Mach Learn (2013) 93:321–356
DOI 10.1007/s10994-013-5354-7

Efficient regularized least-squares algorithms for conditional ranking on relational data

Tapio Pahikkala · Antti Airola · Michiel Stock · Bernard De Baets · Willem Waegeman

Received: 2 December 2011 / Accepted: 10 April 2013 / Published online: 7 June 2013

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. We propose efficient algorithms for conditional ranking by optimizing squared regression and ranking loss functions. We show theoretically, that learning with the ranking loss is likely to generalize better than with the regression loss. Further, we prove that symmetry or reciprocity properties of relations can be efficiently enforced in the learned models. Experiments on synthetic and real-world data illustrate that the proposed methods deliver state-of-the-art performance in terms of predictive power and computational efficiency. Moreover, we also show empirically that incorporating symmetry or reciprocity properties can improve the generalization performance.

Editors: Eyke Hüllermeier and Johannes Fürnkranz.

T. Pahikkala (✉) · A. Airola
Department of Information Technology and Turku Centre for Computer Science, University of Turku, 20014 Turku, Finland
e-mail: Tapio.Pahikkala@utu.fi

A. Airola
e-mail: Antti.Airola@utu.fi

M. Stock · B. De Baets · W. Waegeman
Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Coupure links 653, 9000 Ghent, Belgium

M. Stock
e-mail: michiel.stock@ugent.be

B. De Baets
e-mail: Bernard.DeBaets@ugent.be

W. Waegeman
e-mail: willem.waegeman@ugent.be

Keywords Reciprocal relations · Symmetric relations · Learning to rank · Kernel methods · Regularized least-squares

1 Introduction

We first motivate the study by presenting some examples relevant for the considered learning setting in Sect. 1.1. Next, we briefly review and compartmentalize related work in Sect. 1.2, and present the main contributions of the paper in Sect. 1.3.

1.1 Background

Let us start with two introductory examples to explain the problem setting of conditional ranking. First, 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 unexperienced players. 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 (Fisher 2008), in the sense that player A beats player B (with probability greater than 0.5), who on his term beats C (with probability greater than 0.5), while player A loses from player C (as well with probability greater than 0.5). 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 (Pahikkala et al. 2010b).

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 (Yamanishi et al. 2004). 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 (Weston et al. 2004). 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 database objects, documents, websites, or images in information retrieval (Geerts et al. 2004; Grangier and Bengio 2008; Ng et al. 2011), interactions between genes or proteins in bioinformatics, graph matching (Caetano et al. 2009), et cetera. 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. First, the type of information stored in the edges defines the learning task: binary-valued edge labels lead to bipartite ranking tasks (Freund et al. 2003), ordinal-valued edge labels to multipartite or layered ranking tasks (Fürnkranz et al. 2009) and continuous labels result in rankings that are nothing more than total orders (when no ties occur). Second, the relations that are represented by the edges might have interesting properties, namely symmetry or reciprocity, for which conditional ranking can be interpreted differently.

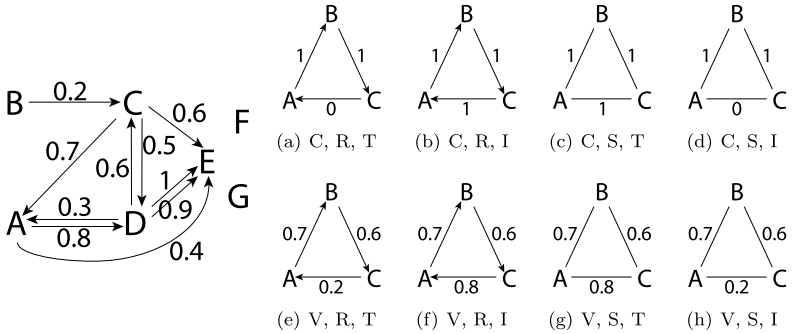

Fig. 1 *Left*: example of a multi-graph representing the most general case where no additional properties of relations are assumed. *Right*: examples of eight different types of relations in a graph of cardinality three. The following relational properties are illustrated: (C) crisp, (V) valued, (R) reciprocal, (S) symmetric, (T) transitive and (I) intransitive. For the reciprocal relations, (I) refers to a relation that does not satisfy weak stochastic transitivity, while (T) is showing an example of a relation fulfilling strong stochastic transitivity. For the symmetric relations, (I) refers to a relation that does not satisfy T-transitivity w.r.t. the Łukasiewicz t-norm $T_{\mathbf{L}}(a,b) = \max(a+b-1,0)$, while (T) is showing an example of a relation that fulfills T-transitivity w.r.t. the product t-norm $T_{\mathbf{P}}(a,b) = ab$—see e.g. Luce and Suppes (1965), De Baets et al. (2006) for formal definitions

1.2 Learning setting and related work

We present 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 Fig. 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 participate in the training dataset. From this perspective, one can define four different learning settings in total:

- Setting 1: predict a ranking of objects for a given conditioning object, where both the objects to be ranked and the conditioning object were contained in the training dataset (but not the ranking of the objects for that particular conditioning object).
- Setting 2: given a new conditioning object unseen in the training phase, predict a ranking of the objects encountered in the training phase.
- Setting 3: given a set of new objects unseen in the training phase, predict rankings of those objects with respect to the conditioning objects encountered in the training phase.
- Setting 4: predict a ranking of objects for a given conditioning object, where neither the conditioning object nor the objects to be ranked were observed during the training phase.

These four settings cover as special cases different types of conventional machine learning problems. The framework that we propose in this article can be used for all four settings, in contrast to many existing methods. In this paper, we focus mainly on setting 4.

Setting 1 corresponds to the imputation type of link prediction setting, where missing relational values between known objects are predicted. In this setting matrix factorization methods (see e.g. Srebro et al. 2005) are often applied. Many of the approaches are based solely on exploiting the available link structure, though approaches incorporating feature information have also been proposed (Menon and Elkan 2010; Raymond and Kashima 2010). Setting 2 corresponds to the label ranking problem (Hüllermeier et al. 2008), where a fixed set of labels is ranked given a new object. Setting 3 can be modeled as a multi-task ranking

problem where the number of ranking tasks is fixed in advance (Agarwal 2006). Finally, setting 4 requires that the used methods are able to generalize both over new conditioning objects and objects to be ranked (see e.g. Park and Marcotte 2012 for some recent discussion about this setting). Learning in setting 4 may be realized by using joint feature representations of conditioning objects and objects to be ranked.

In its most general form the conditional ranking problem can be considered as a special case of the listwise ranking problem, encountered especially in the learning to rank for information retrieval literature (see e.g. Cao et al. 2006; Yue et al. 2007; Xia et al. 2008; Qin et al. 2008; Liu 2009; Chapelle and Keerthi 2010; Qin et al. 2008; Kersting and Xu 2009; Xu et al. 2010; Airola et al. 2011b). For example, in document retrieval one is supplied both with query objects and associated documents that are ranked according to how well they match the query. The aim is to learn a model that can generalize to new queries and documents, predicting rankings that capture well the relative degree to which each document matches the test query.

Previous learning approaches in this setting have typically been based on using hand-crafted low-dimensional joint feature representations of query-document pairs. In our graph-based terminology, this corresponds to having a given feature representation for edges, possibly encoding prior knowledge about the ranking task. A typical example of this kind of joint feature particularly designed for the domain of information retrieval is the well-known Okapi BM25 score, indicating how well a given query set of words matches a given set of words extracted from a text document. While this type of features are often very efficient and absolutely necessary in certain practical tasks, designing such application-specific features requires human experts and detailed information about every different problem to be solved, which may not be always available.

In contrast, we focus on a setting in which we are only given a feature representation of nodes from which the feature representations of the edges have to be constructed, that is, the learning must be performed without access to the prior information about the edges. This opens many possibilities for applications, since we are not restricted to the setting where explicit feature representations of the edges are provided. In our experiments, we present several examples of learning tasks for which our approach can be efficiently used. In addition, the focus of our work is the special case where both the conditioning objects and the objects to be ranked come from the same domain (see e.g. Weston et al. 2004; Yang et al. 2009 for a similar settings). This allows us to consider how to enforce relational properties such as symmetry and reciprocity, a subject not studied in previous ranking literature. To summarize, the considered learning setting is the following:

(a) We are given a training set of objects (nodes), which have a feature representation of their own, either an explicit one or an implicit representation obtained via a nonlinear kernel function.
(b) We are also given observed relations between these objects (weighted edges), whose values are known only between the nodes encountered in the training set. This information is distinct from the similarities in point (a).
(c) The aim is to learn to make predictions for pairs of objects (edges) for which the value of this relation is unknown, by taking advantage of the features or kernels given in point (a), and in the conditional ranking setting, the goal is to learn a ranking of the object pairs.
(d) The node kernel is used as a building block to construct a pairwise kernel able to capture similarities of the edges, which in turn, is used for learning to predict the edge weights considered in point (b).

The proposed framework is based on the Kronecker product kernel for generating implicit joint feature representations of conditioning objects and the sets of objects to be

ranked. This kernel has been proposed independently by a number of research groups for modelling pairwise inputs in different application domains (Basilico and Hofmann 2004; Oyama and Manning 2004; Ben-Hur and Noble 2005). From a different perspective, it has been considered in structured output prediction methods for defining joint feature representations of inputs and outputs (Tsochantaridis et al. 2005; Weston et al. 2007).

While the usefulness of Kronecker product kernels for pairwise learning has been clearly established, computational efficiency of the resulting algorithms remains a major challenge. Previously proposed methods require the explicit computation of the kernel matrix over the data object pairs, hereby introducing bottlenecks in terms of processing and memory usage, even for modest dataset sizes. To overcome this problem, one typically applies sampling strategies to avoid computing the whole kernel matrix for training. However, non-approximate methods can be implemented by taking advantage of the Kronecker structure of the kernel matrix. This idea has been traditionally used to solve certain linear regression problems (see e.g. Van Loan 2000 and references therein). More recent and related applications have emerged in link prediction tasks by Kashima et al. (2009a), Raymond and Kashima (2010), which can be considered under setting 1.

An alternative approach known as the Cartesian kernel has been proposed by Kashima et al. (2009b) for overcoming the computational challenges associated with the Kronecker product kernels. This kernel indeed exhibits interesting computational properties, but it can be solely employed in selected applications, because it cannot make predictions for (couples of) objects that are not observed in the training dataset, that is, in setting 4 (see Waegeman et al. 2012 for further discussion and experimental results).

There exists a large body of literature about relational kernels, with values obtained from e.g. similarity graphs of data points, random walk and path kernels, et cetera. These can be considered to be complementary to the Kronecker product based pairwise kernels in the sense that they are used to infer similarities for data points rather than for pairs of data points. Thus, if relational information, such as paths or random walks for example, is used to form a kernel for the points in a graph, a pairwise kernel could be subsequently used to construct a kernel for the edges of the same graph.

Yet another family of related methods consists of the generalization of the pairwise Kronecker kernels framework to tasks, in which the condition and target objects come from different domains. Typical examples of this type of Kronecker kernel applications are found in bioinformatics, such as the task of predicting interactions between drugs and targets (van Laarhoven et al. 2011). To our knowledge, none of these studies still concern the fourth setting. While the algorithms considered in this paper can be straightforwardly generalized to the two domain case, we only focus on the single domain case for simplicity, because the concepts of symmetric and reciprocal relations would not be meaningful with two domains.

1.3 Contributions

The main contributions of this paper can be summarized as follows:

1. We propose kernel-based conditional ranking algorithms for setting 4, that is, for cases where predictions are performed for (couples of) objects that are not observed in the training dataset. We consider both regression and ranking based losses, extending regularized least-squares (RLS) (Saunders et al. 1998; Evgeniou et al. 2000) and the RankRLS (Pahikkala et al. 2009) algorithms to conditional ranking. We propose both update rules for iterative optimization algorithms, as well as closed-form solutions, exploiting the structure of the Kronecker product in order to make learning efficient. The proposed methods scale to graphs consisting of millions of labeled edges.

2. We show how prior knowledge about the underlying relation can be efficiently incorporated in the learning process, considering the specific cases of symmetric and reciprocal relations. These properties are enforced on the learned models via corresponding modifications of the Kronecker product kernel, namely the symmetric kernel studied by Ben-Hur and Noble (2005) and the reciprocal kernel introduced by us (Pahikkala et al. 2010b). We prove that, for RLS, symmetry and reciprocity can be implicitly encoded by including each edge in the training set two times, once for each direction. To our knowledge, the only related result so far has been established for the support vector machine classifiers by Brunner et al. (2012). We also prove that this implicitness, in turn, ensures the computational efficiency of the training phase with symmetric and reciprocal Kronecker kernels. These results are, to our knowledge, completely novel in the field of both machine learning and matrix algebra.
3. We present new generalization bounds, showing the benefits of applying RankRLS instead of basic RLS regression in conditional ranking tasks. The analysis presented in this paper shows that the larger expressive power of the space of regression functions compared to the corresponding space of conditional ranking functions indicates the learning to be likely to generalize better if the space of functions available for the training algorithm is restricted to conditional ranking functions rather than all regression functions.
4. Finally, we evaluate the proposed algorithms with an array of different practical problems. The results demonstrate the ability of the algorithms to solve learning problems in setting 4. Moreover, in our scalability experiments, we show that the algorithms proposed by us scale considerably better to large data sets than the state-of-the-art RankSVM solvers, even in cases where the SVMs use fast primal training methods for linear kernels.

1.4 Organization

The article is organized as follows. We start in Sect. 2 with a formal description of conditional ranking from a graph-theoretic perspective. The Kronecker product kernel is reviewed in Sect. 3 as a general edge kernel that allows for modelling the most general type of relations. In addition, we briefly recall two important subclasses of relations, namely symmetric and reciprocal relations, for which more specific, knowledge-based kernels can be derived. The proposed learning algorithms are presented in Sect. 4, and the connections and differences with related learning algorithms are discussed in Sect. 5, with a particular emphasis on the computational complexity of the algorithms. In Sect. 6 we present promising experimental results on synthetic and real-world data, illustrating the advantages of our approach in terms of predictive power and computational scalability.

2 General framework

Let us start with introducing some notations. We consider ranking of data structured as a graph $G = (V, E, Q)$, where $V \subseteq \mathcal{V}$ corresponds to the set of nodes, where nodes are sampled from a space $\mathcal{V}$, and $E \subseteq 2^{V^2}$ represents the set of edges e, for which labels are provided in terms of relations. Moreover, these relations are represented by weights y_e on the edges and they are generated from an unknown underlying relation $Q : \mathcal{V}^2 \to [0, 1]$. We remark that the interval $[0, 1]$ is used here only due to certain properties that are historically defined for such relations. However, relations taking values in arbitrary closed real intervals can be straightforwardly transformed to this interval with an appropriate increasing bijection and vice versa.

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). Given an input space $\mathcal{X}$ and a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the RKHS associated with K can be considered as the completion of

$$\left\{ f \in \mathbb{R}^{\mathcal{X}} \;\middle|\; f(x) = \sum_{i=1}^{m} \beta_i K(x, x_i) \right\},$$

in the norm

$$\|f\|_K = \sqrt{\sum_{i,j} \beta_i \beta_j K(x_i, x_j)},$$

where $\beta_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X}$.

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 need to be estimated based on training data. Let us denote a training dataset of cardinality $q = |E|$ as a set $T = \{(e, y_e) \mid e \in E\}$ 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

$$\mathcal{A}(T) = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{L}(h, T) + \lambda \|h\|_{\mathcal{H}}^2 \tag{1}$$

with $\mathcal{L}$ a given loss function and $\lambda > 0$ a regularization parameter.

According to the representer theorem (Kimeldorf and Wahba 1971), any minimizer $h \in \mathcal{H}$ of (1) admits a dual representation of the following form:

$$h(\overline{e}) = \big\langle \mathbf{w}, \Phi(\overline{e}) \big\rangle = \sum_{e \in E} a_e K^{\Phi}(e, \overline{e}), \tag{2}$$

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

$$v' \succeq_v v'' \quad \Leftrightarrow \quad Q\big(v, v'\big) \geq Q\big(v, v''\big). \tag{3}$$

Let the number of correctly ranked pairs for all nodes in the dataset serve as evaluation criterion for verifying (3), then one aims to minimize the following empirical loss when computing the loss over all conditional rankings simultaneously:

$$\mathcal{L}(h, T) = \sum_{v \in V} \sum_{e, \overline{e} \in E_v : y_e < y_{\overline{e}}} I\big(h(e) - h(\overline{e})\big), \tag{4}$$

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, E_v denotes the set of all edges starting from, or the set of all edges 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 Fig. 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 (4) is neither convex nor differentiable, we look for an approximation that has these properties as this considerably simplifies the development of efficient algorithms for solving the learning problem. Let us to this end start by considering the following squared loss function over the q observed edges in the training set:

$$\mathcal{L}(h,T) = \sum_{e \in E} \big(y_e - h(e)\big)^2. \tag{5}$$

Such a setting would correspond to directly learning the labels on the edges in a regression or classification setting. For the latter case, optimizing (5) instead of the more conventional hinge loss has the advantage that the solution can be found by simply solving a system of linear equations (Saunders et al. 1998; Suykens et al. 2002; Shawe-Taylor and Cristianini 2004; Pahikkala et al. 2009). However, the simple squared loss might not be optimal in conditional ranking tasks. Consider for example 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:

$$\mathcal{L}(h,T) = \sum_{v \in V} \sum_{e,\overline{e} \in E_v} \big(y_e - y_{\overline{e}} - h(e) + h(\overline{e})\big)^2, \tag{6}$$

which can be interpreted as a differentiable and convex approximation of (4).

3 Relational domain knowledge

Above, a framework was defined where kernel functions are constructed over the edges, leading to kernels of the form $K^{\Phi}(e, \overline{e})$. In this section we show how these kernels can be constructed using domain knowledge about the underlying relations. The same discussion was put forward for inferring relations. Details and formal proofs can be found in our previous work on this topic (Pahikkala et al. 2010b; Waegeman et al. 2012).

3.1 Arbitrary relations

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

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

where ϕ represents the feature mapping for individual nodes and $\otimes$ denotes the Kronecker product. As shown by Ben-Hur and Noble (2005), such a pairwise feature mapping yields the Kronecker product pairwise kernel in the dual model:

$$K^{\Phi}_{\otimes}(e, \overline{e}) = K^{\Phi}_{\otimes}\big(v, v', \overline{v}, \overline{v}'\big) = K^{\phi}(v, \overline{v}) K^{\phi}\big(v', \overline{v}'\big), \tag{7}$$

with K^{ϕ} the kernel corresponding to ϕ.

It can be formally proven that with an appropriate choice of the node kernel K^{ϕ}, such as the Gaussian RBF kernel, the RKHS of the corresponding Kronecker product edge kernel

K^{Φ} allows approximating arbitrarily closely any relation that corresponds to a continuous function from $\mathcal{V}^2$ to $\mathbb{R}$. Before summarizing this important result, we recollect the definition of universal kernels.

Definition 1 (Steinwart 2002) A continuous kernel K on a compact metric space $\mathcal{X}$ (i.e. $\mathcal{X}$ is closed and bounded) is called universal if the RKHS induced by K is dense in $C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of all continuous functions $f : \mathcal{X} \to \mathbb{R}$.

Accordingly, the hypothesis space induced by the kernel K can approximate any function in $C(\mathcal{X})$ arbitrarily well, and hence it has the universal approximating property.

Theorem 1 (Waegeman et al. 2012) *Let us assume that the space of nodes $\mathcal{V}$ is a compact metric space. If a continuous kernel K^{ϕ} is universal on $\mathcal{V}$, then $K^{\Phi}_{\otimes}$ defines a universal kernel on $\mathcal{V}^2$.*

The proof is based on the so-called Stone-Weierstraß theorem (see e.g. Rudin 1991). The above result is interesting because it shows, given that an appropriate loss is optimized and a universal kernel applied on the node level K^{ϕ}, that the Kronecker product pairwise kernel has the ability to assure universal consistency, guaranteeing that the expected prediction error converges to its the lowest possible value when the amount of training data approaches infinity. We refer to Steinwart and Christmann (2008) for a more detailed discussion on the relationship between universal kernels and consistency. As a consequence, the Kronecker product kernel can always be considered a valid choice for learning relations if no specific a priori information other than a kernel for the nodes is provided about the relation that underlies the data. However, we would like to emphasize that Theorem 1 does not guarantee anything about the speed of the convergence or how large training sets are required for approximating the function closely enough. As a rule of thumb, whenever we have an access to useful prior information about the relation to be learned, it is beneficial to restrict the expressive power of the hypothesis space accordingly. The following two sections illustrate this more in detail for two particular types of relational domain knowledge: symmetry and reciprocity.

3.2 Symmetric relations

Symmetric relations form an important subclass of relations in our framework. As a specific type of symmetric relations, similarity relations constitute the underlying relation in many application domains where relations between objects need to be learned. Symmetric relations are formally defined as follows.

Definition 2 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 $Q(v, v') = Q(v', v)$.

More generally, symmetry can be defined for real-valued relations analogously as follows.

Definition 3 A binary relation $h : \mathcal{V}^2 \to \mathbb{R}$ is called a symmetric relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $h(v, v') = h(v', v)$.

For symmetric relations, edges in multi-graphs like Fig. 1 become undirected. Applications arise in many domains and metric learning or learning similarity measures can be seen as special cases. If the relation is 2-valued as $Q : \mathcal{V}^2 \to \{0, 1\}$, then we end up with a classification setting instead of a regression setting.

Symmetry can be easily incorporated in our framework via the following modification of the Kronecker kernel:

$$K^{\Phi}_{\otimes S}(e, \overline{e}) = \frac{1}{2}\big(K^{\phi}(v, \overline{v})K^{\phi}\big(v', \overline{v}'\big) + K^{\phi}\big(v, \overline{v}'\big)K^{\phi}\big(v', \overline{v}\big)\big). \tag{8}$$

The symmetric Kronecker kernel has been previously used for predicting protein-protein interactions in bioinformatics (Ben-Hur and Noble 2005). The following theorem shows that the RKHS of the symmetric Kronecker kernel can approximate arbitrarily well any type of continuous symmetric relation.

Theorem 2 (Waegeman et al. 2012) *Let*

$$S\big(\mathcal{V}^2\big) = \big\{t \mid t \in C\big(\mathcal{V}^2\big), t\big(v, v'\big) = t\big(v', v\big)\big\}$$

be the space of all continuous symmetric relations from $\mathcal{V}^2$ *to* $\mathbb{R}$. *If* K^{ϕ} *on* $\mathcal{V}$ *is universal, then the RKHS induced by the kernel* $K^{\Phi}_{\otimes S}$ *defined in* (8) *is dense in* $S(\mathcal{V}^2)$.

In other words the above theorem states that using the symmetric Kronecker product kernel is a way to incorporate the prior knowledge about the symmetry of the relation to be learned by only sacrificing the unnecessary expressive power. Thus, consistency can still be assured, despite considering a smaller hypothesis space.

3.3 Reciprocal relations

Let us start with a definition of this type of relation.

Definition 4 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 general real-valued relations, the notion of antisymmetry can be used in place of reciprocity:

Definition 5 A binary relation $h : \mathcal{V}^2 \to \mathbb{R}$ is called an antisymmetric relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $h(v, v') = -h(v', v)$.

For reciprocal and antisymmetric relations, every edge $e = (v, v')$ in a multi-graph like Fig. 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, et cetera. 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 Figs. 1(a) and (e) should be interpreted as edges starting from A instead of C. If the relation is 3-valued 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. Analogously to symmetry, reciprocity can also be easily incorporated in our framework via the following modification of the Kronecker kernel:

$$K^{\Phi}_{\otimes R}(e,\overline{e}) = \frac{1}{2}\big(K^{\phi}(v,\overline{v})K^{\phi}\big(v',\overline{v}'\big) - K^{\phi}\big(v,\overline{v}'\big)K^{\phi}\big(v',\overline{v}\big)\big). \tag{9}$$

Thus, the addition of kernels in the symmetric case becomes a subtraction of kernels in the reciprocal case. One can also prove that the RKHS of this so-called reciprocal Kronecker kernel allows approximating arbitrarily well any type of continuous reciprocal relation.

Theorem 3 (Waegeman et al. 2012) *Let*

$$R\big(\mathcal{V}^2\big) = \big\{t \mid t \in C\big(\mathcal{V}^2\big), t\big(v,v'\big) = -t\big(v',v\big)\big\}$$

be the space of all continuous antisymmetric relations from $\mathcal{V}^2$ *to* $\mathbb{R}$. *If* K^{ϕ} *on* $\mathcal{V}$ *is universal, then the RKHS induced by the kernel* $K^{\Phi}_{\otimes R}$ *defined in* (9) *is dense in* $R(\mathcal{V}^2)$.

Unlike many existing kernel-based methods for relational data, the models obtained with the presented kernels are able to represent any symmetric or reciprocal relation, respectively, without imposing additional transitivity properties of the relations.

4 Algorithmic aspects

This section gives a detailed description of the different algorithms that we propose for conditional ranking tasks. Our algorithms are primarily based on solving specific systems of linear equations, in which domain knowledge about the underlying relations is taken into account. In addition, a detailed discussion about the differences between optimizing (5) and (6) is provided.

4.1 Matrix representation of symmetric and reciprocal kernels

Let us define the so-called commutation matrix, which provides a powerful tool for formalizing the kernel matrices corresponding to the symmetric and reciprocal kernels.

Definition 6 (Commutation matrix) The $s^2 \times s^2$-matrix

$$\mathbf{P}^{s^2} = \sum_{i=1}^{s}\sum_{j=1}^{s} \mathbf{e}_{(i-1)s+j}\mathbf{e}^{\mathrm{T}}_{(j-1)s+i}$$

is called the commutation matrix (Abadir and Magnus 2005), where $\mathbf{e}_i$ are the standard basis vectors of $\mathbb{R}^{s^2}$.

We use the superscript s^2 to indicate the dimension $s^2 \times s^2$ of the matrix $\mathbf{P}$ but we omit this notation when the dimensionality is clear from the context or when the considerations do not depend on the dimensionality. For $\mathbf{P}$, we have the following properties. First, $\mathbf{PP} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, since $\mathbf{P}$ is a symmetric permutation matrix. Moreover, for every square matrix $\mathbf{M} \in \mathbb{R}^{s\times s}$, we have $\mathbf{P}\mathrm{vec}(\mathbf{M}) = \mathrm{vec}(\mathbf{M}^{\mathrm{T}})$, where vec is the column

vectorizing operator that stacks the columns of an $s \times s$-matrix in an s^2-dimensional column vector, that is,

$$\operatorname{vec}(\mathbf{M}) = (\mathbf{M}_{1,1}, \mathbf{M}_{2,1}, \ldots, \mathbf{M}_{s,1}, \mathbf{M}_{1,2}, \ldots, \mathbf{M}_{s,s})^{\mathrm{T}}. \tag{10}$$

Furthermore, for $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{s \times t}$, we have

$$\mathbf{P}^{s^2}(\mathbf{M} \otimes \mathbf{N}) = (\mathbf{N} \otimes \mathbf{M})\mathbf{P}^{t^2}.$$

The commutation matrix is used as a building block in constructing the following types of matrices:

Definition 7 (Symmetrizer and skew-symmetrizer matrices) The matrices

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

are known as the symmetrizer and skew-symmetrizer matrix, respectively (Abadir and Magnus 2005).

Armed with the above definitions, we will now 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. Note that the next proposition covers also the kernel matrices constructed between, say, nodes encountered in the training set and the nodes encountered at the prediction phase, and hence the considerations involve two different sets of nodes.

Proposition 1 *Let $\mathbf{K} \in \mathbb{R}^{r \times p}$ be a kernel matrix consisting of all kernel evaluations between nodes in sets $V \subseteq \mathcal{V}$ and $\overline{V} \subseteq \mathcal{V}$, with $|V| = r$ and $|\overline{V}| = p$, that is, $\mathbf{K}_{i,j} = K^{\phi}(v_i, \overline{v}_j)$, where $v_i \in V$ and $\overline{v}_j \in \overline{V}$. The ordinary, symmetric and reciprocal Kronecker kernel matrices consisting of all kernel evaluations between edges in $V \times V$ and edges in $\overline{V} \times \overline{V}$ are given by*

$$\overline{\mathbf{K}} = \mathbf{K} \otimes \mathbf{K}, \qquad \overline{\mathbf{K}}^{S} = \mathbf{S}^{r^2}(\mathbf{K} \otimes \mathbf{K}), \qquad \overline{\mathbf{K}}^{R} = \mathbf{A}^{r^2}(\mathbf{K} \otimes \mathbf{K}).$$

Proof The claim concerning the ordinary Kronecker kernel is an immediate consequence of the definition of the Kronecker product, that is, the entries of $\overline{\mathbf{K}}$ are given as

$$\overline{\mathbf{K}}_{(h-1)r+i,(j-1)p+k} = K^{\phi}(v_h, \overline{v}_j)K^{\phi}(v_i, \overline{v}_k),$$

where $1 \leq h, i \leq r$ and $1 \leq j, k \leq p$. To prove the other two claims, we pay closer attention to the entries of $\mathbf{K} \otimes \mathbf{K}$. In particular, the $((j-1)p + k)$-th column of $\mathbf{K} \otimes \mathbf{K}$ contains all kernel evaluations of the edges in $V \times V$ with the edge $(\overline{v}_j, \overline{v}_k) \in \overline{V} \times \overline{V}$. By definition (10) of vec, this column can be written as $\operatorname{vec}(\mathbf{M})$, where $\mathbf{M} \in \mathbb{R}^{r \times r}$ is a matrix whose i, h-th entry contains the kernel evaluation between the edges $(v_h, v_i) \in V \times V$ and $(\overline{v}_j, \overline{v}_k) \in \overline{V} \times \overline{V}$:

$$K^{\phi}(v_h, \overline{v}_j)K^{\phi}(v_i, \overline{v}_k).$$

We have the following properties of the symmetrizer and skew-symmetrizer matrices that straightforwardly follow from those of the commutation matrix. For any $\mathbf{M} \in \mathbb{R}^{s \times s}$

$$\begin{aligned} \mathbf{S}\operatorname{vec}(\mathbf{M}) &= \frac{1}{2}\operatorname{vec}\big(\mathbf{M} + \mathbf{M}^{\mathrm{T}}\big), \\ \mathbf{A}\operatorname{vec}(\mathbf{M}) &= \frac{1}{2}\operatorname{vec}\big(\mathbf{M} - \mathbf{M}^{\mathrm{T}}\big). \end{aligned} \tag{11}$$

Thus, the $((j-1)p+k)$-th column of $\mathbf{S}^{r^2}(\mathbf{K}\otimes\mathbf{K})$ can, according to (11), be written as $\frac{1}{2}\mathrm{vec}(\mathbf{M}+\mathbf{M}^{\mathrm{T}})$, where the i,h-th entry of $\mathbf{M}+\mathbf{M}^{\mathrm{T}}$ contains the kernel evaluation

$$\frac{1}{2}\big(K^{\phi}(v_h,\overline{v}_j)K^{\phi}(v_i,\overline{v}_k)+K^{\phi}(v_i,\overline{v}_j)K^{\phi}(v_h,\overline{v}_k)\big),$$

which corresponds to the symmetric Kronecker kernel between the edges $(v_i,v_h)\in V\times V$ and $(\overline{v}_j,\overline{v}_k)\in\overline{V}\times\overline{V}$. The reciprocal case is analogous. □

We also note that for $\mathbf{M}\in\mathbb{R}^{s\times t}$ the symmetrizer and skew-symmetrizer matrices commute with the $s^2\times t^2$-matrix $\mathbf{M}\otimes\mathbf{M}$ in the following sense:

$$\mathbf{S}^{s^2}(\mathbf{M}\otimes\mathbf{M})=(\mathbf{M}\otimes\mathbf{M})\mathbf{S}^{t^2}, \tag{12}$$

$$\mathbf{A}^{s^2}(\mathbf{M}\otimes\mathbf{M})=(\mathbf{M}\otimes\mathbf{M})\mathbf{A}^{t^2}, \tag{13}$$

where $\mathbf{S}^{s^2}$ and $\mathbf{A}^{s^2}$ in the left-hand sides are $s^2\times s^2$-matrices and $\mathbf{S}^{t^2}$ and $\mathbf{A}^{t^2}$ are $t^2\times t^2$-matrices in the right-hand sides. Thus, due to (12), the above-considered symmetric Kronecker kernel matrix may as well be written as $(\mathbf{K}\otimes\mathbf{K})\mathbf{S}^{p^2}$ or as $\mathbf{S}^{r^2}(\mathbf{K}\otimes\mathbf{K})\mathbf{S}^{p^2}$. The same applies to the reciprocal Kronecker kernel matrix due to (13).

4.2 Regression with symmetric and reciprocal kernels

Let p and q, respectively, represent the number of nodes and edges in T. In the following, we make an assumption that T contains, for each ordered pair of nodes (v,v'), exactly one edge starting from v and ending to v', that is, $q=p^2$ and T corresponds to a complete directed graph on p nodes which includes a loop at each node. As we will show below, this important special case enables the use of many computational short-cuts for the training phase. This assumption is dropped in Sect. 4.4, where we present training algorithms for the more general case.

In practical applications, a fully connected graph is most commonly available in settings where the edges are generated by comparing some direct property of the nodes, such as whether they belong to same or similar class in a classification taxonomy (for examples, see the experiments). In experimental research on small sample data, such as commonly considered in many bioinformatics applications (see e.g. Park and Marcotte 2012), it may also be feasible to gather the edge information directly for all pairs through experimental comparisons. If this is not the case, then imputation techniques can be used to fill in the values for missing edges in the training data in case only a small minority is missing.

Using the notation of Proposition 1, we let $\mathbf{K}\in\mathbb{R}^{p\times p}$ be the kernel matrix of K^{ϕ}, containing similarities between all nodes encountered in T. Due to the above assumption and Proposition 1, the kernel matrix containing the evaluations of the kernels $K^{\Phi}_{\otimes}$, $K^{\Phi S}_{\otimes}$ and $K^{\Phi R}_{\otimes}$ between the edges in T can be expressed as $\overline{\mathbf{K}}$, $\overline{\mathbf{K}}^{S}$ and $\overline{\mathbf{K}}^{R}$, respectively.

Recall that, according to the representer theorem, the prediction function obtained as a solution to problem (1) can be expressed with the dual representation (2), involving a vector of so-called dual parameters, whose dimension equals the number of edges in the training set. Here, we represent the dual solution with a vector $\mathbf{a}\in\mathbb{R}^{p^2}$ containing one entry per each possible edge between the vertices occurring in the training set.

Thus, using standard Tikhonov regularization (Evgeniou et al. 2000), the objective function of problem (1) with kernel $K^{\Phi}_{\otimes}$ can be rewritten in matrix notation as

$$\mathcal{L}(\mathbf{y},\overline{\mathbf{K}}\mathbf{a})+\lambda\mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\mathbf{a}, \tag{14}$$

where $\mathcal{L} : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ is a convex loss function that maps the vector $\mathbf{y}$ of training labels and the vector $\overline{\mathbf{K}}\mathbf{a}$ of predictions to a real value.

Up to multiplication with a constant, the loss (5) can be represented in matrix form as

$$(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}). \tag{15}$$

Thus, for the regression approach, the objective function to be minimized becomes:

$$(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}) + \lambda \mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\mathbf{a}. \tag{16}$$

By taking the derivative of (16) with respect to $\mathbf{a}$, setting it to zero, and solving with respect to $\mathbf{a}$, we get the following system of linear equations:

$$(\overline{\mathbf{K}}\overline{\mathbf{K}} + \lambda\overline{\mathbf{K}})\mathbf{a} = \overline{\mathbf{K}}\mathbf{y}. \tag{17}$$

If the kernel matrix $\overline{\mathbf{K}}$ is not strictly positive definite but only positive semi-definite, $\overline{\mathbf{K}}$ should be interpreted as $\lim_{\epsilon \to 0^+}(\overline{\mathbf{K}} + \epsilon\mathbf{I})$. Accordingly, (17) can be simplified to

$$(\overline{\mathbf{K}} + \lambda\mathbf{I})\mathbf{a} = \mathbf{y}. \tag{18}$$

Due to the positive semi-definiteness of the kernel matrix, (18) always has a unique solution. Since the solution of (18) is also a solution of (17), it is enough to concentrate on solving (18).

Using the standard notation and rules of Kronecker product algebra, we show how to efficiently solve shifted Kronecker product systems. For a more in-depth analysis of the shifted Kronecker product systems, we refer to Martin and Van Loan (2006).

Proposition 2 *Let* $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{p \times p}$ *be diagonalizable matrices, that is, the matrices can be eigen decomposed as*

$$\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}, \qquad \mathbf{N} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{-1},$$

where $\mathbf{V}, \mathbf{U} \in \mathbb{R}^{p \times p}$ *contain the eigenvectors and the diagonal matrices* $\boldsymbol{\Lambda}, \boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ *contain the corresponding eigenvalues of* $\mathbf{M}$ *and* $\mathbf{N}$*. Then, the following type of shifted Kronecker product system*

$$(\mathbf{M} \otimes \mathbf{N} + \lambda\mathbf{I})\mathbf{a} = \mathrm{vec}(\mathbf{Y}), \tag{19}$$

where $\lambda > 0$ *and* $\mathbf{Y} \in \mathbb{R}^{p \times p}$*, can be solved with respect to* $\mathbf{a}$ *in* $O(p^3)$ *time if the inverse of* $\mathbf{M} \otimes \mathbf{N} + \lambda\mathbf{I}$ *exists.*

Proof Before starting the actual proof, we recall certain rules concerning the Kronecker product (see e.g. Horn and Johnson 1991) and introduce some notations. Namely, for $\mathbf{M} \in \mathbb{R}^{a \times b}$, $\mathbf{U} \in \mathbb{R}^{c \times d}$, $\mathbf{N} \in \mathbb{R}^{b \times s}$ and $\mathbf{V} \in \mathbb{R}^{d \times t}$, we have:

$$(\mathbf{M} \otimes \mathbf{U})(\mathbf{N} \otimes \mathbf{V}) = (\mathbf{M}\mathbf{N}) \otimes (\mathbf{U}\mathbf{V}).$$

From this, it directly follows that

$$(\mathbf{M} \otimes \mathbf{N})^{-1} = \mathbf{M}^{-1} \otimes \mathbf{N}^{-1}. \tag{20}$$

Moreover, for $\mathbf{M} \in \mathbb{R}^{a \times b}$, $\mathbf{N} \in \mathbb{R}^{b \times c}$, and $\mathbf{U} \in \mathbb{R}^{c \times d}$, we have:

$$\big(\mathbf{U}^{\mathrm{T}} \otimes \mathbf{M}\big)\mathrm{vec}(\mathbf{N}) = \mathrm{vec}(\mathbf{M}\mathbf{N}\mathbf{U}).$$

Furthermore, for $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{a \times b}$, let $\mathbf{M} \odot \mathbf{N}$ denote the Hadamard (elementwise) product, that is, $(\mathbf{M} \odot \mathbf{N})_{i,j} = \mathbf{M}_{i,j}\mathbf{N}_{i,j}$. Further, for a vector $\mathbf{v} \in \mathbb{R}^s$, let $\operatorname{diag}(\mathbf{v})$ denote the diagonal $s \times s$-matrix, whose diagonal entries are given as $\operatorname{diag}(\mathbf{v})_{i,i} = \mathbf{v}_i$. Finally, for $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{a \times b}$, we have:

$$\operatorname{vec}(\mathbf{M} \odot \mathbf{N}) = \operatorname{diag}\big(\operatorname{vec}(\mathbf{M})\big)\operatorname{vec}(\mathbf{N}).$$

By multiplying both sides of Eq. (19) with $(\mathbf{M} \otimes \mathbf{N} + \lambda\mathbf{I})^{-1}$ from the left, we get

$$\begin{aligned}
\mathbf{a} &= (\mathbf{M} \otimes \mathbf{N} + \lambda\mathbf{I})^{-1}\operatorname{vec}(\mathbf{Y}) \\
&= \big((\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}) \otimes (\mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{-1}) + \lambda\mathbf{I}\big)^{-1}\operatorname{vec}(\mathbf{Y}) \\
&= \big((\mathbf{V} \otimes \mathbf{U})(\boldsymbol{\Lambda} \otimes \boldsymbol{\Sigma})(\mathbf{V}^{-1} \otimes \mathbf{U}^{-1}) + \lambda\mathbf{I}\big)^{-1}\operatorname{vec}(\mathbf{Y}) \\
&= (\mathbf{V} \otimes \mathbf{U})(\boldsymbol{\Lambda} \otimes \boldsymbol{\Sigma} + \lambda\mathbf{I})^{-1}(\mathbf{V}^{-1} \otimes \mathbf{U}^{-1})\operatorname{vec}(\mathbf{Y}) \qquad (21) \\
&= (\mathbf{V} \otimes \mathbf{U})(\boldsymbol{\Lambda} \otimes \boldsymbol{\Sigma} + \lambda\mathbf{I})^{-1}\operatorname{vec}\big(\mathbf{U}^{-1}\mathbf{Y}\mathbf{V}^{-\mathrm{T}}\big) \\
&= (\mathbf{V} \otimes \mathbf{U})\operatorname{vec}(\mathbf{C} \odot \mathbf{E}) \\
&= \operatorname{vec}\big(\mathbf{U}(\mathbf{C} \odot \mathbf{E})\mathbf{V}^{\mathrm{T}}\big), \qquad (22)
\end{aligned}$$

where $\mathbf{E} = \mathbf{U}^{-1}\mathbf{Y}\mathbf{V}^{-\mathrm{T}}$ and $\operatorname{diag}(\operatorname{vec}(\mathbf{C})) = (\boldsymbol{\Lambda} \otimes \boldsymbol{\Sigma} + \lambda\mathbf{I})^{-1}$. In line (21), we use (20) and therefore we can write $\lambda\mathbf{I} = \lambda(\mathbf{V} \otimes \mathbf{U})(\mathbf{V}^{-1} \otimes \mathbf{U}^{-1})$ after which we can add λ directly to the eigenvalues $\boldsymbol{\Lambda} \otimes \boldsymbol{\Sigma}$ of $\mathbf{M} \otimes \mathbf{N}$. The eigen decompositions of $\mathbf{M}$ and $\mathbf{N}$ as well as all matrix multiplications in (22) can be computed in $O(p^3)$ time. □

Corollary 1 *A minimizer of* (16) *can be computed in* $O(p^3)$ *time.*

Proof Since the kernel matrix $\mathbf{K}$ is symmetric and positive semi-definite, it is diagonalizable and it has nonnegative eigenvalues. This ensures that the matrix $\mathbf{K} \otimes \mathbf{K} + \lambda\mathbf{I}$ has strictly positive eigenvalues and therefore its inverse exists. Consequently, the claim follows directly from Proposition 2, which can be observed by substituting $\mathbf{K}$ for both $\mathbf{M}$ and $\mathbf{N}$. □

We continue by considering the use of the symmetric and reciprocal Kronecker kernels and show that, with those, the dual solution can be obtained as easily as with the ordinary Kronecker kernel. We first present and prove the following two inversion identities:

Lemma 1 *Let* $\overline{\mathbf{N}} = \mathbf{N} \otimes \mathbf{N}$ *for some square matrix* $\mathbf{N}$. *Then,*

$$(\mathbf{S}\overline{\mathbf{N}}\mathbf{S} + \lambda\mathbf{I})^{-1} = \mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\mathbf{S} + \frac{1}{\lambda}\mathbf{A}, \qquad (23)$$

$$(\mathbf{A}\overline{\mathbf{N}}\mathbf{A} + \lambda\mathbf{I})^{-1} = \mathbf{A}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\mathbf{A} + \frac{1}{\lambda}\mathbf{S}, \qquad (24)$$

if the considered inverses exist.

Proof For a start, we note certain directly verifiable properties of the symmetrizer and skew-symmetrizer matrices. Namely, the matrices $\mathbf{S}$ and $\mathbf{A}$ are idempotent, that is,

$$\mathbf{S}\mathbf{S} = \mathbf{S} \quad \text{and} \quad \mathbf{A}\mathbf{A} = \mathbf{A}.$$

Furthermore, $\mathbf{S}$ and $\mathbf{A}$ are orthogonal to each other, that is,

$$\mathbf{SA} = \mathbf{AS} = \mathbf{0}. \tag{25}$$

We prove (23) by multiplying $\mathbf{S}\overline{\mathbf{N}}\mathbf{S} + \lambda\mathbf{I}$ with its alleged inverse matrix and show that the result is the identity matrix:

$$\begin{aligned}
&(\mathbf{S}\overline{\mathbf{N}}\mathbf{S} + \lambda\mathbf{I})\left(\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\mathbf{S} + \frac{1}{\lambda}\mathbf{A}\right) \\
&= \mathbf{S}\overline{\mathbf{N}}\mathbf{S}\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\mathbf{S} + \frac{1}{\lambda}\mathbf{S}\overline{\mathbf{N}}\mathbf{S}\mathbf{A} \\
&\quad + \lambda\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\mathbf{S} + \lambda\frac{1}{\lambda}\mathbf{A} \qquad (26) \\
&= \mathbf{S}\overline{\mathbf{N}}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1} + \lambda\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1} + \mathbf{A} \qquad (27) \\
&= \mathbf{S}\big(\mathbf{I} - \lambda(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}\big) + \lambda\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1} + \mathbf{A} \qquad (28) \\
&= \mathbf{S} - \lambda\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1} + \lambda\mathbf{S}(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1} + \mathbf{A} \\
&= \mathbf{I}.
\end{aligned}$$

When going from (26) to (27), we use the fact that $\mathbf{S}$ commutes with $(\overline{\mathbf{N}} + \lambda\mathbf{I})^{-1}$, because it commutes with both $\overline{\mathbf{N}}$ and $\mathbf{I}$. Moreover, the second term of (26) vanishes, because of the orthogonality of $\mathbf{S}$ and $\mathbf{A}$ to each other. In (28) we have used the following inversion identity known in matrix calculus literature (Henderson and Searle 1981)

$$\overline{\mathbf{N}}(\overline{\mathbf{N}} + \mathbf{I})^{-1} = \mathbf{I} - (\overline{\mathbf{N}} + \mathbf{I})^{-1}.$$

Identity (24) can be proved analogously. □

These inversion identities indicate that we can invert a diagonally shifted symmetric or reciprocal Kronecker kernel matrix simply by modifying the inverse of a diagonally shifted ordinary Kronecker kernel matrix. This is an advantageous property, since the computational short-cuts provided by Proposition 2 ensure the fast inversion of the shifted ordinary Kronecker kernel matrices, and its results can thus be used to accelerate the computations for the symmetric and reciprocal cases too.

The next result uses the above inversion identities to show that, when learning symmetric or reciprocal relations with kernel ridge regression (Saunders et al. 1998; Suykens et al. 2002; Shawe-Taylor and Cristianini 2004; Pahikkala et al. 2009), we do not explicitly have to use the symmetric and reciprocal Kronecker kernels. Instead, we can just use the ordinary Kronecker kernel to learn the desired model as long as we ensure that the symmetry or reciprocity is encoded in the labels.

Proposition 3 *Using the symmetric Kronecker kernel for RLS regression with a label vector* $\mathbf{y}$ *is equivalent to using an ordinary Kronecker kernel and a label vector* $\mathbf{Sy}$. *One can observe an analogous relationship between the reciprocal Kronecker kernel and a label vector* $\mathbf{Ay}$.

Proof Let

$$\begin{aligned}
\mathbf{a} &= (\mathbf{S}\overline{\mathbf{K}}\mathbf{S} + \lambda\mathbf{I})^{-1}\mathbf{y}, \\
\mathbf{b} &= (\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{Sy}
\end{aligned}$$

be solutions of (16) with the symmetric Kronecker kernel and label vector $\mathbf{y}$ and with the ordinary Kronecker kernel and label vector $\mathbf{Sy}$, respectively. Using identity (23), we get

$$\begin{aligned}\mathbf{a} &= \left(\mathbf{S}(\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{S} + \frac{1}{\lambda}\mathbf{A}\right)\mathbf{y} \\ &= (\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{Sy} + \frac{1}{\lambda}\mathbf{Ay}.\end{aligned}$$

In the last equality, we again used the fact that $\mathbf{S}$ commutes with $(\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}$, because it commutes with both $\overline{\mathbf{K}}$ and $\mathbf{I}$. Let (v, v') be a new couple of nodes for which we are supposed to do a prediction with a regressor determined by the coefficients $\mathbf{a}$. Moreover, let $\mathbf{k}_v, \mathbf{k}_{v'} \in \mathbb{R}^p$ denote, respectively, the base kernel K^ϕ evaluations of the nodes v and v' with the nodes in the training data. Then, $\mathbf{k}_v \otimes \mathbf{k}_{v'} \in \mathbb{R}^q$ contains the Kronecker kernel $K^\Phi_\otimes$ evaluations of the edge (v, v') with all edges in the training data. Further, according to Proposition 1, the corresponding vector of symmetric Kronecker kernel evaluations is $\mathbf{S}(\mathbf{k}_v \otimes \mathbf{k}_{v'})$. Now, the prediction for the couple (v, v') can be expressed as

$$\begin{aligned}(\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{Sa} &= (\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{S}\left((\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{Sy} + \frac{1}{\lambda}\mathbf{Ay}\right) \\ &= (\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{S}(\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{Sy} \\ &\quad + \frac{1}{\lambda}(\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{SAy} \qquad (29) \\ &= (\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{S}(\overline{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{Sy} \\ &= (\mathbf{k}_v \otimes \mathbf{k}_{v'})^{\mathrm{T}}\mathbf{Sb},\end{aligned}$$

where term (29) vanishes due to (25). The analogous result for the reciprocal Kronecker kernel can be shown in a similar way. □

As a consequence of this, we also have a computationally efficient method for RLS regression with symmetric and reciprocal Kronecker kernels. Encoding the properties into the label matrix ensures that the corresponding variations of the Kronecker kernels are implicitly used.

4.3 Conditional ranking with symmetric and reciprocal kernels

Now, we show how loss function (6) can be represented in matrix form. This representation is similar to the RankRLS loss introduced by Pahikkala et al. (2009). Let

$$\mathbf{C}^l = \mathbf{I} - \frac{1}{l}\mathbf{1}^l\mathbf{1}^{l\mathrm{T}}, \qquad (30)$$

where $l \in \mathbb{N}$, $\mathbf{I}$ is the $l \times l$-identity matrix, and $\mathbf{1}^l \in \mathbb{R}^l$ is the vector of which every entry is equal to 1, be the $l \times l$-centering matrix. The matrix $\mathbf{C}^l$ is an idempotent matrix and multiplying it with a vector subtracts the mean of the vector entries from all elements of the vector. Moreover, the following equality can be shown

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

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

$$\mathbf{L} = \begin{pmatrix} \mathbf{C}^{l_1} & & \\ & \ddots & \\ & & \mathbf{C}^{l_p} \end{pmatrix}, \tag{31}$$

where l_i is the number of edges starting from v_i for $i \in \{1, \ldots, p\}$. Again, given the assumption that the training data contains all possible edges between the nodes exactly once and hence $l_i = p$ for all $1 \leq i \leq p$, loss function (6) can be, up to multiplication with a constant, represented in matrix form as

$$(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}), \tag{32}$$

provided that the entries of $\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}$ are ordered in a way compatible with the entries of $\mathbf{L}$, that is, the training edges are arranged according to their starting nodes.

Analogously to the regression case, the training phase corresponds to solving the following system of linear equations:

$$\big(\overline{\mathbf{K}}^{\mathrm{T}}\mathbf{L}\overline{\mathbf{K}} + \lambda\overline{\mathbf{K}}\big)\mathbf{a} = \overline{\mathbf{K}}^{\mathrm{T}}\mathbf{L}\mathbf{y}. \tag{33}$$

If the ordinary Kronecker kernel is used, we get a result analogous to Corollary 1.

Corollary 2 *A solution of* (33) *can be computed in* $O(p^3)$ *time.*

Proof Given that $l_i = p$ for all $1 \leq i \leq p$ and that the ordinary Kronecker kernel is used, matrix (31) can be written as $(\mathbf{I} \otimes \mathbf{C}^p)$ and the system of linear equations (33) becomes:

$$\begin{aligned} &\big((\mathbf{K} \otimes \mathbf{K})\big(\mathbf{I} \otimes \mathbf{C}^p\big)(\mathbf{K} \otimes \mathbf{K}) + \lambda\mathbf{K} \otimes \mathbf{K}\big)\mathbf{a} \\ &\quad = (\mathbf{K} \otimes \mathbf{K})\big(\mathbf{I} \otimes \mathbf{C}^p\big)\mathbf{y}. \end{aligned}$$

While the kernel matrix $\mathbf{K} \otimes \mathbf{K}$ is not necessarily invertible, a solution can still be obtained from the following reduced form:

$$\big((\mathbf{K} \otimes \mathbf{K})\big(\mathbf{I} \otimes \mathbf{C}^p\big) + \lambda\mathbf{I}\big)\mathbf{a} = \big(\mathbf{I} \otimes \mathbf{C}^p\big)\mathbf{y}.$$

This can, in turn, be rewritten as

$$\big(\mathbf{K} \otimes \mathbf{K}\mathbf{C}^p + \lambda\mathbf{I}\big)\mathbf{a} = \big(\mathbf{I} \otimes \mathbf{C}^p\big)\mathbf{y}. \tag{34}$$

The matrix $\mathbf{C}^p$ is symmetric, and hence if $\mathbf{K}$ is strictly positive definite, the product $\mathbf{K}\mathbf{C}^p$ is diagonalizable and has nonnegative eigenvalues (see e.g. Horn and Johnson 1985, p. 465). Therefore, (34) is of the form which can be solved in $O(p^3)$ time due to Proposition 2. The situation is more involved if $\mathbf{K}$ is positive semi-definite. In this case, we can solve the so-called primal form with an empirical kernel map (see e.g. Airola et al. 2011a) instead of (34) and again end up with a Kronecker system solvable in $O(p^3)$ time. We omit the details of this consideration due to its lengthiness and technicality. □

4.4 Conjugate gradient-based training algorithms

Interestingly, if we use the symmetric or reciprocal Kronecker kernel for conditional ranking, we do not have a similar efficient closed-form solution as those indicated by Corollaries 1 and 2. The same concerns both regression and ranking if the above assumption of the training data having every possible edge between all nodes encountered in the training data (i.e. $l_i = p$ for all $1 \leq i \leq p$) is dropped. Fortunately, we can still design algorithms that take advantage of the special structure of the kernel matrices and the loss function in speeding up the training process, while they are not as efficient as the above-described closed-form solutions.

Before proceeding, we introduce some extra notation. 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 data 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 edge corresponds. This matrix covers both the situation in which some of the possible edges are not in the training data and the one in which there are several edges adjacent to the same nodes. Objective function (14) can be written as

$$\mathcal{L}(\mathbf{y}, \mathbf{B}\overline{\mathbf{K}}\mathbf{a}) + \lambda \mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\mathbf{a}$$

with the ordinary Kronecker kernel and analogously with the symmetric and reciprocal kernels. Note that the number of dual variables stored in vector $\mathbf{a}$ is still equal to p^2, while the number of labels in $\mathbf{y}$ is equal to q. If an edge is not in the training data, the corresponding entry in $\mathbf{a}$ is zero, and if a particular edge occurs several times, the corresponding entry is the sum of the corresponding variables a_e in representation (2). For the ranking loss, the system of linear equations to be solved becomes

$$\left(\overline{\mathbf{K}}\mathbf{B}^{\mathrm{T}}\mathbf{L}\mathbf{B}\overline{\mathbf{K}} + \lambda\overline{\mathbf{K}}\right)\mathbf{a} = \overline{\mathbf{K}}\mathbf{B}^{\mathrm{T}}\mathbf{L}\mathbf{y}.$$

If we use an identity matrix instead of $\mathbf{L}$ in (32), the system corresponds to the regression loss.

To solve the above type of linear systems, we consider an approach based on conjugate gradient type of methods with early stopping regularization. The Kronecker product $(\mathbf{K} \otimes \mathbf{K})\mathbf{v}$ can be written as $\mathrm{vec}(\mathbf{KVK})$, where $\mathbf{v} = \mathrm{vec}(\mathbf{V}) \in \mathbb{R}^{p^2}$ and $\mathbf{V} \in \mathbb{R}^{p \times p}$. 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, because they contain only $O(p^2)$ nonzero elements. Similarly, the matrix $\mathbf{B}$ has only q non-zero elements. Finally, we observe from (30) and (31) that the matrix $\mathbf{L}$ can be written as $\mathbf{L} = \mathbf{I} - \mathbf{Q}\mathbf{Q}^{\mathrm{T}}$, where $\mathbf{Q} \in \mathbb{R}^{q \times p}$ is the following quasi-diagonal matrix:

$$\mathbf{Q} = \begin{pmatrix} \frac{1}{\sqrt{l_1}}\mathbf{1}^{l_1} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{l_p}}\mathbf{1}^{l_p} \end{pmatrix}. \tag{35}$$

The matrices $\mathbf{I}$ and $\mathbf{Q}$ both have $O(q)$ nonzero entries, and hence multiplying a vector with the matrix $\mathbf{L}$ can also be performed in $O(q)$ time.

Conjugate gradient methods require, in the worst case, $O(p^4)$ iterations in order to solve the system of linear equations (33) under consideration. However, the number of iterations required in practice is a small constant, as we will show in the experiments. In addition, since

using early stopping with gradient-based methods has a regularizing effect on the learning process (see e.g. Engl et al. 1996), this approach can be used instead of or together with Tikhonov regularization.

4.5 Theoretical considerations

Next, we give theoretical insights to back the idea of using RankRLS-based learning methods instead of ordinary RLS regression. As observed in Sect. 4.3, the main difference between RankRLS and the ordinary RLS is that RankRLS enforces the learned models to be block-wise centered, that is, the aim is to learn models that, for each node v, correctly predict the differences between the utility values of the edges (v, v') and (v, v''), rather than the utility values themselves. This is common for most of the pairwise learning to rank algorithms, since learning the individual utility values is, in ranking tasks, relevant only with relation to other utility values. Below, we consider whether the block-wise centering approach actually helps in achieving this aim. This is done via analyzing the regression performance of the utility value differences.

We start by considering the matrix forms of the objective functions of the ordinary RLS regression

$$J(\mathbf{a}) = (\mathbf{y} - \overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}) + \lambda \mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\mathbf{a} \tag{36}$$

and RankRLS for conditional ranking

$$F(\mathbf{a}) = (\mathbf{y} - \overline{\mathbf{K}}\mathbf{a})^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \overline{\mathbf{K}}\mathbf{a}) + \lambda \mathbf{a}^{\mathrm{T}}\overline{\mathbf{K}}\mathbf{a}, \tag{37}$$

where $\mathbf{L} \in \mathbb{R}^{q \times q}$ is a quasi-diagonal matrix whose diagonal blocks are $p \times p$-centering matrices. Here we make the further assumption that the label vector is block-wise centered, that is, $\mathbf{y} = \mathbf{L}\mathbf{y}$. We are always free to make this assumption with conditional ranking tasks.

The following lemma indicates that we can consider the RankRLS problem as an ordinary RLS regression problem with a modified kernel.

Lemma 2 *Objective functions* (37) *and*

$$W(\mathbf{a}) = (\mathbf{y} - \mathbf{L}\overline{\mathbf{K}}\mathbf{L}\mathbf{a})^{\mathrm{T}}(\mathbf{y} - \mathbf{L}\overline{\mathbf{K}}\mathbf{L}\mathbf{a}) + \lambda \mathbf{a}^{\mathrm{T}}\mathbf{L}\overline{\mathbf{K}}\mathbf{L}\mathbf{a} \tag{38}$$

have a common minimizer.

Proof By repeatedly applying the idempotence of $\mathbf{L}$ and the inversion identities of (Henderson and Searle 1981), one of the solutions of (37) can be written as

$$\begin{aligned}
\mathbf{a} &= (\mathbf{L}\overline{\mathbf{K}} + \lambda \mathbf{I})^{-1}\mathbf{L}\mathbf{y} \\
&= (\mathbf{L}\mathbf{L}\overline{\mathbf{K}} + \lambda \mathbf{I})^{-1}\mathbf{L}\mathbf{y} \\
&= \mathbf{L}(\mathbf{L}\overline{\mathbf{K}}\mathbf{L} + \lambda \mathbf{I})^{-1}\mathbf{y} \\
&= \mathbf{L}(\mathbf{L}\overline{\mathbf{K}}\mathbf{L}\mathbf{L} + \lambda \mathbf{I})^{-1}\mathbf{y} \\
&= (\mathbf{L}\mathbf{L}\overline{\mathbf{K}}\mathbf{L} + \lambda \mathbf{I})^{-1}\mathbf{L}\mathbf{y} \\
&= (\mathbf{L}\overline{\mathbf{K}}\mathbf{L} + \lambda \mathbf{I})^{-1}\mathbf{y},
\end{aligned}$$

which is also a minimizer of (38). □

This lemma provides us a different perspective on RankRLS. Namely, if we have a prior knowledge that the underlying regression function to be learned is block-wise centered (i.e. we have a conditional ranking task), this knowledge is simply encoded into a kernel function, just like we do with the knowledge about the reciprocity and symmetry.

In the literature, there are many results (see e.g. De Vito et al. 2005 and references therein) indicating that the expected prediction error of the regularized least-squared based kernel regression methods obey the following type of probabilistic upper bounds. For simplicity, we only consider the regression error. Namely, for any $0 < \eta < 1$, it holds that

$$P\left[I[\widehat{f}_{\lambda,T}] - \inf_{f\in\mathcal{H}_K} I[f] \leq B(\lambda, K, \eta)\right] \geq 1 - \eta \tag{39}$$

where $P[\cdot]$ denotes the probability, $I[\cdot]$ is the expected prediction error, $\widehat{f}_{\lambda,T}$ is the prediction function obtained via regularized risk minimization on a training set T and a regularization parameter λ, $\mathcal{H}_K$ is the RKHS associated to the kernel K, and $B(\lambda, K, \eta)$ is a complexity term depending on the kernel, the amount of regularization, and the confidence level η.

According to Lemma 2, if the underlying regression function y is block-wise centered, which is the case in the conditional ranking tasks, we can consider learning with conditional RankRLS as performing regression with a block-wise centered kernel, and hence the behaviour of RLS regression and RankRLS can be compared with each other under the framework given in (39). When comparing the two kernels, we first have to pay attention to the corresponding RKHS constructions $\mathcal{H}_K$. The RKHS of the original kernel is more expressive than that of the block-wise centered kernel, because the former is able to express functions that are not block-wise centered while the latter can not. However, since we consider conditional ranking tasks, this extra expressiveness is of no help and the terms $\inf_{f\in\mathcal{H}_K} I[f]$ are equal for the two kernels.

Next, we focus our attention on the complexity term. A typical example of the term is the one proposed by De Vito et al. (2005), which is proportional to $\kappa = \sup_e K(e, e)$. Now, the quantity κ is lower for the block-wise centered kernel than for the original one, and hence the former has tighter error bounds than the latter. This, in turn, indicates that RankRLS is indeed a more promising approach for learning to predict the utility value differences than the ordinary RLS. It would be interesting to extend the analysis from the regression error to the pairwise ranking error itself but the analysis is far more challenging and it is considered as an open problem by De Vito et al. (2005).

5 Links with existing ranking methods

Examining the pairwise loss (4) 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. Joachims 2002; Freund et al. 2003; Pahikkala et al. 2009). 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 (Joachims 2002), which optimizes the pairwise hinge loss. Even much more closely related is the ranking regularized least-squares method RankRLS (Pahikkala et al. 2009), 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 means in practice 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 Sect. 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. Furthermore, 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 $p^2 \times p^2$ 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 (see e.g. Joachims 2006) 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^2d^2 non-zero entries. Again, this would be both memory-wise 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.

6 Experiments

In the experiments we consider conditional ranking tasks on synthetic and real-world data in various application domains, illustrating different aspects of the generality of our approach. The first experiment considers a potential application in game playing, using 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. The second experiment considers a potential application in information retrieval, using the 20-newsgroups data set. Here the task consists of ranking documents according to their similarity to any other document, on which the ranking is conditioned. The third experiment summarizes a potential application of identifying bacterial species in microbiology. The goal consists of retrieving a bipartite ranking for a given species, in which bacteria from the same species have to be ranked before bacteria from a different species. On both the newsgroups and bacterial data we test the capability of the models to generalize to such newsgroups or species that have not been observed during training.

In all the experiments, we run both the conditional ranker that minimizes the convex edgewise ranking loss approximation (6) and the method that minimizes the regression loss (5) over the edges. Furthermore, in the rock-paper-scissors experiment we also train a conditional ranker with RankSVM. For the 20-newsgroups and bacterial species data this is not possible due to the large number of edges present in the relational graph, resulting in too high memory requirements and computational costs for RankSVM training to be practical. We use the Kronecker kernel $K^{\Phi}_{\otimes}$ for edges in all the experiments. We also test the effects of enforcing domain knowledge by applying the reciprocal kernel $K^{\Phi}_{\otimes R}$ in the rock-paper-scissors experiment, and applying the symmetric kernel $K^{\Phi}_{\otimes S}$ in the 20-newsgroups and bacterial data experiments. The linear kernel is used for individual nodes (thus, for K^{ϕ}). In all the experiments, performance is measured using the ranking loss (4) on the test set.

We use a variety of approaches for minimizing the squared conditional ranking and regression losses, depending on the characteristics of the task. All the solvers based on optimizing the standard, or pairwise regularized least-squares loss are from the RLScore software package.[1] For the experiment where the training is performed iteratively, we apply the biconjugate gradient stabilized method (BGSM) (van der Vorst 1992). The RankSVM based conditional ranker baseline considered in the rock-paper-scissors experiment is trained with the TreeRankSVM software (Airola et al. 2011b).

6.1 Game playing: the rock-paper-scissors dataset

The synthetic benchmark data, whose generation process is described in detail by Pahikkala et al. (2010b), 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. In contrast, conditional ranking is a sensible task, where players are ranked according to their estimated probability of winning against a given player.

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 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 noticed that on this data set regularization seemed to be harmful, with methods typically reaching optimal performance for close to zero regularization parameter values. Further, cross-validation as a parameter selection strategy appeared to work very poorly, due to the small training set size and the large amount of noise present in the training data. Thus, we performed the runs using a fixed regularization parameter set to a close to zero value (2^{-30}).

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

Table 1 Overview of the measured rank loss for rock-paper-scissors. The abbreviations c.reg and c.rank here refer to the RLS and RankRLS algorithm, respectively, and (r) refers to the use of a reciprocal Kronecker kernel instead of the ordinary Kronecker kernel

	c.reg	c.reg (r)	c.rank	c.rank (r)	RankSVM	RankSVM (r)
$w1$	0.4875	0.4868	0.4876	0.4880	0.4987	0.4891
$w10$	0.04172	0.04145	0.04519	0.04291	0.04535	0.04116
$w100$	0.001380	0.001366	0.001424	0.001354	0.006997	0.005824

The results of the experiments for the fixed regularization parameter value are presented in Table 1. Clearly, the methods are successful in learning conditional ranking models, and the easier the problem is made, the better the performance is. For all the methods and data sets, except for the conditional ranking method with $w1$ data, the pairwise ranking error is smaller when using the reciprocal kernel. Thus enforcing prior knowledge about the properties of the true underlying relation appears to be beneficial. On this data set, standard regression proves to be competitive with the pairwise ranking approaches. Similar results, where regression approaches can yield an equally good, or even a lower ranking error than rank loss optimizers, are known in the recent literature, see e.g. Pahikkala et al. (2009), Kotlowski et al. (2011). Somewhat surprisingly, RankSVM loses to the other methods in all the experiments other than the $w10$ experiment with reciprocal kernel, with difference being especially large in the $w100$ experiment.

In order to have a more comprehensive view of the differences between the RLS, RankRLS and RankSVM results, we plotted the average test performance for the methods over the 100 repetitions of the experiments, for varying regularization parameter choices. The results are presented in Fig. 2. For $w1$ and $w10$ data sets all the methods share a similar behaviour. The optimal ranking error can be reached for a range of smaller parameter values, until a point is reached where the error starts increasing. However, on $w100$ data sets RankSVM has quite a different type of behaviour.[2] On this data, RankSVM can reach as good as, or even better performance than RLS or RankRLS, but only for a very narrow range of parameter values. Thus, for this data prior knowledge about the suitable parameter value would be needed in order to make RankSVM work, whereas the other approaches are more robust as long as the parameter is set to a fairly small value.

In conclusion, we have shown in this section 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, the methods benefit from the use of the reciprocal Kronecker kernel instead of the ordinary Kronecker kernel. Finally, for this dataset it appears that a regression-based approach performs as good as the pairwise ranking methods.

6.2 Document retrieval: the 20-newsgroups dataset

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

[2] In order to ascertain that the difference was not simply caused by problems in the implementation or the underlying optimization library, we checked our results against those of the SVM^{rank} implementation available at http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html.

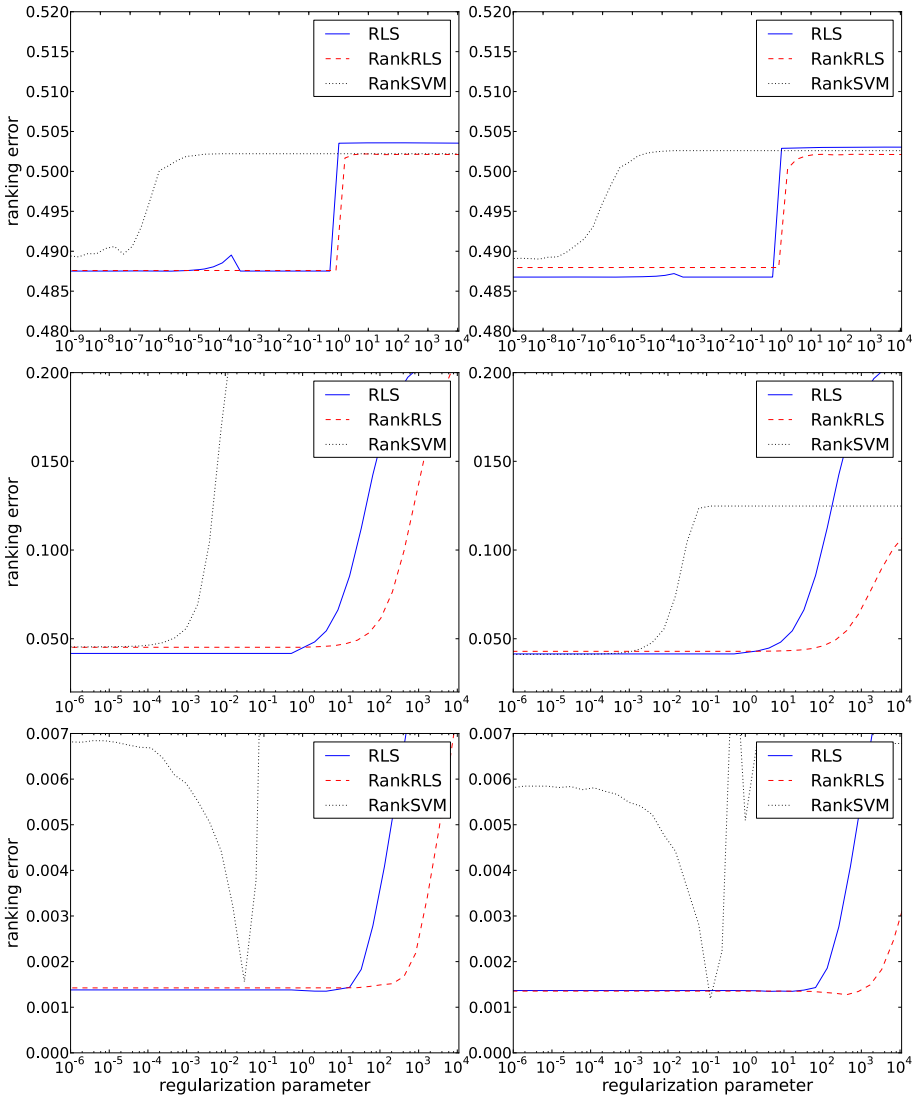

Fig. 2 Rock-paper-scissors data. Ranking test error as a function of regularization parameter for the tested methods. Vertically: w1 (*up*), w10 (*middle*), w100 (*bottom*). Horizontally: standard Kronecker kernel (*left*), reciprocal kernel (*right*)

the publicly available 20-newsgroups data set[3] 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,

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

Table 2 Overview of the measured rank loss for the 20-Newsgroups and the bacterial species ranking tasks in the large-scale experiments, where c.rank and c.reg are trained using the closed-form solutions

	Newsgr. 1	Newsgr. 2	Bacterial 1	Bacterial 2
c. rank	0.2562	0.2895	0.1082	0.07631
c. reg	0.3685	0.3967	0.1084	0.07762

documents from the same newsgroup should be ranked the 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 Agarwal (2006), 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 complete, 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. 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 2. At the end of this section we also present experimental results for the iterative conjugate gradient method, 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, while set 2 contains 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, while the graph formed by set 2 contains approximately 5000 nodes. 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 re-assembled.

The results for the closed-form solution experiments are presented in Table 2. 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 greatly outperforms the one regressing the values for the relations.

Finally, we investigate 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 2. The change in setup results in an increased computational cost, since each iteration of the BGSM method costs as much as using Proposition 2 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 4 newsgroups. The task is now easier than before, since the training and test sets have the same distribution. All the methods are trained for

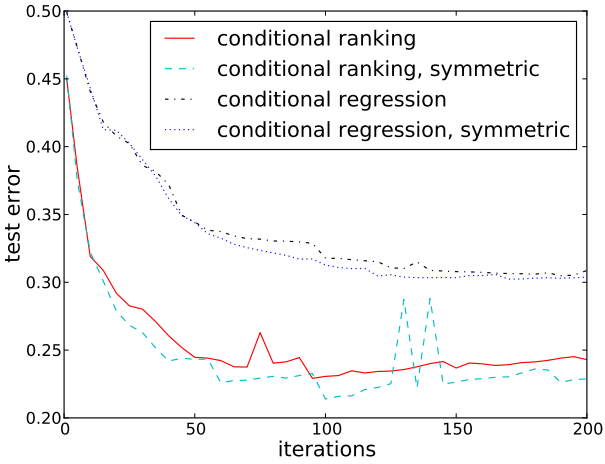

Fig. 3 Experimental results for the 20-Newsgroups data in the small-scale experiment, in which all four models are learned using conjugate gradient descent algorithms

200 iterations, and the test error is plotted. We do not apply any regularization, but rather rely on the regularizing effect of early stopping, as discussed in Sect. 4.

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 error rate curves are not monotonically decreasing, but rather on some iterations the error may momentarily rise sharply. This is due to the behaviour of the conjugate gradient optimization scheme, which sometimes takes steps that lead further away from the optimal solution. 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. 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. We also 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.

6.3 Microbiology: ranking bacterial species

We also illustrate the potential of conditional ranking for multi-class classification problems with a huge number of classes. For such problems it often happens that many classes are not represented in the training dataset, simply because no observations of these classes are known at the moment that the training dataset is constructed and the predictive model is learned. It speaks for itself that existing multi-class classification methods cannot make any correct predictions for observations of these classes, which might occur in the test set.

However, by reformulating the problem as a conditional ranking task, one is still capable of extracting some useful information for these classes during the test phase. The conditional ranking algorithms that we introduced in this article have the ability to condition a ranking on a target object that is unknown during the training phase. In a multi-class classification

setting, we can condition the ranking on objects of classes that are not present in the training data. To this end, we consider bacterial species identification in microbiology.

In this application domain, one normally defines a multi-class classification problem with a huge number of classes as identifying bacterial species, given their fatty acid methyl ester (FAME) profile as input for the model (Slabbinck et al. 2010; MacLeod et al. 2010). Here we reformulate this task as a conditional ranking task. For a given target FAME profile of a bacteria that is not necessarily present in the training dataset, the algorithm should rank all remaining FAME profiles of the same species higher than FAME profiles of other species. For the most challenging scenario, none of these FAME profiles appears in the training dataset.

As a result, the underlying relational graph consists of two types of edges, those connecting FAME profiles of identical species and those connecting FAME profiles of different species. When conditioned on a single node, this setting realizes a bipartite ranking problem, based on an underlying symmetric relation.

The data we used is described in more detail in Slabbinck et al. (2010). Its original version consists of 955 FAME profiles, divided into 73 different classes that represent different bacterial species. A training set and two separate test sets were formed as follows. The data points belonging to the largest two classes were randomly divided between the training set, and test set 1. Of the remaining smaller classes, 26 were included entirely in the training set, and 27 were combined together to form test set 2. The difference between the test sets was thus that FAME profiles from classes contained in test set 1 appear also in the training set, while this is not the case for test set 2. The resulting set sizes were as follows. Training set: 473 nodes, test set 1: 308 nodes and test set 2: 174 nodes. Since the graphs are fully connected, the number of edges grows quadratically with respect to the number of nodes. The regularization parameter is chosen on a separate holdout set.

Due to the large number of edges, we train the rankers using the closed-form solution. We also ran an experiment where we tested the effects of using the symmetric Kronecker kernel, together with the iterative training algorithm. In this experiment, using the symmetric Kronecker kernel leads to a very similar performance as not using it, therefore we do not present these results separately.

Table 2 summarizes the resulting rank loss for the two different test sets, obtained after training the conditional regression and ranking methods using the closed-form solutions. Both methods are capable of training accurate ranking models that can distinguish bacteria of the same and different species groups, as the conditioning data points. Furthermore, comparing the results for test sets 1 and 2, we note that for this problem it is not necessary to have bacteria from the same species present in both the test and training sets, for the models to generalize. In fact, the test error on test set 2 is lower than the error on test set 1. The ranking-based loss function leads to a slightly better test performance than regression.

6.4 Bioinformatics: functional ranking of enzymes

As a last application we consider the problem of ranking a database of enzymes according to their catalytic similarity to a query protein. This catalytic similarity, which serves as the relation of interest, represents the relationship between enzymes w.r.t. their biological function. For newly discovered enzymes, this catalytic similarity is usually not known, so one can think of trying to predict it using machine learning algorithms and kernels that describe the structure-based or sequence-based similarity between enzymes. The Enzyme Commission (EC) functional classification is commonly used to subdivide enzymes into functional classes. EC numbers adopt a four-label hierarchical structure, representing different levels of catalytic detail.

We base the conditional rankings on the EC numbers of the enzymes, information which we assume to be available for the training data, but not at prediction time. This ground truth ranking can be deduced from the catalytic similarity (i.e. ground truth similarity) between the query and all database enzymes. To this end, we count the number of successive correspondences from left to right, starting from the first digit in the EC label of the query and the database enzymes, and stopping as soon as the first mismatch occurs. For example, an enzyme query with number EC 2.4.2.23 has a similarity of two with a database enzyme labeled EC 2.4.99.12, since both enzymes belong to the family of glycosyl transferases. The same query manifests a similarity of one with an enzyme labeled EC 2.8.2.23. Both enzymes are transferases in this case, but they show no further similarity in the chemistry of the reactions to be catalyzed.

Our models were built and tested using a dataset of 1730 enzymes with known protein structures. All the enzyme structures had a resolution of at least 2.5 Å, they had a binding site volume between 350 and 3500 Å^3, and they were fully EC annotated. For evaluation purposes our database contained at least 20 observations for every EC number, leading to a total of 21 different EC numbers comprising members of all 6 top level codes. A heat map of the catalytic similarity of the enzymes is given in Fig. 4. This catalytic similarity will be our relation of interest, constituting the output of the algorithm. As input we consider five state-of-the-art kernel matrices for enzymes, denoted cb (CavBase similarity), mcs (maximum common subgraph), lpcs (labeled point cloud superposition), wp (fingerprints) and wfp (weighted fingerprints). More details about the generation of these kernel matrices can be found in Stock et al. (2012).

The dataset was randomized and split in four equal parts. Each part was withheld as a test set while the other three parts of the dataset were used for training and model selection. This process was repeated for each part so that every instance was used for training and testing (thus, four-fold outer cross-validation). In addition, a 10-fold inner cross validation loop was implemented for estimating the optimal regularization parameter λ, as recommended by Varma and Simon (2006). The value of the hyperparameter was selected from a grid containing all the powers of 10 from 10^{-4} to 10^{5}. The final model was trained using the whole training set and the median of the best hyperparameter values over the ten folds.

We benchmark our algorithms against an unsupervised procedure that is commonly used in bioinformatics for retrieval of enzymes. Given a specific enzyme query and one of the above similarity measures, a ranking is constructed by computing the similarity between the query and all other enzymes in the database. Enzymes having a high similarity to the query appear on top of the ranking, those exhibiting a low similarity end up at the bottom. More formally, let us represent the similarity between two enzymes by $K : \mathcal{V}^2 \to \mathbb{R}$, where $\mathcal{V}$ represents the set of all potential enzymes. Given the similarities $K(v, v')$ and $K(v, v'')$ we compose the ranking of v' and v'' conditioned on the query v as:

$$v' \succeq_v v'' \quad \Leftrightarrow \quad K\big(v, v'\big) \geq K\big(v, v''\big). \tag{40}$$

This approach follows in principle the same methodology as a nearest neighbour classifier, but rather a ranking than a class label should be seen as the output of the algorithm.

Table 3 gives a global summary of the results obtained for the different ranking approaches. All models score relatively well. One can observe that supervised ranking models outperform unsupervised ranking for all five kernels. Three important reasons can be put forward for explaining the improvement in performance. First of all, the traditional benefit of supervised learning plays an important role. One can expect that supervised ranking methods outperform unsupervised ranking methods, because they take ground-truth rankings into account during the training phase to steer towards retrieval of enzymes with a

Table 3 A summary of the results obtained for the enzyme ranking problem

	cb	fp	wfp	mcs	lpcs
unsupervised	0.0938	0.1185	0.1533	0.1077	0.1123
c. reg	0.0052	0.0050	0.0019	0.0054	0.0073
c. rank	0.0049	0.0050	0.0019	0.0056	0.0048

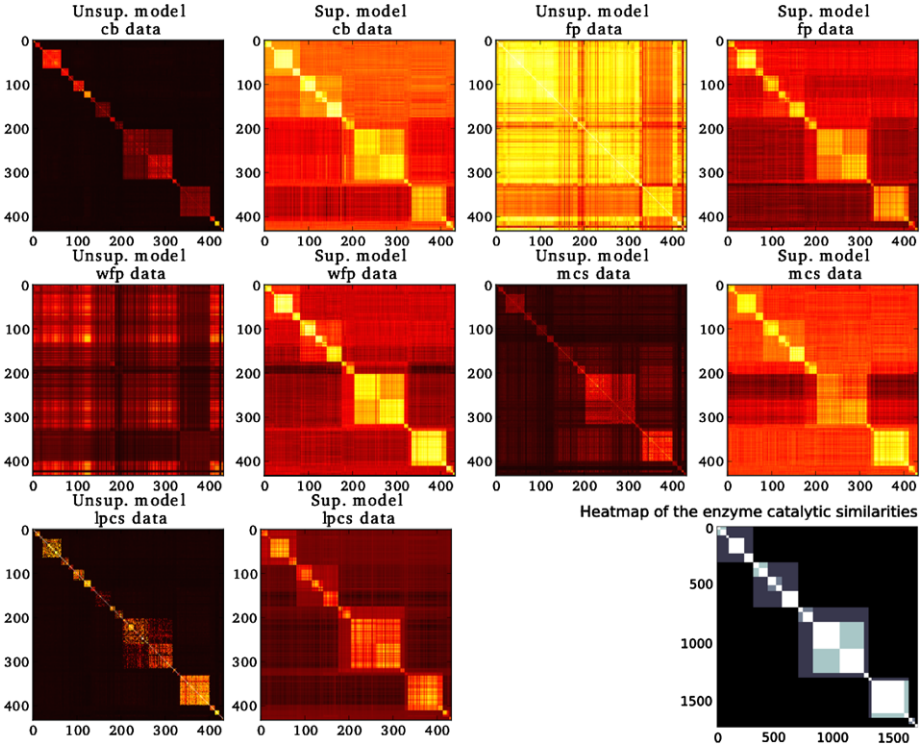

Fig. 4 Heatmaps of the values used for ranking the database during one fold in the testing phase. Each *row* of the heat map corresponds to one query. The corresponding ground truth is given in the *lower right picture*. The supervised model is trained by optimizing the pairwise ranking loss

similar EC number. Conversely, unsupervised methods solely rely on the characterization of a meaningful similarity measure between enzymes, while ignoring EC numbers.

Second, we also advocate that supervised ranking methods have the ability to preserve the hierarchical structure of EC numbers in their predicted rankings. Figure 4 supports this claim. It summarizes the values used for ranking one fold of the test set obtained by the different models as well as the corresponding ground truth. So, for supervised ranking it visualizes the values $h(v, v')$, for unsupervised ranking it visualizes $K(v, v')$. Each row of the heatmap corresponds to one query. For the supervised models one notices a much better correspondence with the ground truth. Furthermore, the different levels of catalytic similarity can be better distinguished.

A third reason for improvement by the supervised ranking method can be found in the exploitation of dependencies between different similarity values. Roughly speaking, if one is interested in the similarity between enzymes v and u, one can try to compute the similarity in

a direct way, or derive it from the similarity with a third enzyme z. In the context of inferring protein-protein interaction and signal transduction networks, both methods are known as the direct and indirect approach, respectively (Vert et al. 2007; Geurts et al. 2007). We argue that unsupervised ranking boils down to a direct approach, while supervised ranking should be interpreted as indirect. Especially when the kernel matrix contains noisy values, one can expect that the indirect approach allows for detecting the *back bone* entries and correcting the noisy ones.

The results for the two supervised conditional ranking approaches are in many cases similar, with both models having same predictive performance on two of the kernels (fp and wfp). For one of the kernels (lpcs) ranking loss gives much better performance than the regression one, for another kernel (cb) ranking loss has a slight advantage, and in the remaining experiment (mcs) the regression approach performs slightly better. An appropriate choice of the node-level kernel proves to be much more important than the choice of the loss function, as the supervised models trained using the wfp kernel clearly outperform all other approaches.

6.5 Runtime performance

In the runtime experiment we compare the computational efficiency of the conditional ranking approaches considered in Sect. 4. We consider conjugate gradient training with early stopping and the closed-form solution, as well as two off-the-shelf ranking algorithms trained directly on the edges, namely RankRLS and RankSVM. For kernel RankSVM, we use the SVM^{light} package, implementing the Kronecker product kernel in the *kernel.h* file. The linear RankSVM experiments are run using the SVM^{rank} package. The other methods are implemented in the RLScore software. All experiments are run on a single core with an Intel Core i7-3770 processor, with 16 GB memory available. The experiments are performed with regularization parameter $\lambda = 1$, and a limit of 200 iterations for the conjugate gradient method.

In the first two experiments, we consider the efficiency of the different kernelized solvers on samples of the Reuters data. First, we measure the scalability of the methods on a fully connected graph with a varying number of nodes. Second, we fix the number of nodes to 1000 and vary the number of edges.

The results are presented in Fig. 5 (top and middle). First, let us consider the scaling behaviour on a fully connected graph. The kernel RankRLS solver has cubic time complexity, training it on all the edges in the fully connected training graph thus results in $O(p^6)$ time complexity. It can be observed that in practice the approach does not scale beyond tens of nodes (thousands of edges), meaning that it cannot be applied beyond small toy problems. The RankSVM solver has even worse scaling. In contrast, the iterative training algorithm (Early stopping CG) and the closed-form solution allow efficient scaling to graphs with thousands of nodes, and hence millions of edges. While the iterative training method and the closed-form share the same $O(p^3)$ asymptotic behaviour, the closed-form solution allows an order of magnitude faster training, making it the method of choice whenever applicable.

Next, we consider training the algorithms on sparse graphs. When the graph is very sparse, meaning that there are only on average around ten or less edges for each node, the RankSVM solver is the most efficient method to use. Once the graph becomes denser, using the Kronecker product shortcuts becomes necessary. Beyond 32000 edges only the Early stopping CG method, whose iteration cost does not depend on the number of edges, proves feasible to use.

Finally, we performed an experiment where we compare the proposed algorithms to existing linear solvers, using low-dimensional data and the linear kernel. We sampled 100 data

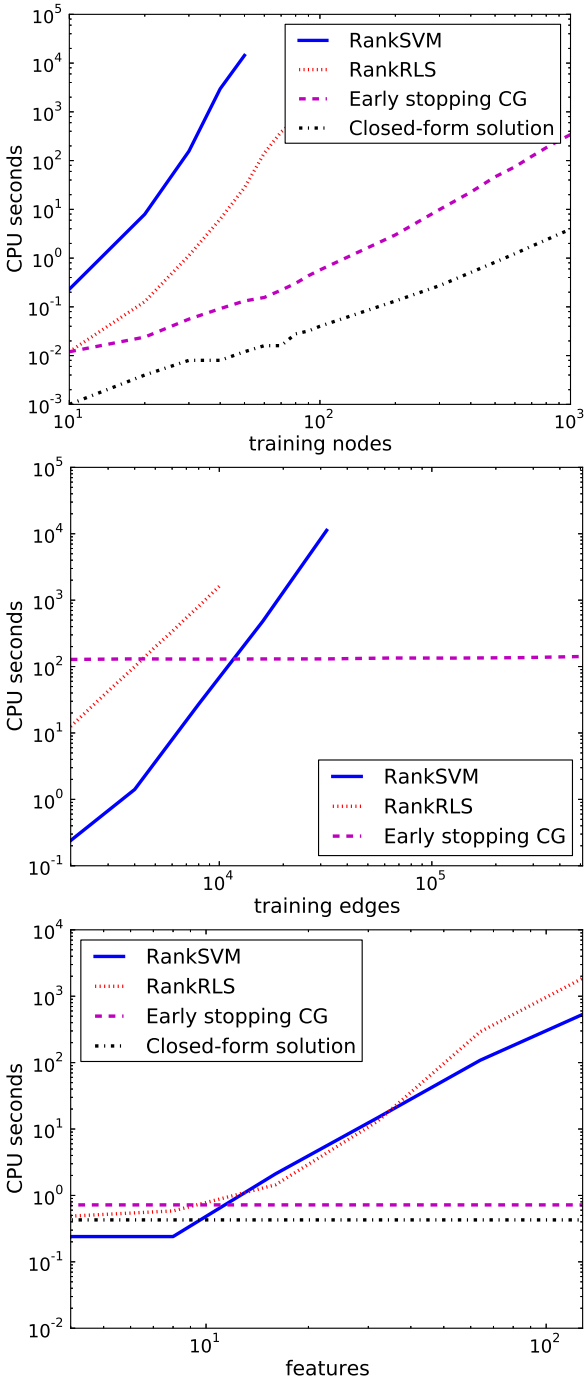

Fig. 5 Scalability experiments for training different algorithms on a sample of the 20-Newsgroups data. We consider both a fully connected training graph with varying amount of nodes (*top*) as well as a graph with 1000 nodes and varying degrees of sparsity (*middle*). Finally, we consider linear solvers with a fully connected graph of 100 nodes and varying number of features (*bottom*)

points from the UCI repository USPS data set, and generated a fully connected label graph by assigning label 1 to data point pairs belonging to the same, and 0 to pairs belonging to different classes. We vary the dimensionality of the data by sampling the features. The linear solvers are trained on explicitly computed Kronecker product features. As shown in Fig. 5 (bottom), the RankSVM and RankRLS solvers are feasible to use and even competitive if the number of features is very low (e.g. 10 or less), as in this case the number of generated product features is also low enough to allow for efficient training. As the number of features grows, using basic RankSVM or RankRLS, however, becomes first inefficient and then infeasible, we did not perform experiments for more than 128 features since soon after this point the data matrix no longer fits into memory. We also performed experiments with 1000 nodes, in this case linear RankRLS and RankSVM did not scale beyond 20 features.

The results further demonstrate our claims about the scalability of the proposed algorithms to large dense graphs. Even with a non-optimized high-level programming language implementation (Python), one can handle training a kernel solver on million edges in a matter of minutes. On very sparse graphs, or when applying linear models with low-dimensional data using existing solvers may also prove feasible.

7 Conclusion

We presented a general framework for conditional ranking from various types of relational data, where rankings can be conditioned on unseen objects. We proposed efficient least-squares algorithms for optimizing regression and ranking-based loss functions, and presented generalization bounds motivating the advantages of using the ranking based loss. Symmetric or reciprocal relations can be treated as two important special cases of the framework, we prove that such prior knowledge can be enforced without having to sacrifice computational efficiency. Experimental results on both synthetic and real-world datasets confirm that the conditional ranking problem can be solved efficiently and accurately, and that optimizing a ranking-based loss can be beneficial, instead of aiming to predict the underlying relations directly. Moreover, we also showed empirically that incorporating domain knowledge about the underlying relations can boost the predictive performance.

Briefly summarized, we have discussed the following three approaches for solving conditional ranking problems:

- 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;
- 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;
- the closed-form solution presented in Proposition 2 is recommended in case the training graph is fully connected, since its computational complexity is equivalent to that of a single iteration of the conjugate gradient method.

Both the computational complexity analysis and the scalability experiments demonstrate, that the introduced algorithms allow solving orders of magnitudes larger conditional ranking problems than was previously possible with existing ranking algorithms.

Acknowledgements We would like to thank the anonymous reviewers for their insightful comments. T.P. and A.A. are both supported for this work by the Academy of Finland (grant 134020 and 128061, respectively). W.W. is supported by the Research Foundation of Flanders. A preliminary version of this work was presented at the European Conference on Machine Learning in 2010 (Pahikkala et al. 2010a).

References

Abadir, M., & Magnus, J. (2005). *Matrix algebra*. Cambridge: Cambridge University Press.

Agarwal, S. (2006). Ranking on graph data. In W. W. Cohen & A. Moore (Eds.), *ACM international conference proceeding series: Vol. 148. Proceedings of the 23rd international conference on machine learning* (pp. 25–32). New York: ACM.

Airola, A., Pahikkala, T., & Salakoski, T. (2011a). On learning and cross-validation with decomposed Nyström approximation of kernel matrix. *Neural Processing Letters*, *33*(1), 17–30.

Airola, A., Pahikkala, T., & Salakoski, T. (2011b). Training linear ranking SVMs in linearithmic time using red-black trees. *Pattern Recognition Letters*, *32*(9), 1328–1336.

Basilico, J., & Hofmann, T. (2004). Unifying collaborative and content-based filtering. In C. E. Brodley (Ed.), *ACM international conference proceeding series: Vol. 69. Proceedings of the twenty-first international conference on machine learning (ICML'04)*, New York: ACM.

Ben-Hur, A., & Noble, W. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, *21*(1), 38–46.

Brunner, C., Fischer, A., Luig, K., & Thies, T. (2012). Pairwise support vector machines and their application to large scale problems. *Journal of Machine Learning Research*, *13*, 2279–2292.

Caetano, T., McAuley, J., Cheng, L., Le, Q., & Smola, A. (2009). Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(6), 1048–1058.

Cao, Y., Xu, J., Liu, T. Y., Li, H., Huang, Y., & Hon, H. W. (2006). Adapting ranking SVM to document retrieval. In E. N. Efthimiadis, S. T. Dumais, D. Hawking, & K. Järvelin (Eds.), *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2006)* (pp. 186–193). New York: ACM.

Chapelle, O., & Keerthi, S. S. (2010). Efficient algorithms for ranking with SVMs. *Information Retrieval*, *13*(3), 201–215.

De Baets, B., De Meyer, H., De Schuymer, B., & Jenei, S. (2006). Cyclic evaluation of transitivity of reciprocal relations. *Social Choice and Welfare*, *26*, 217–238.

De Vito, E., Rosasco, L., Caponnetto, A., De Giovannini, U., & Odone, F. (2005). Learning from examples as an inverse problem. *Journal of Machine Learning Research*, *6*, 883–904.

Engl, H., Hanke, M., & Neubauer, A. (1996). *Mathematics and its applications: Vol. 375. Regularization of inverse problems*. Dordrecht: Kluwer Academic.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*(1), 1–50.

Fisher, L. (2008). *Rock, paper, scissors: game theory in everyday life*. New York: Basic Books.

Freund, Y., Yier, R., Schapire, R., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Fürnkranz, J., Hüllermeier, E., & Vanderlooy, S. (2009). Binary decomposition methods for multipartite ranking. In W. L. Buntine, M. Grobelnik, D. Mladenic, & J. Shawe-Taylor (Eds.), *Lecture notes in computer science: Vol. 5781. Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML/PKDD'09)* (pp. 359–374). Berlin: Springer.

Geerts, F., Mannila, H., & Terzi, E. (2004). Relational link-based ranking. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, & K. B. Schiefer (Eds.), *Proceedings of the thirtieth international conference on very large data bases* (pp. 552–563). San Mateo: Morgan Kaufmann.

Geurts, P., Touleimat, N., Dutreix, M., & d'Alché-Buc, F. (2007). Inferring biological networks with output kernel trees. *BMC Bioinformatics*, *8*(2), S4.

Grangier, D., & Bengio, S. (2008). A discriminative kernel-based approach to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(8), 1371–1384.

Henderson, H. V., & Searle, S. R. (1981). On deriving the inverse of a sum of matrices. *SIAM Review*, *23*(1), 53–60.

Horn, R. A., & Johnson, C. R. (1985). *Matrix analysis*. Cambridge: Cambridge University Press.

Horn, R. A., & Johnson, C. R. (1991). *Topics in matrix analysis*. New York: Cambridge University Press.

Hüllermeier, E., Fürnkranz, J., Cheng, W., & Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, *172*(16–17), 1897–1916.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In D. Hand, D. Keim, & R. Ng (Eds.), *Proceedings of the 8th ACM SIGKDD conference on knowledge discovery and data mining (KDD'02)* (pp. 133–142). New York: ACM.

Joachims, T. (2006). Training linear SVMs in linear time. In T. Eliassi-Rad, L. H. Ungar, M. Craven, & D. Gunopulos (Eds.), *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06)* (pp. 217–226). New York: ACM.

Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., & Tsuda, K. (2009a). Link propagation: a fast semi-supervised learning algorithm for link prediction. In *Proceedings of the SIAM international conference on data mining (SDM 2009)* (pp. 1099–1110). Philadelphia: SIAM.

Kashima, H., Oyama, S., Yamanishi, Y., & Tsuda, K. (2009b). On pairwise kernels: an efficient alternative and generalization analysis. In T. Theeramunkong, B. Kijsirikul, N. Cercone, & T. B. Ho (Eds.), *Lecture notes in computer science: Vol. 5476. Proceedings of the 13th Pacific-Asia conference on advances in knowledge discovery and data mining* (pp. 1030–1037). Berlin: Springer.

Kersting, K., & Xu, Z. (2009). Learning preferences with hidden common cause relations. In W. L. Buntine, M. Grobelnik, D. Mladenic, & J. Shawe-Taylor (Eds.), *Lecture notes in computer science: Vol. 5781. Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML/PKDD'09)* (pp. 676–691). Berlin: Springer.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*(1), 82–95.

Kotlowski, W., Dembczynski, K., & Hüllermeier, E. (2011). Bipartite ranking through minimization of univariate loss. In L. Getoor & T. Scheffer (Eds.), *Proceedings of the 28th international conference on machine learning (ICML 2011)* (pp. 1113–1120). New York: Omnipress.

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

Luce, R., & Suppes, P. (1965). Preference, utility and subjective probability. In *Handbook of mathematical psychology* (pp. 249–410). New York: Wiley.

MacLeod, N., Benfield, M., & Culverhouse, P. (2010). Time to automate identification. *Nature*, *467*, 154–155.

Martin, C. D., & Van Loan, C. F. (2006). Shifted Kronecker product systems. *SIAM Journal on Matrix Analysis and Applications*, *29*(1), 184–198.

Menon, A., & Elkan, C. (2010). Predicting labels for dyadic data. *Data Mining and Knowledge Discovery*, *21*(2), 327–343.

Ng, M. K. P., Li, X., & Ye, Y. (2011). Multirank: co-ranking for objects and relations in multi-relational data. In C. Apté, J. Ghosh, & P. Smyth (Eds.), *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'11)* (pp. 1217–1225). New York: ACM.

Oyama, S., & Manning, C. (2004). Using feature conjunctions across examples for learning pairwise classifiers. In J. F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *Lecture notes in computer science: Vol. 3201. Proceedings of the 15th European conference on machine learning* (pp. 322–333). Berlin: Springer.

Pahikkala, T., Tsivtsivadze, E., Airola, A., Järvinen, J., & Boberg, J. (2009). An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, *75*(1), 129–165.

Pahikkala, T., Waegeman, W., Airola, A., Salakoski, T., & De Baets, B. (2010a). Conditional ranking on relational data. In J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Eds.), *Lecture notes in computer science: Vol. 6322. Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML/PKDD'10), part II* (pp. 499–514). Berlin: Springer.

Pahikkala, T., Waegeman, W., Tsivtsivadze, E., Salakoski, T., & De Baets, B. (2010b). Learning intransitive reciprocal relations with kernel methods. *European Journal of Operational Research*, *206*(3), 676–685.

Park, Y., & Marcotte, E. M. (2012). Flaws in evaluation schemes for pair-input computational predictions. *Nature Methods*, *9*(12), 1134–1136.

Qin, T., Liu, T. Y., Zhang, X. D., Wang, D. S., Xiong, W. Y., & Li, H. (2008). Learning to rank relational objects and its application to web search. In J. Huai, R. Chen, H. W. Hon, Y. Liu, W. Y. Ma, A. Tomkins, & X. Zhang (Eds.), *Proceedings of the 17th international conference on world wide web* (pp. 407–416). New York: ACM.

Raymond, R., & Kashima, H. (2010). Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Eds.), *Lecture notes in computer science: Vol. 6323. Proceedings of the 2010 European conference on machine learning and knowledge discovery in databases: part III* (pp. 131–147). Berlin: Springer.

Rudin, W. (1991). *International series in pure and applied mathematics: Functional analysis* (2nd ed.). New York: McGraw-Hill

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In J. W. Shavlik (Ed.), *Proceedings of the fifteenth international conference on machine learning* (pp. 515–521). San Mateo: Morgan Kaufmann

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Slabbinck, B., Waegeman, W., Dawyndt, P., De Vos, P., & De Baets, B. (2010). From learning taxonomies to phylogenetic learning: integration of 16S rRNA gene data into FAME-based bacterial classification. *BMC Bioinformatics*, *11*(1), 69.

Srebro, N., Rennie, J. D. M., & Jaakkola, T. S. (2005). Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 17, pp. 1433–1440). Cambridge: MIT Press.

Steinwart, I. (2002). On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, *2*, 67–93.

Steinwart, I., & Christmann, A. (2008). *Information science and statistics: Support vector machines*. New York: Springer.

Stock, M., Pahikkala, T., Airola, A., Salakoski, T., De Baets, B., & Waegeman, W. (2012). Learning monadic and dyadic relations: three case studies in systems biology. In *Proceedings of the ECML/PKDD 2012 workshop on learning and discovery in symbolic systems biology* (pp. 74–84).

Suykens, J., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines*. Singapore: World Scientific

Tsochantaridis, Y., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and independent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

van der Vorst, H. A. (1992). 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), 631–644.

van Laarhoven, T., Nabuurs, S. B., & Marchiori, E. (2011). Gaussian interaction profile kernels for predicting drug-target interaction. *Bioinformatics*, *27*(21), 3036–3043.

Van Loan, C. F. (2000). The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, *123*(1–2), 85–100.

Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. *Bioinformatics*, *7*(1), 91.

Vert, J., Qiu, J., & Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, *8*, S8.

Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., Stock, M., & De Baets, B. (2012). A kernel-based framework for learning graded relations from data. *IEEE Transactions on Fuzzy Systems*, *20*(6), 1090–1101.

Weston, J., Eliseeff, A., Zhou, D., Leslie, C., & Noble, W. S. (2004). 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), 6559–6563.

Weston, J., Schölkopf, B., Bousquet, O., Mann, T., & Noble, W. (2007). Joint kernel maps. In G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, & S. Vishwanathan (Eds.), *Neural information processing series: Predicting structured data* (pp. 67–83). Cambridge: MIT Press.

Xia, F., Liu, T. Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In W. W. Cohen, A. McCallum, & S. T. Roweis (Eds.), *ACM international conference proceeding series: Vol. 307. Proceedings of the 25th international conference on machine learning* (pp. 1192–1199). New York: ACM.

Xu, Z., Kersting, K., & Joachims, T. (2010). Fast active exploration for link-based preference learning using Gaussian processes. In J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Eds.), *Lecture notes in computer science: Vol. 6323. Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML/PKDD'10), part III* (pp. 499–514). Berlin: Springer.

Yamanishi, Y., Vert, J. P., & Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, *20*(1), i363–i370.

Yang, Y., Bansal, N., Dakka, W., Ipeirotis, P., Koudas, N., & Papadias, D. (2009). Query by document. In R. A. Baeza-Yates, P. Boldi, B. A. Ribeiro-Neto, & B. B. Cambazoglu (Eds.), *Proceedings of the 2nd international conference on web search and data mining (WSDM 2009)* (pp. 34–43). New York: ACM.

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, & N. Kando (Eds.), *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2007)* (pp. 271–278). New York: ACM.