Conditional Ranking on Relational Data

Tapio Pahikkala[1], Willem Waegeman[2], Antti Airola[1], Tapio Salakoski[1], and Bernard De Baets[2]

[1] University of Turku and Turku Centre for Computer Science,
Joukahaisenkatu 3-5 B, FIN-20520, Turku, Finland
`forname.surname@utu.fi`

[2] Department of Applied Mathematics, Biometrics and Process Control, Ghent University, Coupure links 653, B-9000 Ghent, Belgium
`forname.surname@UGent.be`

Abstract. In domains like bioinformatics, information retrieval and social network analysis, one can find learning tasks where the goal consists of inferring a ranking of objects, conditioned on a particular target object. We present a general kernel framework for learning conditional rankings from various types of relational data, where rankings can be conditioned on unseen data objects. Conditional ranking from symmetric or reciprocal relations can in this framework be treated as two important special cases. Furthermore, we propose an efficient algorithm for conditional ranking by optimizing a squared ranking loss function. Experiments on synthetic and real-world data illustrate that such an approach delivers state-of-the-art performance in terms of predictive power and computational complexity. Moreover, we also show empirically that incorporating domain knowledge in the model about the underlying relations can improve the generalization performance.

1 Introduction

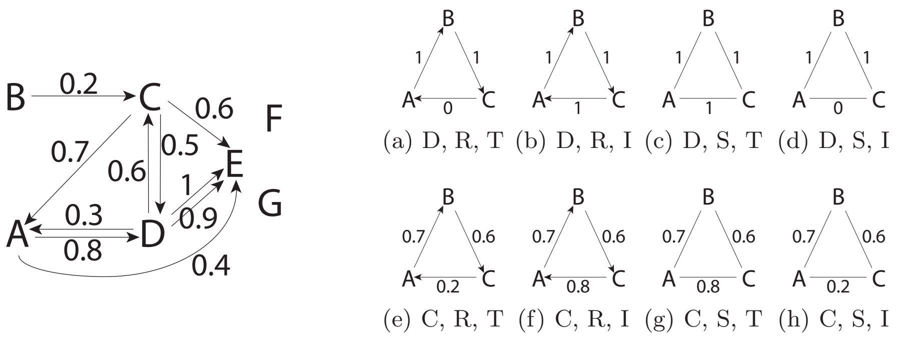
Let us start with two introductory examples to explain the problem setting of conditional ranking. Firstly, suppose that a number of persons are playing an online computer game. For many people it is always more fun to play against someone with similar skills, so players might be interested in receiving a ranking of other players, ranging from extremely difficult to beat to novice players with no experience at all. Unfortunately, pairwise strategies of players in many games – not only in computer games but also in board or sports games – tend to exhibit a rock-paper-scissors type of relationship [1], in the sense that player A beats with a high probability player B, who on his term beats with a high probability person C, while player A has a high chance of losing from the same player C. Mathematically speaking, the relation between players is not transitive, leading to a cyclic relationship and implying that no global (consistent) ranking of skills exists, yet a conditional ranking can always be obtained for a specific player [2].

As a second introductory example, let us consider the supervised inference of biological networks, like protein-protein interaction networks, where the goal usually consists of predicting new interactions from a set of highly-confident interactions [3]. Similarly, one can also define a conditional ranking task in such a context, as predicting a ranking of all proteins in the network that are likely to interact with a given target protein [4]. However, this conditional ranking task differs from the previous one because (a) rankings are computed from symmetric relations instead of reciprocal ones and (b) the values of the relations are here usually not continuous but discrete.

Applications for conditional ranking tasks arise in many domains where relational information between objects is observed, such as relations between persons in preference modelling, social network analysis and game theory, links between documents or websites in information retrieval and text mining, interactions between genes or proteins in bioinformatics, etc. When approaching conditional ranking from a graph inference point of view, the goal consists of returning a ranking of all nodes given a particular target node, in which the nodes provide information in terms of features and edges in terms of labels or relations. At least two properties of graphs play a key role in such a setting. Firstly, the type of information stored in the edges defines the learning task: binary-valued edge labels lead to bipartite ranking tasks [5], ordinal-valued edge labels to multipartite or layered ranking tasks [6, 7] and continuous labels result in rankings that are nothing more than total orders (when no ties occur). Secondly, the relations that are represented by the edges might have interesting properties, namely symmetry or reciprocity, for which conditional ranking can be interpreted differently.

We present in this article a kernel framework for conditional ranking, which covers all above situations. Unlike existing single-task or multi-task ranking algorithms, where the conditioning is respectively ignored or only happening for training objects, our approach also allows to condition on new data objects that are not known during the training phase. Thus, in light of Figure 1 that will be explained below, the algorithm is not only able to predict conditional rankings for objects A to E, but also for objects F and G that do not contribute to the training dataset. One might argue that existing ranking methods like RankSVM [8] or RankRLS [9, 10] can learn conditional rankings, by interpreting the conditioning object as a query. We will show in Section 4 that these methods become computationally intractable in many applications, unlike our approach, which is much more efficient due to exploiting the knowledge that the objects to be ranked and the queries (here objects too) are sampled from the same domain.

As a second difference, information retrieval methods that could be interpreted as conditional ranking predominantly use similarity as underlying relation, often in a pure intuitive manner, as a nearest neighbor type of learning. Think in this context at the example of protein ranking given above [4] or algorithms like *query by document* [11]. These methods simply look for rankings where the most similar objects w.r.t. the conditioning object appear on top, contrary to our approach, which should be considered as much more general, since we learn rankings from any type of binary relation. Nonetheless, similarity relations will of course still occupy a prominent place in our framework as an important special case. We will demonstrate below that domain knowledge about the underlying relations can be easily incorporated in our framework.



(a) D, R, T (b) D, R, I (c) D, S, T (d) D, S, I

(e) C, R, T (f) C, R, I (g) C, S, T (h) C, S, I

Fig. 1. Left: example of a multi-graph representing the most general case, where no additional properties of relations are assumed. Right: examples of eight different types of relations in a graph of cardinality three. The following relational properties are illustrated: (D) discrete, (C) continuous, (R) reciprocal, (S) symmetric, (T) transitive and (I) intransitive.

2 General framework

Let us start with introducing some notations. We consider ranking of data structured as a graph $G = (\mathcal{V}, \mathcal{E}, Q)$, where $\mathcal{V}$ corresponds to the set of nodes and $\mathcal{E} \subseteq \mathcal{V}^2$ represents the set of edges e, for which training labels are provided in terms of relations. Moreover, these relations are represented by training weights y_e on the edges and these relations are generated from an unknown underlying relation $Q : \mathcal{V}^2 \to [0, 1]$. Relations are required to take values in the interval $[0, 1]$ because some properties that we need are historically defined for such relations, but an extension to real-valued relations $h : \mathcal{V}^2 \to \mathbb{R}$ can always be realized with a simple monotonic mapping $g : \mathbb{R} \to [0, 1]$ such that

$$Q(v, v') = g(h(v, v')) , \quad \forall (v, v') \in \mathcal{V}^2 . \tag{1}$$

Following the standard notations for kernel methods, we formulate our learning problem as the selection of a suitable function $h \in \mathcal{H}$, with $\mathcal{H}$ a certain hypothesis space, in particular a reproducing kernel Hilbert space (RKHS). Hypotheses $h : \mathcal{V}^2 \to \mathbb{R}$ are usually denoted as $h(e) = \langle \mathbf{w}, \Phi(e) \rangle$ with $\mathbf{w}$ a vector of parameters that needs to be estimated based on training data. Let us denote a training dataset of cardinality $q = |\mathcal{E}|$ as a sequence $T = \{(e, y_e) \mid e \in \mathcal{E}\}$ of input-label pairs, then we formally consider the following variational problem in which we select an appropriate hypothesis h from $\mathcal{H}$ for training data T. Namely, we consider an algorithm

$$\mathcal{A}(T) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathcal{L}(h, T) + \lambda \|h\|^2_{\mathcal{H}} \tag{2}$$

with $\mathcal{L}$ a given loss function and $\lambda > 0$ a regularization parameter. According to the representer theorem [12], any minimizer $h \in \mathcal{H}$ of (2) admits a dual representation of the following form:

$$h(\overline{e}) = \langle \mathbf{w}, \Phi(\overline{e}) \rangle = \sum_{e \in \mathcal{E}} a_e K^{\Phi}(e, \overline{e}) \,, \tag{3}$$

with $a_e \in \mathbb{R}$ dual parameters, K^{Φ} the kernel function associated with the RKHS and Φ the feature mapping corresponding to K^{Φ}.

Given two relations $Q(v, v')$ and $Q(v, v'')$ defined on any triplet of nodes in $\mathcal{V}$, we compose the ranking of v' and v'' conditioned on v as

$$v' \succeq_v v'' \Leftrightarrow Q(v, v') \geq Q(v, v'') \,. \tag{4}$$

Let the number of correctly ranked pairs for all nodes in the dataset serve as evaluation criterion for verifying (4), then one aims to minimize the following empirical loss when computing the loss over all conditional rankings simultaneously:

$$\mathcal{L}(h, T) = \sum_{v \in \mathcal{V}} \sum_{e, \overline{e} \in \mathcal{E}_v : y_e < y_{\overline{e}}} I(h(e) - h(\overline{e})) \,, \tag{5}$$

with I the Heaviside function returning one when its argument is strictly positive, returning $1/2$ when its argument is exactly zero and returning zero otherwise. Importantly, $\mathcal{E}_v$ denotes the set of all edges starting from or ending at the node v, depending on the specific task. For example, concerning the relation "trust" in a social network, the former loss would correspond to ranking the persons in the network who *are trusted by* a specific person, while the latter loss corresponds to ranking the persons who *trust* that person. So, taking Figure 1 into account, we would in such an application respectively use the rankings $A \succ_C E \succ_C D$ (outgoing edges) and $D \succ_C B$ (incoming edges) as training info for node C.

Since (5) is neither convex nor differentiable, one should look for a differentiable and convex approximation of it. Let us to this end start by considering the following squared loss function over the q observed edges in the training set:

$$\mathcal{L}(h, T) = \sum_{e \in \mathcal{E}} (y_e - h(e))^2. \tag{6}$$

Such a setting would correspond to directly learning the labels on the edges in a regression or classification setting. For the latter case, optimizing (6) instead of the more conventional hinge loss has the advantage that the solution can be found by simply solving a system of linear equations [13]. However, when doing conditional ranking, the simple squared loss might not be optimal. Consider for example that we have a node v and we aim to learn to predict which of the two other nodes, v' or v'', would be closer to it. Let us denote $e = (v, v')$ and $\overline{e} = (v, v'')$, and let y_e and $y_{\overline{e}}$ denote the relation between v and v' and between v and v'', respectively. Then, it would be beneficial for the regression function to have a minimal squared difference $(y_e - y_{\overline{e}} - h(e) + h(\overline{e}))^2$, leading to the following loss function:

$$\mathcal{L}(h, T) = \sum_{v \in \mathcal{V}} \sum_{e, \overline{e} \in \mathcal{E}_v} (y_e - y_{\overline{e}} - h(e) + h(\overline{e}))^2, \tag{7}$$

which can be interpreted as a differentiable and convex approximation of (5).

When no further restrictions on the underlying relation can be specified, then the following Kronecker product feature mapping is used to express pairwise interactions between features of nodes:

$$\Phi(e) = \Phi(v, v') = \phi(v) \otimes \phi(v') \, ,$$

where ϕ represents the feature mapping for individual nodes. As shown in [14], such a pairwise feature mapping yields the tensor product pairwise kernel in the dual model:

$$K^{\Phi}_{\otimes}(e, \overline{e}) = K^{\Phi}_{\otimes}(v, v', \overline{v}, \overline{v}') = K^{\phi}(v, \overline{v}) K^{\phi}(v', \overline{v}') \, ,$$

with K^{ϕ} the kernel corresponding to ϕ. With an appropriate choice for K^{ϕ}, such as the Gaussian RBF kernel, the kernel K^{Φ} generates a class $\mathcal{H}$ of universally approximating functions for learning any type of relation (formal proof omitted).

3 Special relations

In the above framework, we generally allow that for any pair of nodes in the graph several edges can exist, in which an edge in one direction not necessarily imposes constraints on the edge in the opposite direction and multiple edges in the same direction can connect two nodes, leading to a multi-graph like Figure 1, where two different edges in the same direction connect nodes D and E. This construction is required to allow repeated measurements. However, within the context of conditional ranking, two particular cases deserve further attention: symmetric relations and reciprocal relations

A binary relation $Q : \mathcal{V}^2 \to [0,1]$ is called a symmetric relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $Q(v, v') = Q(v', v)$. For symmetric relations, edges in multi-graphs like Figure 1 become undirected. Applications arise in many domains and metric learning or learning similarity measures can be seen as special cases. If the relation becomes discrete as $Q : \mathcal{V}^2 \to \{0, 1\}$, then we end up with a classification setting instead of a regression setting.

A binary relation $Q : \mathcal{V}^2 \to [0,1]$ is called a reciprocal relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $Q(v, v') = 1 - Q(v', v)$. For reciprocal relations, every edge $e = (v, v')$ in a multi-graph like Figure 1 induces an unobserved invisible edge $e_R = (v', v)$ with appropriate weight in the opposite direction. Applications arise here in domains such as preference learning, game theory and bioinformatics for representing preference relations, choice probabilities, winning probabilities, gene regulation, etc. The weight on the edge defines the real direction of such an edge. If the weight on the edge $e = (v, v')$ is higher than 0.5, then the direction is from v to v', but when the weight is lower than 0.5, then the direction should be interpreted as inverted, for example, the edges from A to C in Figures 1 (a) and (e) should be interpreted as edges starting from A instead of C. If the relation becomes discrete as $Q : \mathcal{V}^2 \to \{0, 1/2, 1\}$, then we end up with a three-class ordinal regression setting instead of an ordinary regression setting.

Both for reciprocal relations and symmetric relations, additional properties can be assumed. Among these properties, we will further only use transitivity, which is differently defined for reciprocal relations and symmetric relations. Due to lack of space, we omit a formal discussion here, but Figure 1 shows with examples what transitivity means for symmetric and reciprocal relations that are discrete and continuous. Symmetry and reciprocity can be easily incorporated in our framework. Let Ψ be a feature mapping on $\mathcal{V}^2$, let $g : \mathbb{R} \rightarrow [0,1]$ be a monotonically increasing mapping and let h be a hypothesis defined by (3), then the relation Q of type (1) is

1. reciprocal, if Φ is given by $\Phi_R(e) = \Phi_R(v,v') = \Psi(v,v') - \Psi(v',v)$ and g satisfies $g(1/2) = 0$ and for all $x \in \mathbb{R}$, $g(x) = 1 - g(-x)$.
2. symmetric, if Φ is given by $\Phi_S(e) = \Phi_S(v,v') = \Psi(v,v') + \Psi(v',v)$.

In addition, one can easily show that reciprocity and symmetry as domain knowledge can be enforced in the dual formulation. Let us in the least restrictive form now consider the Kronecker product for Ψ, then one obtains for Φ_R and Φ_S respectively the kernels $K^{\Phi}_{\otimes R}$ and $K^{\Phi}_{\otimes S}$ given by

$$K^{\Phi}_{\otimes R}(e,\overline{e}) = 2\big(K^{\phi}(v,\overline{v})K^{\phi}(v',\overline{v}') - K^{\phi}(v,\overline{v}')K^{\phi}(v',\overline{v})\big),$$
$$K^{\Phi}_{\otimes S}(e,\overline{e}) = 2\big(K^{\phi}(v,\overline{v})K^{\phi}(v',\overline{v}') + K^{\phi}(v,\overline{v}')K^{\phi}(v',\overline{v})\big).$$

The former kernel has been proposed for learning preference relations where transitivity not necessarily holds [2], while the latter one is used for predicting protein-protein interactions in bioinformatics [14]. Unlike many existing kernel-based methods for pairwise data, the models obtained with these kernels are able to represent any reciprocal or symmetric relation respectively, without imposing additional transitivity properties of the relations.

For arbitrary relations, the rankings obtained when conditioning on the first or second argument of Q are not necessarily consistent – recall *trusts* versus *is trusted by*. Yet, for symmetric relations, the two conditional rankings obtained from $Q(v,\cdot)$ and $Q(\cdot,v)$ are identical since for all $v,v',v'' \in \mathcal{V}$ it holds that

$$Q(v,v') \geq Q(v,v'') \Leftrightarrow Q(v',v) \geq Q(v'',v)\,. \tag{8}$$

Conversely, for reciprocal relations, the first ranking corresponds to reversing the second one and vice versa, since then for all $v,v',v'' \in \mathcal{V}$ it holds that

$$Q(v,v') \geq Q(v,v'') \Leftrightarrow Q(v',v) \leq Q(v'',v)\,. \tag{9}$$

In the examples in Figure 1, if we condition on the first or the last node, respectively, then we get for the symmetric graph (g) twice $C \succ_A B$, but the reciprocal graph (e) yields $C \succ_A B$ versus $B \succ_A C$.

In addition to enforcing symmetry or reciprocity as domain knowledge in the kernel, one can incorporate the extra information available for this type of relations in the loss function. Remark that in (5), each edge in the training set is considered only once, in the ranking conditioned either on the first or the second node of the edge. For reciprocal or symmetric relations, each observed edge can be twice taken into account, in the rankings conditioned on both nodes, as (9) and (8) respectively hold.

4 Links with existing ranking methods

Examining the pairwise loss (5) reveals that there exists a quite straightforward mapping from the task of conditional ranking to that of traditional ranking. Relation graph edges are in this mapping explicitly used for training and prediction. In recent years, several algorithms for learning to rank have been proposed, which can be used for conditional ranking, by interpreting the conditioning node as a query (see e.g. [15, 8, 5, 10, 16]). The main application has been in information retrieval, where the examples are joint feature representations of queries and documents, and preferences are induced only between documents connected to the same query. One of the earliest and most successful of these methods is the ranking support vector machine RankSVM [8], which optimizes the pairwise hinge loss. Even much more closely related is the ranking regularized least-squares method RankRLS [9, 10], previously proposed by some of the present authors. The method is based on minimizing the pairwise regularized squared loss and becomes equivalent to the algorithms proposed in this article, if it is trained directly on the relation graph edges.

What this in practice means is that when the training relation graph is sparse enough, say consisting of only a few thousand edges, existing methods for learning to rank can be used to train conditional ranking models. In fact this is how we perform the rock-paper-scissors experiments, as discussed in Section 6.1. However, if the training graph is dense, existing methods for learning to rank are of quite limited use.

Let us assume a training graph that has p nodes. Further, we assume that most of the edges in the graph are connected, meaning that the number of edges is of the order p^2. Using a learning algorithm that explicitly calculates the kernel matrix for the edges would thus need to construct and store a $p^2 * p^2$ matrix, which is intractable already when p is less than thousand. When the standard Kronecker kernel is used together with a linear kernel for the nodes, primal training algorithms could be used without forming the kernel matrix. Assuming on average d non-zero features per node, this would result in having to form a data matrix with $p^2 * d^2$ non-zero entries. Again, this would be both memorywise and computationally infeasible for relatively modest values of p and d.

Thus, building practical algorithms for solving the conditional ranking task requires computational shortcuts to avoid the above-mentioned space and time complexities. The methods presented in this article are based on such shortcuts, because queries and objects come from the same domain, resulting in a special structure of the Kronecker product kernel and a closed-form solution for the minimizer of the pairwise regularized squared loss.

5 Algorithmic aspects

Let p and q respectively represent the number of nodes and edges in T. Let $\mathbf{K} \in \mathbb{R}^{p \times p}$ be the kernel matrix of K^{ϕ}, containing similarities for all nodes in T, then $\overline{\mathbf{K}} = \mathbf{K} \otimes \mathbf{K} \in \mathbb{R}^{p^2 \times p^2}$ is the kernel matrix of $K^{\Phi}_{\otimes}$, computed on all possible couples of nodes in T. Moreover, let $\mathbf{B} \in \{0,1\}^{q \times p^2}$ be a bookkeeping matrix of the training data, that is, its rows and columns are indexed by the edges in the training set and the set of all possible pairs of nodes, respectively. Each row of $\mathbf{B}$ contains a single nonzero entry indicating to which pair of nodes the corresponding edge is connected.

Now, we show how the loss function (7) can be represented in a matrix form. This representation is similar to the RankRLS loss introduced by [9, 10]. Let

$$\mathbf{L}_l = \mathbf{I} - \frac{1}{l}\mathbf{1}\mathbf{1}^{\mathrm{T}} \tag{10}$$

be the $l \times l$-centering matrix with $l \in \mathbb{N}$. The matrix $\mathbf{L}$ is an 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

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

where c_i are the entries of any vector $\mathbf{c}$. Now, let us consider the following quasi-diagonal matrix:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{l_1} & & \\ & \ddots & \\ & & \mathbf{L}_{l_p} \end{pmatrix}, \tag{11}$$

where $l_i = |\mathcal{E}_{v_i}|$ for $i \in \{1, \ldots, p\}$. Up to multiplication with a constant, the loss function (7) can be represented in a matrix form as

$$\mathcal{L} = (\mathbf{y} - \mathbf{B}\overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{B}\overline{\mathbf{K}}\mathbf{a})\,, \tag{12}$$

provided that the entries of $\mathbf{y}$ and $\mathbf{B}$ are ordered in a way compatible with the entries of $\mathbf{L}$, that is, the training edges are arranged according to their starting nodes. Note that if we use an identity matrix instead of $\mathbf{L}$ in (12), the loss becomes the regression loss (6). In our experiments in Section 6, the conditional ranking and regression approaches are empirically compared.

Next, we will consider how the kernel matrices corresponding to the reciprocal kernel $K^{\Phi}_{\otimes R}$ and the symmetric kernel $K^{\Phi}_{\otimes S}$ can be represented in a matrix notation. Below, we assume that $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{r \times r}$. Let us consider the $r^2 \times r^2$-matrix defined as

$$\mathbf{P} = \sum_{i=1}^{r}\sum_{j=1}^{r}\mathbf{e}_{(i-1)r+j}\mathbf{e}^{\mathrm{T}}_{(j-1)r+i},$$

where $\mathbf{e}_i$ are the standard basis vectors of $\mathbb{R}^{r^2}$. In the literature, $\mathbf{P}$ is called the commutation matrix by [17]. For $\mathbf{P}$, we have the following properties. Firstly, $\mathbf{P}\mathbf{P} = \mathbf{I}$, since $\mathbf{P}$ is a symmetric permutation matrix. Moreover, we have $\mathbf{P}\mathrm{vec}(\mathbf{M}) = \mathrm{vec}(\mathbf{M}^{\mathrm{T}})$. Furthermore, we have $\mathbf{P}(\mathbf{M} \otimes \mathbf{N}) = (\mathbf{N} \otimes \mathbf{M})\mathbf{P}$. Next, we consider the matrices

$$\mathbf{S} = \frac{1}{2}(\mathbf{I} + \mathbf{P}),\ \mathbf{A} = \frac{1}{2}(\mathbf{I} - \mathbf{P})\,,$$

where $\mathbf{I}$ is the identity matrix. In the literature, $\mathbf{S}$ and $\mathbf{A}$ are known as the symmetrizer and skew-symmetrizer matrix, respectively (see e.g. [17]). From the properties of the commutation matrix, it is straightforward to determine the following properties of $\mathbf{S}$ and $\mathbf{A}$. Firstly, $\mathbf{S}\mathrm{vec}(\mathbf{M}) = \frac{1}{2}\mathrm{vec}(\mathbf{M} + \mathbf{M}^{\mathrm{T}})$ and $\mathbf{A}\mathrm{vec}(\mathbf{M}) = \frac{1}{2}\mathrm{vec}(\mathbf{M} - \mathbf{M}^{\mathrm{T}})$. Secondly, the matrices $\mathbf{S}$ and $\mathbf{A}$ are idempotent. Finally, the matrices $\mathbf{S}$ and $\mathbf{A}$ commute with the matrix $\mathbf{M} \otimes \mathbf{M}$.

It can be shown that the symmetry and reciprocity of the pairwise kernels can be enforced by using the matrices $\mathbf{S}$ and $\mathbf{A}$ of the same size as $\mathbf{K} \otimes \mathbf{K}$. Namely, the symmetrized and skew-symmetrized Kronecker kernel matrices for the couples are, up to multiplication with a constant, $\overline{\mathbf{K}} = \mathbf{S}(\mathbf{K} \otimes \mathbf{K})$ and $\overline{\mathbf{K}} = \mathbf{A}(\mathbf{K} \otimes \mathbf{K})$, respectively. Subsequently, we consider different approaches for solving the least-squares problem with loss (12). As already discussed in Section 4, the existing ranking algorithms can be used but this is feasible only in limited cases.

Alternatively, we can also use iterative training algorithms for solving the system of linear equations providing a solution to the learning task:

$$(\mathbf{B}^{\mathrm{T}}\mathbf{L}\mathbf{B}(\mathbf{K} \otimes \mathbf{K})\mathbf{S} + \lambda\mathbf{I})\mathbf{a} = \mathbf{B}^{\mathrm{T}}\mathbf{L}\mathbf{y},$$

where the symmetrizer matrix $\mathbf{S}$ can be replaced with the skew-symmetrizer or identity matrix, depending on the prior knowledge we have about the underlying relation. Here, we consider an approach based on conjugate gradient type of methods, which take advantage of the special structure of the kernel matrices and the loss function. The Kronecker product $(\mathbf{K} \otimes \mathbf{K})\mathbf{v}$ can be written as $\mathrm{vec}(\mathbf{KVK})$, where $\mathbf{v} = \mathrm{vec}(\mathbf{V}) \in \mathbb{R}^{p^2}$, $\mathbf{V} \in \mathbb{R}^{p \times p}$, and vec is the column vectorizing operator, which stacks the columns of a matrix in a column vector. Computing this product is cubic in the number of nodes. Moreover, multiplying a vector with the matrices $\mathbf{S}$ or $\mathbf{A}$ does not increase the computational complexity, since they are sparse. Finally, multiplying a vector with matrices $\mathbf{L}$ or $\mathbf{B}$ can be performed in $O(q)$ time, since the multiplication with the former is equivalent to performing a series of centering operations on the vector and the latter has only q non-zero elements. Conjugate gradient methods require, in the worst case, $O(p^4)$ iterations in order to solve the system of linear equations under consideration. However, the number of iterations required in practice is a small constant, as we show in the experiments. In addition, since using early stopping with the gradient-based methods has a regularizing effect on the learning process (see e.g. [18]), this approach can be used instead of the quadratic regularizer.

As a third approach, we note that in the special case in which $\mathbf{B}$ is the identity matrix – meaning that our training set consists of a graph having exactly two edges between each node, one for both directions – and in which the ordinary Kronecker kernel is used, the training of the conditional ranking method admits the following closed-form solution.

Proposition 1. *If $\mathbf{B}$ is the identity matrix and $\overline{\mathbf{K}} = \mathbf{K} \otimes \mathbf{K}$, the dual parameters corresponding to the minimizer of (2) with the loss (12) can be obtained from*

$$\mathbf{a} = \mathrm{vec}(\mathbf{U}(\mathbf{C} \odot (\mathbf{U}^{-1}\mathbf{L}_p\mathbf{Y}\mathbf{V}))\mathbf{V}^{\mathrm{T}}) \tag{13}$$

where $\odot$ is the Hadamard product, $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$*,*

$$\mathrm{diag}(\mathrm{vec}(\mathbf{C})) = (\mathbf{\Lambda} \otimes \mathbf{\Sigma} + \lambda \mathbf{I})^{-1},$$

diag *is the operator that maps vectors to diagonal matrices, and* $\mathbf{V\Lambda V}^{\mathrm{T}}$ *and* $\mathbf{U\Sigma U}^{-1}$ *are the eigen decompositions of* $\mathbf{K}$ *and* $\mathbf{L}_p\mathbf{K}$*, respectively.*

The proof consists of standard Kronecker product algebra. Since the eigen decompositions and matrix products in (13) can be performed in $O(p^3)$ time, this is also the time complexity of solving the above special case.

To conclude, we have the following three approaches for solving conditional ranking problems: (a) off-the-shelf ranking algorithms can be used when they can be computationally afforded, i.e., when the number of edges in the training set is small; (b) the above-presented approach based on the conjugate gradient method with early stopping and taking advantage of the special matrix structures is recommended when using off-the-shelf methods becomes intractable; (c) the closed-form solution presented in Proposition 1 is recommended if its requirements are fulfilled, since its computational complexity is equivalent to that of a single iteration of the conjugate gradient method.

6 Experiments

In the experiments we consider conditional ranking tasks on both synthetic and real-world data, illustrating different aspects of the generality of our approach. The first experiment is run on the synthetic rock-paper-scissors data set, in which the underlying relation is both reciprocal and intransitive. The task is to learn a model for ranking players according to their likelihood of winning against any other player on whom the ranking is conditioned. In the second experiment, run on the 20-newsgroups data set, the task is to rank documents according to their similarity to any other document, on which the ranking is conditioned.

In all the experiments, we run both the conditional ranker that minimizes the convex edgewise ranking loss approximation (7) and the method that minimizes the regression loss (6) over the edges. Further, in the rock-paper-scissors experiment we also train a conditional ranker with RankSVM. For the 20-newsgroups data this is not possible due to the large amount of edges present in the relational graph, resulting in too high memory requirements and computational costs for RankSVM training. We use the Kronecker kernel $K^{\Phi}_{\otimes}$ for edges in all the experiments, and also test the effects of enforcing domain knowledge by applying the reciprocal kernel $K^{\Phi}_{\otimes R}$ to the rock-paper-scissors dataset, and applying the symmetric kernel $K^{\Phi}_{\otimes S}$ in the 20-newsgroups experiments. The linear kernel is used for individual nodes (thus, for K^{ϕ}).

In all the experiments, performance is measured using the ranking loss (5) on the test set. As a test of statistical significance for comparing the performance differences between the learning methods, we use the paired Wilcoxon-signed-rank test with significance level 0.05. For the rock-paper-scissors dataset, the test error is calculated over 100 repetitions of the experiments and for the 20-newsgroups dataset the test error is calculated separately for each test node.

We use a variety of approaches for minimizing the squared conditional ranking and regression losses, depending on the characteristics of the task. All the used solvers are written in the Python programming language, and rely on the Numpy and Scipy libraries. In the rock-paper-scissors experiment, we train the methods directly on the feature representations of the edges using the standard RankRLS and RLS solvers available in the RLScore software package[3]. The majority of the 20-newsgroups experiments are run by solving the closed-form solution of the conditional ranker presented in Proposition 1, and the analogous solution for the conditional regressor, using standard matrix operations. For the experiment where the training is performed iteratively, we apply the biconjugate gradient stabilized method (BGSM)[19]. The RankSVM conditional ranker is trained with the SVM^{rank} software[4] described in [20].

6.1 Rock-paper-scissors

The synthetic benchmark data, whose generation process is described in detail in [2], consists of simulated games of the well-known game of rock-paper-scissors between pairs of players. The training set contains the outcomes of 1000 games played between 100 players, the outcomes are labeled according to which of the players won. The test set consists of another group of 100 players, and for each pair of players the probability of the first player winning against the second one. Different players differ in how often they play each of the three possible moves in the game. The data set can be considered as a directed graph where players are nodes and edges played games, the true underlying relation generating the data is in this case reciprocal. Moreover, the relation is intransitive. It represents the probability that one player wins against another player. Thus, it is not meaningful to try to construct a global ranking of the players. The task of conditional ranking, where players are ranked according to their estimated probability of winning against a given player, however is a sensible task.

We experiment with three different variations of the data set, the w1, w10 and w100 sets. These data sets differ in how balanced the strategies played by the players are. In w1 all the players have close to equal probability of playing any of the three available moves, while in w100 each of the players has a favorite strategy he/she will use much more often than the other strategies. Both the training and test sets in the three cases are generated one hundred times and the hundred ranking results are averaged for each of the three cases and for every tested learning method.

Since the training set consists of only one thousand games, it is feasible to adapt existing ranking algorithm implementations for solving the conditional ranking task. Each game is represented as two edges, labeled as $+1$ if the edge starts from the winner, and as -1 if the edge starts from the loser. Each node has

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

[4] Available at `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`

Method	RLS	RLS (rec)	RankRLS	RankRLS (rec)	RankSVM	RankSVM (rec)
$w = 1$	0.4875	0.4868	0.4876	0.4880	0.4984	0.4930
$w = 10$	0.04172	0.04145	0.04519	0.04291	0.04724	0.04273
$w = 100$	0.001384	0.001370	0.001428	0.001358	0.007408	0.006123

Fig. 2. Overview of the results for rock-paper-scissors. With the abbreviation rec we refer to the use of a reciprocal Kronecker kernel.

only 3 features, and thus the explicit feature representation where the Kronecker kernel is used together with a linear kernel results in 9 product features for each edge. In addition, we generate an analogous feature representation for the reciprocal Kronecker kernel. We use these generated feature representations for the edges to train three algorithms. RLS regresses directly the edge scores, RankRLS minimizes pairwise regularized squared loss on the edges, and RankSVM minimizes pairwise hinge loss on the edges. For RankRLS and RankSVM, pairwise preferences are generated only between edges starting from the same node.

In initial preliminary experiments we found out that on this data set regularization is harmful. All the considered methods reach their optimal performance with close to zero regularization parameter values, while the performance almost monotonically decreases with increased regularization. Since some regularization is necessary to guarantee the numeric stability and convergence of the solvers, we set the regularization parameter close to zero for all the considered methods.

The results of the experiments are presented in Figure 2. Clearly the methods are successful in learning conditional ranking models, and the easier the problem is made, the better is the performance. The one hundred repetitions are used to calculate the statistical significances between the performance differences of the methods and the general trends indicated by the tests are discussed next.

For w1, the results are very close to random and there are no statistical significances between the performances of the different methods. For w10, the pairwise ranking methods using the ordinary Kronecker kernel are statistically significantly worse than the pairwise ranking methods using the reciprocal Kronecker kernel. They are also significantly worse than both regression methods. However, there are no significant differences between the pairwise ranking methods using the reciprocal Kronecker kernel and the regression methods. The results for w100 are analogous to those for w10 except that RankSVM performs worse than the other methods. A possible reason for this is that for small values of the regularization parameter λ the training algorithms for optimizing least-squares types of losses are much more stable than the cutting plane training algorithm for RankSVM. The observation that regression can yield as low as, or even lower ranking error than using a pairwise loss is compatible with earlier results in the literature. For example the results presented in [10] show that for some data sets the regression approach may work as well as pairwise models.

In conclusion, we have in this section shown that highly intransitive relations can be modeled and successfully learned in the conditional ranking setting. Moreover, we have shown that when the relation graph of the training set is sparse enough, existing ranking algorithms can be applied by explicitly using the edges of the graph as training examples. Further, both pairwise ranking methods benefit from the use of the reciprocal Kronecker kernels instead of the ordinary Kronecker kernels, while it does not have an effect on the regression method. Finally, with this data, it appears that a regression-based approach performs as well as the pairwise ranking methods.

6.2 20-Newsgroups

In the second set of experiments we aim to learn to rank newsgroup documents according to their similarity with respect to a document the ranking is conditioned on. We use the publicly available 20-newsgroups data set[5] for the experiments. The data set consists of documents from 20 newsgroups, each containing approximately 1000 documents, the document features are word frequencies. Some of the newsgroups are considered to have similar topics, such as the rec.sport.baseball, and rec.sport.hockey newsgroups, which both contain messages about sports. We define a three-level conditional ranking task. Given a document, documents from the same newsgroup should be ranked highest, documents from similar newsgroups next, and documents from unrelated newsgroups last. Thus, we aim to learn the conditional ranking model from an undirected graph, and the underlying similarity relation is a symmetric relation. The setup is similar to that of [21], the difference is that we aim to learn a model for conditional ranking instead of just ranking documents against a fixed newsgroup.

Since the training relation graph is fully connected, the number of edges grows quadratically with the number of nodes. For 5000 training nodes, as considered in one of the experiments, this results already in a graph of approximately 25 million edges, with $1.25 * 10^{11}$ pairwise preferences. Thus unlike in the previous rock-paper-scissors experiment, training a ranking algorithm directly on the edges of the graph is no longer feasible. Instead, we solve the closed-form presented in Proposition 1. At the end of this section we also present experimental results for the iterative BGSM training algorithm, as this allows us to examine the effects of early stopping, and enforcing symmetry on the prediction function.

In the first two experiments, where the closed form solution is applied, we assume a setting where the set of available newsgroups is not static, but rather over time old newsgroups may wither and die out, or new groups may be added. Thus we cannot assume, when seeing new examples, that we have seen documents from the same newsgroup already when training our model. We simulate this by selecting different newsgroups for testing than for training. We form two disjoint sets of newsgroups. Set 1 contains the messages from the newsgroups rec.autos, rec.sport.baseball, comp.sys.ibm.pc.hardware and comp.windows.x, set 2 the messages from the newsgroups rec.motorcycles, rec.sport.hockey, comp.graphics, comp.os.ms-windows.misc and comp.sys.mac.hardware. Thus the graph formed by set 1 consists of approximately 4000 nodes and 16 million edges, and the graph

[5] Available at: `http://people.csail.mit.edu/jrennie/20Newsgroups/`

Method	Exp. 1	Exp. 2
c.rank	**0.2562**	**0.2895**
c.reg	0.3685	0.3967
p-value	< 0.001	< 0.001

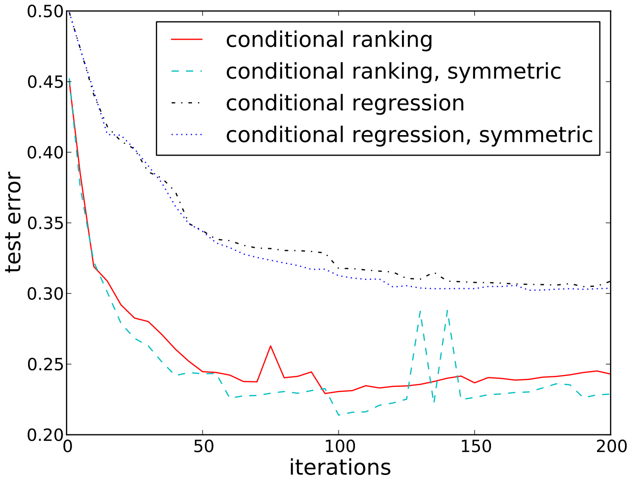


Fig. 3. Experimental results on the newsgroup data. Results for the large-scale experiments with closed-form solution (left table). Results for the small-scale experiment with BGSM and early stopping (right image).

formed by set 2 contains approximately 5000 nodes and 25 million edges. In the first experiment, set 1 is used for training and set 2 for testing. In the second experiment, set 2 is used for training and set 1 for testing. The regularization parameter is selected by using half of the training newsgroups as a holdout set against which the parameters are tested. When training the final model all the training data is combined back together.

The results for the closed form solution experiments are presented in Figure 3. Both methods are successful in learning a conditional ranking model that generalizes to new newsgroups which were not seen during the training phase. The method optimizing a ranking based loss over the pairs outperforms the one regressing the values for the pairwise relations in a statistically significant way.

Finally, we study whether enforcing the prior knowledge about the underlying relation being symmetric is beneficial. In this final experiment we use the iterative BGSM method, as it is compatible with the symmetric Kronecker kernel, unlike the solution of Proposition 1. The change in setup results in an increased computational cost, since each iteration of the BGSM method costs as much as using Proposition 1 to calculate the solution. Therefore, we simplify the previous experimental setup by sampling a training set of 1000 nodes, and a test set of 500 nodes from 5 newsgroups. The task is now easier than before, since the training and test sets have the same distribution. All the methods are trained for 200 iterations, and test error is plotted. We do not apply any regularization, but rather rely on the regularizing effect of early stopping, as discussed in Section 5.

Figure 3 contains the performance curves. Again, we see that the pairwise ranking loss quite clearly outperforms the regression loss. Using prior knowledge about the learned relation by enforcing symmetry leads to increased performance, most notably for the ranking loss. The performance curves flatten out within the 200 iterations, demonstrating the feasibility of early stopping.

In conclusion, we have demonstrated various characteristics of our approach in the newsgroups experiments. First, we showed that the introduced methods scale to training graphs that consist of tens of millions of edges, each having a high-dimensional feature representation. Second, we showed the generality of our approach, as it is possible to learn conditional ranking models even when the test newsgroups are not represented in the training data, as long as data from similar newsgroups is available. Unlike the earlier experiments on the rock-paper-scissors data, the pairwise loss yields a dramatic improvement in performance compared to a regression based loss. Finally, enforcing prior knowledge about the type of the underlying relation with kernels was shown to be advantageous.

7 Conclusion

We presented in this article a general framework for conditional ranking from various types of relational data, where rankings can be conditioned on unseen objects and reciprocal or symmetric relations can be treated as two important special cases. We proposed in addition an efficient least-squares algorithm that optimizes a ranking-based loss function of type (5). Experimental results on a synthetic and a real-world dataset confirm that such an approach can lead to statistically significant improvements in performance when the task consists of conditional ranking instead of just trying to predict the underlying relations. Moreover, we also showed empirically that incorporating domain knowledge about the underlying relations can boost the generalization performance.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. T.P. is supported for this work by the Academy of Finland and W.W. by the Research Foundation of Flanders.

References

1. Fisher, L.: Rock, Paper, Scissors: Game Theory in Everyday Life. Basic Books (2008)
2. Pahikkala, T., Waegeman, W., Tsivtsivadze, E., Salakoski, T., De Baets, B.: Learning intransitive reciprocal relations with kernel methods. European Journal of Operational Research **206**(3) (2010) 676–685
3. Yamanishi, Y., Vert, J., Kanehisa, M.: Protein network inference from multiple genomic data: a supervised approach. Bioinformatics **20** (2004) 1363–1370
4. Weston, J., Eliseeff, A., Zhou, D., Leslie, C., Noble, W.S.: Protein ranking: from local to global structure in the protein similarity network. Proceedings of the National Academy of Sciences of the United States of America **101**(17) (2004) 6559–6563

5. Freund, Y., Yier, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4** (2003) 933–969
6. Waegeman, W., De Baets, B., Boullart, L.: Learning layered ranking functions with structured support vector machines. Neural Networks **21**(10) (2008) 1511–1523
7. Fürnkranz, J., Hüllermeier, E., Vanderlooy, S.: Binary decomposition methods for multipartite ranking. In Buntine, W.L., Grobelnik, M., Mladenic, D., Shawe-Taylor, J., eds.: ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases. Volume 5781 of Lecture Notes in Computer Science., Springer (2009) 359–374
8. 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'02), ACM Press (2002) 133–142
9. 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. (2007) 27–33
10. Pahikkala, T., Tsivtsivadze, E., Airola, A., Järvinen, J., Boberg, J.: An efficient algorithm for learning to rank from preference graphs. Machine Learning **75**(1) (2009) 129–165
11. Yang, Y., Bansal, N., Dakka, W., Ipeirotis, P., Koudas, N., Papadias, D.: Query by document. In Baeza-Yates, R.A., Boldi, P., Ribeiro-Neto, B.A., Cambazoglu, B.B., eds.: Proceedings of the 2nd International Conference on Web Search and Data Mining, ACM Press (2009) 34–43
12. Schölkopf, B., Smola, A.: Learning with Kernels, Support Vector Machines, Regularisation, Optimization and Beyond. The MIT Press (2002)
13. Suykens, J., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific Pub. Co., Singapore (2002)
14. Ben-Hur, A., Noble, W.: Kernel methods for predicting protein-protein interactions. Bioinformatics **21 Suppl 1** (2005) 38–46
15. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In De Raedt, L., Wrobel, S., eds.: Proceedings of the 22nd international conference on Machine learning. Volume 119 of ACM International Conference Proceeding Series., ACM Press (2005) 89–96
16. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI '09: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, AUAI Press (2009)
17. Abadir, M., Magnus, J.: Matrix Algebra. Cambridge University Press (2005)
18. Engl, H., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Volume 375 of Mathematics and Its Applications. Kluwer Academic Publishers (1996)
19. van der Vorst, H.A.: BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing **13**(2) (1992) 631–644
20. Joachims, T.: Training linear SVMs in linear time. In Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D., eds.: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press (2006) 217–226
21. Agarwal, S.: Ranking on graph data. In Cohen, W.W., Moore, A., eds.: Proceedings of the 23rd International Conference on Machine Learning. Volume 148 of ACM International Conference Proceeding Series., ACM Press (2006) 25–32