Y} = (\sum_U E + \gamma I_{VV})^{-1}(\sum_U F + \gamma f_{\overline{V}}(X_V)),
$$

that is equal to (12). □

When the matrix products in (12) are calculated in the optimal order, the computational complexity of predicting $\widehat{Y}$ with the transductive RankRLS is $O(m^3)$.

Acknowledgments

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

4. REFERENCES

[1] 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.

[2] O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 421–427. The MIT Press, Cambridge, MA, 1999.

[3] 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.

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

[5] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Proceedings of the Ninth International Conference on Articial Neural Networks*, pages 97–102, London, 1999. Institute of Electrical Engineers.

[6] 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.

[7] 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.

[8] T. Pahikkala, E. Tsivtsivadze, J. Järvinen, and J. Boberg. An efficient algorithm for learning preferences from comparability graphs, 2007. submitted.

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

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