A two-step learning approach for solving full and almost full cold start problems in dyadic prediction

Tapio Pahikkala[1], Michiel Stock[2], Antti Airola[1], Tero Aittokallio[3], Bernard De Baets[2] and Willem Waegeman[2]

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

[2] Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Coupure links 653, B-9000 Ghent, Belgium
firstname.surname@UGent.be

[3] Institute for Molecular Medicine Finland (FIMM), P.O. Box 20 (Tukholmankatu 8), FI-00014 University of Helsinki, Helsinki, Finland.
firstname.surname@fimm.fi

Abstract. Dyadic prediction methods operate on pairs of objects (dyads), aiming to infer labels for out-of-sample dyads. We consider the full and almost full cold start problem in dyadic prediction, a setting that occurs when both objects in an out-of-sample dyad have not been observed during training, or if one of them has been observed, but very few times. A popular approach for addressing this problem is to train a model that makes predictions based on a pairwise feature representation of the dyads, or, in case of kernel methods, based on a tensor product pairwise kernel. As an alternative to such a kernel approach, we introduce a novel two-step learning algorithm that borrows ideas from the fields of pairwise learning and spectral filtering. We show theoretically that the two-step method is very closely related to the tensor product kernel approach, and experimentally that it yields a slightly better predictive performance. Moreover, unlike existing tensor product kernel methods, the two-step method allows closed-form solutions for training and parameter selection via cross-validation estimates both in the full and almost full cold start settings, making the approach much more efficient and straightforward to implement.

Keywords: dyadic prediction, pairwise learning, transfer learning, kernel ridge regression, kernel methods

1 A subdivision of dyadic prediction methods

Many real-world machine learning problems can be naturally represented as pairwise learning or dyadic prediction problems, for which feature representations of two different types of objects (aka a dyad) are jointly used to predict a relationship between those objects. Amongst others, applications of that kind emerge

in biology (e.g. predicting mRNA-miRNA interactions), medicine (e.g. design of personalized drugs), chemistry (e.g. prediction of binding between two types of molecules), social network analysis (e.g. link prediction) and recommender systems (e.g. personalized product recommendation).

For many dyadic prediction problems it is extremely important to implement appropriate training and evaluation procedures. [29] make in a recent Nature-review on dyadic prediction an important distinction between four main settings. Given $\mathbf{t}$ and $\mathbf{d}$ as the feature representations of the two types of objects, those four settings can be summarized as follows:

- **Setting A**: Both $\mathbf{t}$ and $\mathbf{d}$ are observed during training, as parts of separate dyads, but the label of the dyad $(\mathbf{t}, \mathbf{d})$ must be predicted.
- **Setting B**: Only $\mathbf{t}$ is known during training, while $\mathbf{d}$ is not observed in any dyad, and the label of the dyad $(\mathbf{t}, \mathbf{d})$ must be predicted.
- **Setting C**: Only $\mathbf{d}$ is known during training, while $\mathbf{t}$ is not observed in any dyad, and the label of the dyad $(\mathbf{t}, \mathbf{d})$ must be predicted.
- **Setting D**: Neither $\mathbf{t}$ nor $\mathbf{d}$ occur in any training dyad, but the label of the dyad $(\mathbf{t}, \mathbf{d})$ must be predicted (referred to as the full cold start problem).

Setting A is of all four settings by far the most studied setting in the machine learning literature. Motivated by applications in collaborative filtering and link prediction, matrix factorization and related techniques are often applied to complete partially observed matrices, where missing values represent $(\mathbf{t}, \mathbf{d})$ combinations that are not observed during training - see e.g. [15] for a review.

Settings B and C are very similar, and a variety of machine learning methods can be applied for these settings. From a recommender systems viewpoint, those settings resemble the cold start problem (new user or new item), for which hybrid and content-based filtering techniques are often applied – see e.g. [1, 10, 20, 35, 39] for a not at all exhaustive list. From a bioinformatics viewpoint, Settings B and C are often analyzed using graph-based methods that take the structure of a biological network into account – see e.g. [33] for a recent review. When the features of $\mathbf{t}$ are negligible or unavailable, while those of $\mathbf{d}$ are informative, Setting B can be interpreted as a multi-label classification problem (binary labels), a multivariate regression problems (continuous labels) or a specific multi-task learning problem. Here as well, a large number of applicable methods exists in the literature.

1.1 The problem setting considered in this article

Matrix factorization and hybrid filtering strategies are not applicable to Setting D. We will refer to this setting as the *full cold start* problem, which finds important applications in domains such as bioinformatics and chemistry – see experiments. Compared to the other three settings, Setting D has received less attention in the literature (with some exceptions, see e.g. [20, 23, 25, 28]), and it will be our main focus in this article. Furthermore, we will also investigate the transition phase between Settings C and D, when $\mathbf{t}$ occurs very few times in the

training dataset, while $\mathbf{d}$ of the dyad $(\mathbf{d}, \mathbf{t})$ is only observed in the prediction phase. We refer to this setting as the *almost full cold start* problem.

Full and almost full cold start problems can only be solved by considering feature representations of dyads (aka side information in the recommender systems literature). Similar to several existing papers dealing with Setting D, we consider tensor product feature representations and their kernel duals. Such feature representations have been successfully applied in order to solve problems such as product recommendation [5, 28], prediction of protein-protein interactions [7, 14], drug design [13], prediction of game outcomes [26] and document retrieval [23]. For classification and regression problems a standard recipe exists of plugging pairwise kernels in support vector machines, kernel ridge regression (KRR), or any other kernel method. Efficient optimization approaches based on gradient descent [14, 23, 28] and closed form solutions [23] have been proposed. We compare KRR with a tensor product pairwise kernel (tensor KRR) both theoretically and experimentally to the two-step approach introduced in this paper.

1.2 Formulation as a transfer learning problem

As discussed above, dyadic prediction is closely related to several subfields of machine learning. Further on in this article we decide to adopt a multi-task learning or transfer learning terminology, using $\mathbf{d}$ and $\mathbf{t}$ to denote the feature representations of instances and tasks, respectively. From this viewpoint, Setting C corresponds to a specific instantiation of a traditional transfer learning scenario, in which the aim is to transfer knowledge obtained from already learned *auxiliary tasks* to the *target task* of interest [27]. Stretching the concept of transfer learning even further, in the case of so-called *zero-data learning*, one arrives at Setting D, which is characterized by no available labeled training data for the target task [16]. If the target task is unknown during the training time, the learning method must be able to generalize to it "on the fly" at prediction time. The only available data here is coming from auxiliary training tasks.

We present a simple but elegant two-step approach to tackle these settings. First, a KRR model trained on auxiliary tasks is used to predict labels for the related target task. Next, a second model is constructed, using KRR on the target data, augmented by the predictions of the first phase. We show via spectral filtering that this approach is closely related to learning a pairwise model using a tensor product pairwise kernel. However, the two-step approach is much simpler to implement and it allows more heterogeneous transfer learning settings than the ordinary pairwise kernel ridge regression, as well as a more flexible model selection. Furthermore, it allows for a more efficient generalization to new tasks not known during training time, since the model built on auxiliary tasks does not need to be re-trained in such settings. In the experiments we consider three distinct dyadic prediction problems, concerning drug-target, newsgroup document similarity and protein functional similarity predictions. Our results show that the two-step transfer learning approach can be highly beneficial when there is no labeled data at all, or only a small amount of labeled data available

for the target task, while in settings where there is a significant amount of labeled data available for the target task a single-task model suffices. In related work, [34] have recently proposed a similar two-step approach based on tree-based ensemble methods for biological network inference.

2 Solving full and almost full cold start problems via transfer learning

Adopting a multi-task learning methodology, the training set is assumed to consist of a set $\{\mathbf{x}_h\}_{h=1}^n$ of object-task pairs and a vector $\mathbf{y} \in \mathbb{R}^n$ of their real-valued labels. We assume that each training input can be represented as $\mathbf{x} = (\mathbf{d}, \mathbf{t})$, where $\mathbf{d} \in \mathcal{D}$ and $\mathbf{t} \in \mathcal{T}$ are the objects and tasks, respectively, and $\mathcal{D}$ and $\mathcal{T}$ are the corresponding spaces of objects and tasks. Moreover, let $D = \{\mathbf{d}_i\}_{i=1}^m$ and $T = \{\mathbf{t}_j\}_{j=1}^q$ denote, respectively, the sets of distinct objects and tasks encountered in the training set with $m = |D|$ and $q = |T|$. We say that the training set is *complete* if it contains every object-task pair with object in D and task in T exactly once. For complete training sets, we introduce a further notation for the matrix of labels $\mathbf{Y} \in \mathbb{R}^{m \times q}$, so that its rows are indexed by the objects in D and the columns by the tasks in T. In full and almost full cold start prediction problems, this matrix will not contain any target task information.

2.1 Kernel ridge regression with tensor product kernels

Several authors (see [2, 4] and references therein) have extended KRR to involve task correlations via matrix-valued kernels. However, most of the literature concerns kernels for which the tasks are fixed at training time. An alternative approach, allowing the generalization to new tasks more straightforwardly, is to use the tensor product pairwise kernel [5, 7, 8, 12, 21, 23, 28], in which kernels are defined on object-task pairs

$$\Gamma(\mathbf{x}, \overline{\mathbf{x}}) = \Gamma\left((\mathbf{d}, \mathbf{t}), (\overline{\mathbf{d}}, \overline{\mathbf{t}})\right) = k\left(\mathbf{d}, \overline{\mathbf{d}}\right) g\left(\mathbf{t}, \overline{\mathbf{t}}\right) \tag{1}$$

as a product of the data kernel k and the task kernel g. Let

$$\mathbf{K} \in \mathbb{R}^{m \times m} \text{ and } \mathbf{G} \in \mathbb{R}^{q \times q} \tag{2}$$

be the kernel matrices for the data points and tasks, respectively. Then, the kernel matrix for the object-task pairs is, for a complete training set, the tensor product $\mathbf{\Gamma} = \mathbf{K} \otimes \mathbf{G}$, which is usually infeasible to use directly due to its large size. Tensor KRR seeks for a prediction function of type

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \Gamma(\mathbf{x}, \mathbf{x}_i),$$

where α_i are parameters that minimize the following objective function:

$$J(\boldsymbol{\alpha}) = (\mathbf{\Gamma}\boldsymbol{\alpha} - \mathbf{y})^{\mathrm{T}}(\mathbf{\Gamma}\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{\Gamma} \boldsymbol{\alpha}, \tag{3}$$

Algorithm 1 Two-step kernel ridge regression

1: $\mathbf{C} \leftarrow \operatorname{argmin}_{\mathbf{C} \in \mathbb{R}^{m \times q}} \left\{ \|\mathbf{CG} - \mathbf{Y}\|_F^2 + \lambda_t \operatorname{tr}(\mathbf{CGC}^{\mathrm{T}}) \right\}$
2: $\mathbf{z} \leftarrow \left(\mathbf{z}_{\mathcal{L}}^{\mathrm{T}}, (\mathbf{C}_{\mathcal{U}}\mathbf{g})^{\mathrm{T}} \right)^{\mathrm{T}}$
3: $\mathbf{a} \leftarrow \operatorname{argmin}_{\mathbf{a} \in \mathbb{R}^m} \left\{ (\mathbf{Ka} - \mathbf{z})^{\mathrm{T}} (\mathbf{Ka} - \mathbf{z}) + \lambda_d \mathbf{a}^{\mathrm{T}} \mathbf{Ka} \right\}$
4: **return** $f_{\mathbf{t}}(\cdot) = \sum_{i=1}^{m} a_i k(\mathbf{d}_i, \cdot)$

whose minimizer can be found by solving the following system of linear equations:

$$(\boldsymbol{\Gamma} + \lambda \mathbf{I})\, \boldsymbol{\alpha} = \mathbf{y}. \tag{4}$$

Several authors have pointed out that, while the size of the above system is considerably large, its solution can be found efficiently via tensor algebraic optimization [2, 14, 19, 25, 30, 37]. Namely, the complexity scales roughly of order $O(|D|^3 + |T|^3)$ which is required by computing the singular value decomposition (SVD) of both the object and task kernel matrices, but the complexities can be scaled down even further by using sparse kernel matrix approximations.

However, the above computational short-cut only concerns the case in which the training set is complete. If some of the pairs are missing or if there are several occurrences of certain pairs, one has to resort, for example, to gradient descent based training approaches. While these approaches can also be accelerated via tensor algebraic optimization, they still remain considerably slower than the SVD-based approach. A serious short-coming of the approach is that when generalizing to new tasks, the whole training procedure needs to be re-done with the new training set that contains the union of the auxiliary data and the target data. If the amount of auxiliary data is large, as one would hope in order to expect any positive transfer to happen, this makes generalization to new tasks on-the-fly computationally impractical.

2.2 Two-step kernel ridge regression

Next, we present a two-step procedure for performing transfer learning. In the following, we assume that we are provided a training set in which every auxiliary task has the same labeled training objects. This assumption is fairly realistic in many practical settings, since one can carry out, for example, a preliminary completion step by using the extensive toolkit of missing value imputation or matrix completion algorithms. A newly given target task, in contrast, is assumed to have only a subset of the training objects labeled. That is, the training set consisting of both the auxiliary and the target tasks is incomplete, because of the missing object labels of the target task, ruling out the direct application of the SVD-based training. To cope with this incompleteness, we consider an approach of performing the learning in two steps, of which the first step is used for completing the training set for the target tasks part and the second step for building a model for the target task. A further benefit is that the first phase where a model is trained on auxiliary data needs to be performed only once, and the resulting model may be subsequently re-used when new target tasks appear.

Algorithm 2 Two-step with LOOCV-based automatic model selection

Require: $\mathbf{Y} \in \mathbb{R}^{m \times q}, \mathbf{\Phi} \in \mathbb{R}^{m \times d}, \mathbf{\Psi} \in \mathbb{R}^{q \times r}, \mathbf{g} \in \mathbb{R}^{q}, \mathbf{z}_{\mathcal{L}} \in \mathbb{R}^{|\mathcal{L}|}$ with $d \leq m$ and $r \leq q$.
1: $\mathbf{U}, \sqrt{\mathbf{\Sigma}}, \mathbf{V} \leftarrow \text{SVD}(\mathbf{\Phi})$, with $\mathbf{U} \in \mathbb{R}^{m \times d}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$ ▷ $\mathcal{O}(qr^2)$
2: $\mathbf{P}, \sqrt{\mathbf{S}}, \mathbf{Q} \leftarrow \text{SVD}(\mathbf{\Psi})$, with $\mathbf{P} \in \mathbb{R}^{q \times r}, \mathbf{Q} \in \mathbb{R}^{r \times r}$ ▷ $\mathcal{O}(md^2)$
3: $e \leftarrow \infty$
4: **for** $\overline{\lambda_t} \in \{\text{Grid of parameter values}\}$ **do**
5: **for** $j = 1, \ldots, q$ **do** $\widetilde{G}_{j,j} \leftarrow \mathbf{P}_j(\text{diag}((\mathbf{S} + \overline{\lambda_t}\mathbf{I})^{-1}) \odot \mathbf{P}_j^{\text{T}})$ ▷ $\mathcal{O}(qr)$
6: $\overline{\mathbf{C}} \leftarrow \mathbf{YP}(\mathbf{S} + \overline{\lambda_t}\mathbf{I})^{-1}\mathbf{P}^{\text{T}}$ ▷ $\mathcal{O}(mqr)$
7: **for** $i = 1, \ldots, m$ **and** $j = 1, \ldots, q$ **do** $\overline{R}_{i,j} \leftarrow Y_{i,j} - \left(\widetilde{G}_{j,j}\right)^{-1} \overline{C}_{i,j}$ ▷ $\mathcal{O}(mq)$
8: $\overline{e} \leftarrow \mathcal{E}(\overline{\mathbf{R}}, \mathbf{Y})$ ▷ Error between labels and LOO predictions
9: **if** $\overline{e} < e$ **then** $\lambda_t, e, \mathbf{R}, \mathbf{C} \leftarrow \overline{\lambda_t}, \overline{e}, \overline{\mathbf{R}}, \overline{\mathbf{C}}$
10: $e \leftarrow \infty$
11: **for** $\overline{\lambda_d} \in \{\text{Grid of parameter values}\}$ **do**
12: **for** $i = 1, \ldots, m$ **do** $\widetilde{K}_{i,i} \leftarrow \mathbf{U}_i(\text{diag}((\mathbf{\Sigma} + \overline{\lambda_d}\mathbf{I})^{-1}) \odot \mathbf{U}_i^{\text{T}})$ ▷ $\mathcal{O}(md)$
13: $\overline{\mathbf{A}} \leftarrow \mathbf{U}(\mathbf{\Sigma} + \overline{\lambda_d}\mathbf{I})^{-1}\mathbf{U}^{\text{T}}\mathbf{R}$ ▷ $\mathcal{O}(mqd)$
14: **for** $i = 1, \ldots, m$ **and** $j = 1, \ldots, q$ **do** $\overline{T}_{i,j} \leftarrow Y_{i,j} - \left(\widetilde{K}_{i,i}\right)^{-1} \overline{A}_{i,j}$ ▷ $\mathcal{O}(mq)$
15: $\overline{e} \leftarrow \mathcal{E}(\overline{\mathbf{T}}, \mathbf{Y})$ ▷ Error between labels and LOO predictions
16: **if** $\overline{e} < e$ **then** $\lambda_d, e, \mathbf{T}, \mathbf{A} \leftarrow \overline{\lambda_d}, \overline{e}, \overline{\mathbf{T}}, \overline{\mathbf{A}}$
17: $\mathbf{z} \leftarrow \left(\mathbf{z}_{\mathcal{L}}^{\text{T}}, (\mathbf{C}_{\mathcal{U}}\mathbf{g})^{\text{T}}\right)^{\text{T}}$
18: $\mathbf{a} \leftarrow \mathbf{U}(\mathbf{\Sigma} + \lambda_d\mathbf{I})^{-1}\mathbf{U}^{\text{T}}\mathbf{z}$ ▷ $\mathcal{O}(md)$
19: **return** $f_{\mathbf{t}}(\cdot) = \sum_{i=1}^{m} \mathbf{a}_i k(\mathbf{d}_i, \cdot)$

Let $\mathcal{L} \subseteq D$ and $\mathcal{U} \subseteq D$ be the set of objects that are, respectively, labeled and unlabeled for the target task. Moreover, let $\mathbf{Y}$ now denote the matrix of labels for the auxiliary tasks and $\mathbf{z}_{\mathcal{L}} \in \mathbb{R}^{|\mathcal{L}|}$ the vector of known labels for the target task. Furthermore, in addition to the kernel matrices $\mathbf{K}$ and $\mathbf{G}$ for the training data points and auxiliary tasks defined in (2), let $\mathbf{g} \in \mathbb{R}^q$ denote the vector of task kernel evaluations between the target task and the auxiliary tasks, e.g. $\mathbf{g} = (g(\mathbf{t}, \mathbf{t}_1), \ldots, g(\mathbf{t}, \mathbf{t}_q))^{\text{T}}$, where $\mathbf{t}$ is the target task and $\mathbf{t}_i$ the auxiliary tasks. Finally, let λ_t and λ_d be the regularization parameters for the first and the second learning steps, respectively. The two-step approach is summarized in Algorithm 1. The first training step is carried out by training a multi-output KRR model (line 1), in which a matrix $\mathbf{C}$ of parameters is estimated, and this model is used for predicting the labels indexed by $\mathcal{U}$ for the target task (line 2). The second step trains a single-output KRR for the targer task (line 3), in which a vector $\mathbf{a}$ of parameters is fitted to the data.

2.3 Computational considerations and model selection

Let d and r denote the feature space dimensionalities of the object and task kernels, respectively. These dimensions can be reduced, for example, by the Nyström method in order to lower both the time and space complexities of kernel methods [32], and hence in the following we assume that $d \leq m$ and $r \leq q$. Let $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$

and $\mathbf{\Psi} \in \mathbb{R}^{q \times r}$ be the matrices containing the feature representations of the training objects and tasks in D and T, respectively, so that $\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{K}$ and $\mathbf{\Psi}\mathbf{\Psi}^{\mathrm{T}} = \mathbf{G}$. Let $\mathbf{\Phi} = \mathbf{U}\sqrt{\mathbf{\Sigma}}\mathbf{V}^{\mathrm{T}}$ and $\mathbf{\Psi} = \mathbf{P}\sqrt{\mathbf{S}}\mathbf{Q}^{\mathrm{T}}$ be the SVDs of $\mathbf{\Phi}$ and $\mathbf{\Psi}$, respectively. Since the ranks of the feature matrices are at most the dimensions of the feature spaces, we can save both space and time by only computing the singular vectors that correspond to the nonzero singular values. That is, we compute the matrices $\mathbf{U} \in \mathbb{R}^{m \times d}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$, $\mathbf{P} \in \mathbb{R}^{q \times r}$, and $\mathbf{Q} \in \mathbb{R}^{r \times r}$ via the economy sized SVD, requiring $\mathcal{O}(md^2 + qr^2)$ time. The outcomes of the first and second steps of the two-step KRR (e.g. the first and third lines of Algorithm 1) can be, respectively, written as $\mathbf{C} = \mathbf{Y}\widetilde{\mathbf{G}}$ and $\mathbf{a} = \widetilde{\mathbf{K}}\mathbf{z}$, where $\widetilde{\mathbf{G}} = (\mathbf{G} + \lambda_t \mathbf{I})^{-1} = \mathbf{U}(\mathbf{\Sigma} + \lambda_d \mathbf{I})^{-1}\mathbf{U}^{\mathrm{T}}$ and $\widetilde{\mathbf{K}} = (\mathbf{K} + \lambda_d \mathbf{I})^{-1} = \mathbf{U}(\mathbf{\Sigma} + \lambda_d \mathbf{I})^{-1}\mathbf{U}^{\mathrm{T}}$. Given that the above described SVD components are available, the computational complexity is dominated by the multiplication of the eigenvectors with the label matrix, which requires $\mathcal{O}(mqr)$ time if the matrix multiplications are performed in the optimal order.

We next present an automatic model selection and training approach for the two-step KRR that uses leave-one-out cross-validation (LOOCV) for selecting the values of both λ_t and λ_d. This is illustrated in Algorithm 2. It is well known that, for KRR, the LOOCV performance can be efficiently computed without training the model from scratch during each CV round (we refer to [24, 31] for details). Adapting this to the first step of the two-step KRR, the "leave-column-out" performance for the ith datum on the jth task (e.g. a CV in which each of the columns of $\mathbf{Y}$ are held out at a time to measure the generalization ability to new columns) can be obtained in constant time from $Y_{i,j} - \left(\widetilde{G}_{j,j}\right)^{-1} C_{i,j}$, given that the diagonal entries of $\widetilde{\mathbf{G}}$ and the dual variables $C_{i,j}$ are computed and stored in memory. Using the SVD components, both $\widetilde{G}_{j,j}$ and $C_{i,j}$ can be computed in $\mathcal{O}(r)$ time, which enables the efficient selection of the regularization parameter value with LOOCV. If the value is selected from a set of t candidates and LOOCV is computed for all data points and tasks, the overall complexity is $\mathcal{O}(mqrt)$. This is depicted in lines 4-9 of Algorithm 2, where the overline symbols denote temporary variables used in the search of the optimal candidate value and $\mathcal{E}$ denotes a prediction performance (such as the mean squared error, classification accuracy, or area under curve, for example).

By the definition of the two-step KRR, the second step consists of training a model using the predictions made during the first step as training labels, while the aim is to make good predictions of the true labels. Therefore, we select the regularization parameter value for the second step using LOOCV on a multi-output KRR model trained using the LOO prediction matrix $\mathbf{R}$ obtained from the first step as a label matrix. The second regularization parameter value is thus selected so that the error $\mathcal{E}(\overline{\mathbf{T}}, \mathbf{Y})$ between the LOO predictions made during the second step and the original labels $\mathbf{Y}$ is as small as possible. In contrast to the first step, the aim of the second step is to generalize to new data points, and hence the CV is done in the leave-row-out sense, which can again be efficiently computed as $Y_{i,j} - \left(\widetilde{K}_{i,i}\right)^{-1} A_{i,j}$, where $A_{i,j}$ are the model parameters of the multi-output KRR trained row-wise. This is done in lines 11-16 of Algorithm 2.

The overall computational complexity of the two-step KRR with automatic model selection is $\mathcal{O}(md^2 + qr^2 + mqrt + mqdt)$, where the first two terms denote the time required by SVD computations and the two latter the time spent for CV and grid search for the regularization parameter. The two-step KRR, in addition to enabling non-zero training sets for the target task, provides a very flexible machinery for CV and model selection. This is in contrast to the tensor KRR for which such short-cuts are not available to our knowledge, and there is no efficient closed form solution available for the almost full cold start settings. Note also that, while the above described method separately selects the regularization parameter values for the tasks and the data, the method is easy to modify so that it would select a separate regularization parameter value for each task and for each datum (e.g. altogether $m + q$ parameters), thus allowing considerably more degrees of freedom. However, the consideration of this variation is omitted due to the lack of space.

3 Theoretical considerations

Here, we analyze the two-step learning approach by studying its connections to learning with pairwise tensor product kernels as in (1). These two approaches coincide in an interesting way for the full cold start problems (e.g. the special case in which there is no labeled data available for the target task). This, in turn, allows us to show the consistency of the two-step KRR via its universal approximation and spectral regularization properties.

The connection between the two-step and tensor KRR is characterized by the following result.

Proposition 1. *Let us consider a full cold start setting with a complete training set. Let $f_{\mathbf{t}}(\cdot)$ be a model trained with two-step KRR for the target task $\mathbf{t}$ and $f(\cdot,\cdot)$ be a model trained with an ordinary kernel least-squares regression (OKLS) on the object-task pairs with the following pairwise kernel function on $\mathcal{D} \times \mathcal{T}$:*

$$\Upsilon\left((\mathbf{d},\mathbf{t}),(\overline{\mathbf{d}},\overline{\mathbf{t}})\right) = \left(k\left(\mathbf{d},\overline{\mathbf{d}}\right) + \lambda_d \delta\left(\mathbf{d},\overline{\mathbf{d}}\right)\right)\left(g\left(\mathbf{t},\overline{\mathbf{t}}\right) + \lambda_t \delta\left(\mathbf{t},\overline{\mathbf{t}}\right)\right) \tag{5}$$

where δ is the delta kernel whose value is 1 if the arguments are equal and 0 otherwise. Then, $f_{\mathbf{t}}(\mathbf{d}) = f(\mathbf{t},\mathbf{d})$ for any $\mathbf{d} \in \mathcal{D}$.

Proof. Writing the steps of the algorithm together and denoting $\widetilde{\mathbf{G}} = (\mathbf{G} + \lambda \mathbf{I})^{-1}$ and $\widetilde{\mathbf{K}} = (\mathbf{K} + \lambda \mathbf{I})^{-1}$, we observe that the model parameters $\mathbf{a}$ of the target task can also be obtained from the following closed form:

$$\mathbf{a} = \widetilde{\mathbf{K}}\mathbf{Y}\widetilde{\mathbf{G}}\mathbf{g}\,. \tag{6}$$

The prediction for a datum $\mathbf{d}$ is $f_{\mathbf{t}}(\mathbf{d}) = \mathbf{k}^{\mathrm{T}}\mathbf{a}$, where $\mathbf{k} \in \mathbb{R}^m$ is the vector containing all kernel evaluations between $\mathbf{d}$ and the training data points.

The kernel matrix of Υ for the full cold start setting can be expressed as: $\boldsymbol{\Upsilon} = (\mathbf{G} + \lambda_t \mathbf{I}) \otimes (\mathbf{K} + \lambda_d \mathbf{I})$. The OKLS problem with kernel Υ being

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^{mq}}{\operatorname{argmin}} \left\{ \left(\operatorname{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon}\boldsymbol{\alpha}\right)^{\mathrm{T}} \left(\operatorname{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon}\boldsymbol{\alpha}\right) \right\},$$

its minimizer can be expressed as

$$\begin{aligned}\boldsymbol{\alpha} = \boldsymbol{\Upsilon}^{-1}\mathrm{vec}(\mathbf{Y}) &= \left((\mathbf{G}+\lambda_t\mathbf{I})^{-1} \otimes (\mathbf{K}+\lambda_d\mathbf{I})^{-1}\right)\mathrm{vec}(\mathbf{Y}) \\ &= \mathrm{vec}\left((\mathbf{K}+\lambda_d\mathbf{I})^{-1}\,\mathbf{Y}\,(\mathbf{G}+\lambda_t\mathbf{I})^{-1}\right) = \mathrm{vec}\left(\widetilde{\mathbf{K}}\mathbf{Y}\widetilde{\mathbf{G}}\right)\ . \qquad (7)\end{aligned}$$

The prediction for the datum $\mathbf{d}$ is $(\mathbf{g}\otimes\mathbf{k})^{\mathrm{T}}\mathrm{vec}\left(\widetilde{\mathbf{K}}\mathbf{Y}\widetilde{\mathbf{G}}\right) = \mathbf{k}^{\mathrm{T}}\widetilde{\mathbf{K}}\mathbf{Y}\widetilde{\mathbf{G}}\mathbf{g}$, where we make use of the rule $\mathrm{vec}(\mathbf{MXN}) = (\mathbf{N}^{\mathrm{T}}\otimes\mathbf{N})\mathrm{vec}(\mathbf{X})$ that holds for any conformable matrices $\mathbf{M}$, $\mathbf{X}$, and $\mathbf{N}$. □

The kernel point of view allows us to consider the universal approximation properties of the learned knowledge transfer models. Recall the concept of universal kernel functions:

Definition 1. [36] *A continuous kernel k on a compact metric space $\mathcal{X}$ (i.e. $\mathcal{X}$ is closed and bounded) is called universal if the reproducing kernel Hilbert space (RKHS) induced by k is dense in $C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of all continuous functions $f : \mathcal{X} \to \mathbb{R}$.*

The universality property indicates that the hypothesis space induced by an universal kernel can approximate any continuous function to be learned arbitrarily well, given that the available set of training data is large and representative enough, and the learning algorithm can efficiently find the approximation [36].

Proposition 2. *The kernel Υ on $\mathcal{D}\times\mathcal{T}$ defined in (5) is universal if the kernels k on $\mathcal{D}$ and g on $\mathcal{T}$ are both universal.*

Proof. We provide here a high-level sketch of the proof. The details are omitted due to lack of space but they can be easily verified from the existing literature. The RKHS of sums of reproducing kernels was characterized by [3] as follows: Let $\mathcal{H}(k_1)$ and $\mathcal{H}(k_2)$ be RKHSs over $\mathcal{X}$ with reproducing kernels k_1 and k_2, respectively. If $k = k_1 + k_2$ and $\mathcal{H}(k)$ denotes the corresponding RKHS, then $\mathcal{H}(k) = \{f_1 + f_2 : f_i \in \mathcal{H}(k_i), i = 1, 2\}$. Thus, if the object kernel is universal, the sum of the object and delta kernels is also universal and the same concerns the task kernel. The product of two universal kernels is also universal, as considered in our previous work [38]. □

The full cold start setting with complete auxiliary training set allows us to consider the two-step approach from the spectral filtering regularization point of view [18], an approach that has recently gained some attention due to its ability to study various types of regularization approaches under the same framework. Continuing from (4), we observe that

$$\boldsymbol{\alpha} = \varphi_\lambda(\boldsymbol{\Gamma})\mathrm{vec}(\mathbf{Y}) = \mathbf{W}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^{\mathrm{T}}\mathrm{vec}(\mathbf{Y}),$$

where $\boldsymbol{\Gamma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^{\mathrm{T}}$ is the eigen decomposition of the kernel matrix $\boldsymbol{\Gamma}$ and φ_λ is a filter function, parameterized by λ, such that if $\mathbf{v}$ is an eigenvector of $\boldsymbol{\Gamma}$ and

σ is its corresponding eigenvalue, then $\mathbf{\Gamma v} = \varphi_\lambda(\sigma)\mathbf{v}$. The filter function corresponding to the Tikhonov regularization being $\varphi_\lambda(\sigma) = \frac{1}{\sigma+\lambda}$, and the ordinary least-squares approach corresponding to the $\lambda = 0$ case, several other learning approaches, such as spectral cut-off and gradient descent, can also be expressed as filter functions, but which cannot be expressed as a penalized empirical error minimization problem analogous to (3).

The eigenvalues of the kernel matrix obtained with the tensor product kernel on a complete training set can be expressed as the tensor product $\mathbf{\Lambda} = \mathbf{\Sigma} \otimes \mathbf{S}$ of the eigenvalues $\mathbf{\Sigma}$ and $\mathbf{S}$ of the object and task kernel matrices. Now, instead of considering the two-step learning approach from the kernel point of view, one can also cast it into the spectral filtering regularization framework, resulting to the following filter function:

$$\varphi_\lambda(\sigma) = \frac{1}{(\sigma_1 + \lambda_t)(\sigma_2 + \lambda_d)} = \frac{1}{\sigma_1\sigma_2 + \lambda_d\sigma_1 + \lambda_t\sigma_2 + \lambda_t\lambda_d}\,, \tag{8}$$

where σ_1, σ_2 are the factors of σ, namely eigenvalues of $\mathbf{K}$ and $\mathbf{G}$. This differs from the Tikhonov regularization only by the two middle terms in the denominator if one sets $\lambda = \lambda_t\lambda_d$. In the experiments, we observe that this difference is rather small also in practical cases, making the two-step learning approach a viable alternative for tensor KRR.

We assume Γ being bounded with $\kappa > 0$ such that $\sup_{\mathbf{x}\in\mathcal{X}} \sqrt{\Gamma(\mathbf{x},\mathbf{x})} \leq \kappa$, indicating that the eigenvalues of kernel matrices are in $[0, \kappa^2]$. To further analyze the above filter functions, we follow [4, 6, 18] and say that a function $\varphi_\lambda : [0, \kappa^2] \to \mathbb{R}, 0 < \lambda \leq \kappa^2$, parameterized by $0 < \lambda \leq \kappa^2$, is an admissible regularizer if there exists constants $D, B, \gamma \in \mathbb{R}$ and $\bar{\nu}, \gamma_\nu > 0$ such that

$$\sup_{0<\sigma\leq\kappa^2} |\sigma\varphi_\lambda(\sigma)| \leq D, \quad \sup_{0<\sigma\leq\kappa^2} |\varphi_\lambda(\sigma)| \leq \frac{B}{\lambda}, \quad \sup_{0<\sigma\leq\kappa^2} |1 - \sigma\varphi_\lambda(\sigma)| \leq \gamma\,,$$

$$\text{and} \quad \sup_{0<\sigma\leq\kappa^2} |1 - \sigma\varphi_\lambda(\sigma)|\sigma^\nu \leq \gamma_\nu\lambda^\nu, \forall\nu \in (0, \bar{\nu}].$$

The admissibility, in turn, ensures that

$$R(\hat{f}^\lambda) - \inf_{f\in\mathcal{H}} R(f) = \mathcal{O}\left(n^{-\frac{\bar{\nu}}{2\bar{\nu}+1}}\right) \tag{9}$$

holds with high probability, where R denotes the expected prediction error with respect to some unknown probability measure $\rho(\mathbf{x}, y)$ on the joint space $\mathcal{X} \times \mathbb{R}$ of inputs and labels that is, $R(f) = \int_{\mathcal{X}\times\mathbb{R}} (f(\mathbf{x}) - y)^2 d\rho(\mathbf{x}, y)$. We refer to [4, 6, 18] for a detailed consideration and further results. It is straightforward to see that, analogously to the Tikhonov regularization, the admissibility of the function (8) is confirmed by $D, B, \gamma, \gamma_\nu, \bar{\nu} = 1$ for arbitrary factorizations of $\lambda = \lambda_t\lambda_d$ and $\sigma = \sigma_1\sigma_2$ such that $\lambda_t, \lambda_d > 0$ and $\sigma_1, \sigma_2 \geq 0$. Thus, function (8) can be considered under the spectral filtering regularization framework with separate regularization parameter values for objects and tasks. The universality of the kernel ensures that $\inf_{f\in\mathcal{H}} R(f)$ in (9) is the error of the underlying regression function to be learned, and the admissibility of the regularizer ensures that $R(\hat{f}^\lambda)$ converges to it when the size of the training set approaches infinity. This, in turn, guarantees the consistency of the two-step KRR method.

4 Experiments

We compare different types of transfer learning settings in solving three dyadic prediction problems: drug-target, document similarity and protein similarity prediction. We simulate the full and almost full cold start problem as follows. In each experiment, one drug, document or protein is considered to be the target task in question, where the task is to predict the interactions of drugs or similarities of documents or proteins with respect to the target. Further, other tasks formed in the same way are provided as auxiliary information, leading to a full cold start or almost full cold start setting. The experiments are performed 100 times with different training/test set splits. The performances are averages over all repetitions and over all target tasks, and are measured using the concordance index [11] (C-index), also known as the pairwise ranking accuracy $\frac{1}{|\{(i,j)|y_i>y_j\}|}\sum_{y_i>y_j} H(\hat{y}_i - \hat{y}_j)$, where y_i denote the true and $\hat{y}_i$ the predicted labels, and H is the Heaviside step function. The regularization parameter selection is performed using LOOCV on the training data. For the two-step approach, we select the first regularization parameter via LOOCV on the auxiliary tasks, and the second one via LOOCV on the target task data augmented with predictions from the first step. The algorithms used in the experiments are implemented in the RLScore open source machine learning library[4].

The drug-target interaction prediction data[5] [9, 22] consists of 68 drug compounds and 442 protein targets. The kernel between the drugs is based on the 3D Tanimoto coefficient similarity, and the sequence similarity between the protein targets was computed using the normalized version of the Smith-Waterman score. Further, for each drug-protein pair we have a real-valued label, negative logarithm of the kinase disassociation constant K_d, that characterizes the interaction affinity between the drug and target in question. In each experiment, the task of interest corresponds to one of the drugs in the data set. The goal is to learn to predict for the given drug the K_d values for proteins unseen during the training phase. The performances are always computed over a testing set of 192 protein targets for a given task, i.e. we assess whether for a given target we can discriminate between proteins with more or less affinity for this drug.

For each task, we vary the number of available training proteins, from 5 to 250. In addition, we have available the training data for the 250 training proteins for the 67 auxiliary tasks. As summarized in Figure 1, we evaluate a number of different approaches:

- Single-task: KRR trained with data from the target task only
- Multi-task: both the target and auxiliary tasks have the same amount of training data available (multi-output learning leveraging task correlations, tackled with tensor KRR)
- Full cold start: tensor KRR with no data for the target task
- Almost full cold start: use a varying amount of data from the target task, and all the available data from auxiliary tasks (tackled with two-step KRR)

[4] Available at `https://github.com/aatapa/RLScore`
[5] `http://users.utu.fi/aatapa/data/DrugTarget`

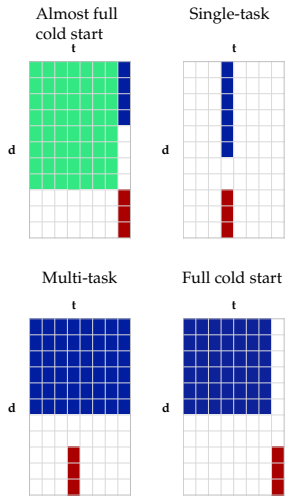

Fig. 1: Overview of the approaches investigated in this article. Green = training data of which the size is constant in the experiments. Blue = training data of which the number of objects varies over different experiments. Red = test data.

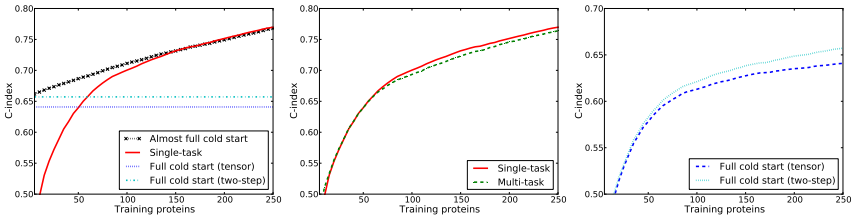

Fig. 2: Learning curves for the drug-target data. Left: target data increased, Middle: target and auxiliary data increased, Right: auxiliary data increased.

We do not consider tensor KRR in the almost full cold start experiment due to computational considerations. Unlike for the two-step KRR no closed-form solution exists for the method in this setting, and the iterative conjugate gradient based method has rather poor scalability.

In Figure 2, we present the results for the drug-target experiments. In Figure 2 (a) we present an experiment, where all the 67 auxiliary tasks have available the data for all 250 training proteins, and the amount of data available for the target task is varied. It can be seen that learning is possible even in the full cold start setting, where both two-step KRR and tensor KRR perform much better than randomly. The single-task approach begins to outperform the full cold start setting after the point when one has access to a bit more than 50 training proteins. Combining these two sources of information leads to the best performance

up until 150 training proteins. However, once there is enough data available for the target task, there is no longer any positive transfer from the auxiliary tasks.

In Figure 2 (b) we consider the setting, where there is the same amount of data available for both the auxiliary tasks and the target tasks. This setting corresponds closely to the traditional multi-output regression problem, the exception being that only the label for the target task is of interest during testing. Here we can see that the multi-task method that uses the task correlation information fails to outperform the simple single-task approach, suggesting that on this type of data one requires significantly more data in the auxiliary tasks compared to the target tasks in order for it to be helpful for learning.

In Figure 2 (c) we consider the full cold start learning setting, while increasing the amount of data available for the auxiliary tasks. Here we observe that the simple two-step approach slightly outperforms tensor KRR, possibly due to the property that it allows regularizing the drugs and the targets separately. Both approaches generalize to the unknown target task, though the results are still much worse than when having significant amount of data for the target task.

Further, we experiment on the 20 Newsgroups data[6]. Here, given any target document, the goal is to predict the similarity of other documents with respect to it. This constitutes a three-level ordinal regression task, where documents from the same newsgroup as the target receive the highest rating, documents from similar newsgroups the second highest, and the rest the lowest rating. These similarities are assigned according to the taxonomy available at the data set web site. We use the bag-of-words feature representation. The number of target domain data ranges from 50 to 1500 documents (transfer learning, single-task, multi-task methods), and the number of auxiliary tasks and data available for each either ranges from 50 to 1500 documents (multi-task, full cold start learning), or stays fixed at 2000 documents (transfer learning).

The results are presented in Figure 3. For the transfer learning approaches, already 50 target domain documents suffices to reach the performance of the single-task method with 1500 documents. The multi-task learning setting does not outperform the single-task setting, and while learning is possible in the full cold start setting, some target task data is still required to reach a high predictive performance. Two-step learning slightly outperforms tensor KRR.

The UniProt data was generated from all the protein amino acid sequences with all the gene ontology (GO) annotations of the Universal Protein Resource (UniProt) database. For the amino acid sequences we used the normalized spectrum kernel [17]. This kernel is a popular tool for comparing biological sequences without alignments. The normalized spectrum kernel is based on the number of k-mers two sequences have in common. In our experiments, k was set to three. Two proteins were labeled as 'similar in function' when they had at least one GO term in common, resulting in a binary classification problem. The experimental setup is the same as for the Newsgroup data, and the results, presented in Figure 3, are very similar, though at 1500 proteins the performance of the two-step method actually falls below that of the single-task approach.

[6] `http://qwone.com/~jason/20Newsgroups/`

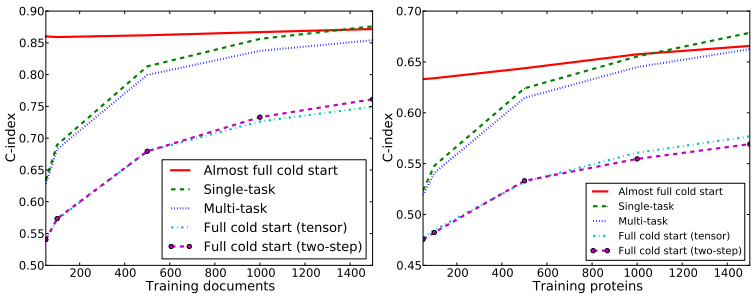

Fig. 3: Learning curves for 20 Newsgroups (left) and Uniprot (right).

In all experiments two-step KRR shows itself to be competitive compared to tensor KRR. Previously, [33] have in their overview article on dyadic prediction in the biological domain made the observation that in terms of predictive accuracy there does not seem to be a clear winner between the single-task and multi-task type of learning approaches. Based on our experimental results, a deciding factor on whether one may expect positive transfer from related tasks seems to be the amount of data available for the target task. The two-step method performs well in the almost full cold start settings with availability of a significant amount of auxiliary data and only very little data for the target task. But when there is enough data available for the target task, auxiliary data is no longer helpful.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments.

References

[1] Adams, R.P., Dahl, G.E., Murray, I.: Incorporating side information into probabilistic matrix factorization using Gaussian processes. The 26th Conference on Uncertainty in Artificial Intelligence, pp. 1–9 (2010)
[2] Álvarez, M., Rosasco, L., Lawrence, N.: Kernels for vector-valued functions: a review. Foundation and Trends in Machine Learning 4(3), 195–266 (2012)
[3] Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society 68 (1950)
[4] Baldassarre, L., Rosasco, L., Barla, A., Verri, A.: Multi-output learning via spectral filtering. Machine Learning 87(3), 259–301 (2012)
[5] Basilico, J., Hofmann, T.: Unifying collaborative and content-based filtering. 21st international conference on Machine learning (ICML'04) (2004)
[6] Bauer, F., Pereverzev, S., Rosasco, L.: On regularization algorithms in learning theory. Journal of Complexity 23(1), 52–72 (2007)
[7] Ben-Hur, A., Noble, W.: Kernel methods for predicting protein-protein interactions. Bioinformatics 21 Suppl 1, 38–46 (2005)

[8] Bonilla, E.V., Agakov, F., Williams, C.: Kernel multi-task learning using task-specific features. The 11th International Conference on Artificial Intelligence and Statistics AISTATS'07, pp. 43–50 (2007)
[9] Davis, M.I., Hunt, J.P., Herrgard, S., Ciceri, P., Wodicka, L.M., Pallares, G., Hocker, M., Treiber, D.K., Zarrinkar, P.P.: Comprehensive analysis of kinase inhibitor selectivity. Nature biotechnology 29(11), 1046–1051 (2011)
[10] Fang, Y., Si, L.: Matrix co-factorization for recommendation with rich side information and implicit feedback. 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, pp. 65–69 (2011)
[11] Gönen, M., Heller, G.: Concordance probability and discriminatory power in proportional hazards regression. Biometrika 92(4), 965–970 (2005)
[12] Hayashi, K., Takenouchi, T., Tomioka, R., Kashima, H.: Self-measuring similarity for multi-task gaussian process. ICML Workshop on Unsupervised and Transfer Learning, JMLR Proceedings, vol. 27, pp. 145–154 (2012)
[13] Jacob, L., Vert, J.: Protein-ligand interaction prediction: an improved chemogenomics approach. Bioinformatics 24(19), 2149–2156 (2008)
[14] Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., Tsuda, K.: Link propagation: A fast semi-supervised learning algorithm for link prediction. SIAM International Conference on Data Mining (SDM'09), pp. 1099–1110 (2009)
[15] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
[16] Larochelle, H., Erhan, D., Bengio, Y.: Zero-data learning of new tasks. 23rd national conference on Artificial intelligence (AAAI'08), pp. 646–651 (2008)
[17] Leslie, C., Eskin, E., Noble, W.S.S.: The spectrum kernel: a string kernel for SVM protein classification. Pacific Symposium on Biocomputing, pp. 564–575 (2002)
[18] Lo Gerfo, L., Rosasco, L., Odone, F., De Vito, E., Verri, A.: Spectral algorithms for supervised learning. Neural Computation 20(7), 1873–1897 (2008)
[19] Martin, C.D., Van Loan, C.F.: Shifted Kronecker product systems. SIAM Journal on Matrix Analysis and Applications 29(1), 184–198 (2006)
[20] Menon, A., Elkan, C.: A log-linear model with latent features for dyadic prediction. The 10th IEEE International Conference on Data Mining (ICDM), pp. 364–373 (2010)
[21] Oyama, S., Manning, C.: Using feature conjunctions across examples for learning pairwise classifiers. European conference on Machine learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 3201, pp. 322–333 (2004)
[22] Pahikkala, T., Airola, A., Pietilä, S., Shakyawar, S., Szwajda, A., Tang, J., Aittokallio, T.: Toward more realistic drug-target interaction predictions. Briefings in Bioinformatics (2014), In press. DOI: 10.1093/bib/bbu010
[23] Pahikkala, T., Airola, A., Stock, M., Baets, B.D., Waegeman, W.: Efficient regularized least-squares algorithms for conditional ranking on relational data. Machine Learning 93(2–3), 321–356 (2013)
[24] Pahikkala, T., Suominen, H., Boberg, J.: Efficient cross-validation for kernelized least-squares regression with sparse basis expansions. Machine Learning 87(3), 381–407 (2012)

[25] Pahikkala, T., Waegeman, W., Airola, A., Salakoski, T., De Baets, B.: Conditional ranking on relational data. European conference on Machine learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 6322, pp. 499–514 (2010)
[26] Pahikkala, T., Waegeman, W., Tsivtsivadze, E., Salakoski, T., De Baets, B.: Learning intransitive reciprocal relations with kernel methods. European Journal of Operational Research 206(3), 676–685 (2010)
[27] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
[28] Park, S.T., Chu, W.: Pairwise preference regression for cold-start recommendation. 3rd ACM Conference on Recommender Systems, pp. 21–28 (2009)
[29] Park, Y., Marcotte, E.M.: Flaws in evaluation schemes for pair-input computational predictions. Nature Methods 9(12), 1134–1136 (2012)
[30] Raymond, R., Kashima, H.: Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. European conference on Machine learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 6323, pp. 131–147 (2010)
[31] Rifkin, R., Lippert, R.: Notes on regularized least squares. Tech. Rep. MIT-CSAIL-TR-2007-025, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA (2007)
[32] Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.: Input space versus feature space in kernel-based methods. IEEE Transactions On Neural Networks 10(5), 1000–1017 (1999)
[33] Schrynemackers, M., Küffner, R., Geurts, P.: On protocols and measures for the validation of supervised methods for the inference of biological networks. Frontiers in Genetics 4, 262 (2013)
[34] Schrynemackers, M., Wehenkel, L., Babu, M.M., Geurts, P.: Classifying pairs with trees for supervised biological network inference. Submitted manuscript (2014)
[35] Shan, H., Banerjee, A.: Generalized probabilistic matrix factorizations for collaborative filtering. The 10th IEEE International Conference on Data Mining (ICDM), pp. 1025–1030 (2010)
[36] Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. Journal of Machine Learning Research 2, 67–93 (2002)
[37] Van Loan, C.F.: The ubiquitous kronecker product. Journal of Computational and Applied Mathematics 123(1–2), 85–100 (2000)
[38] Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., Stock, M., De Baets, B.: A kernel-based framework for learning graded relations from data. IEEE Transactions on Fuzzy Systems 20(6), 1090–1101 (2012)
[39] Zhou, T., Shan, H., Banerjee, A., Sapiro, G.: Kernelized probabilistic matrix factorization: Exploiting graphs and side information. 12th SIAM International Conference on Data Mining, pp. 403–414 (2012)