LETTER Communicated by Hanxiao Liu

A Comparative Study of Pairwise Learning Methods Based on Kernel Ridge Regression

Michiel Stock
michiel.stock@ugent.be
KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, 9000 Ghent, Belgium

Tapio Pahikkala
aatapa@utu.fi
Antti Airola
ajairo@utu.fi
Department of Future Technologies, University of Turku, 20520 Turku, Finland

Bernard De Baets
bernard.debaets@ugent.be
Willem Waegeman
willem.waegeman@ugent.be
KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, 9000 Ghent, Belgium

Many machine learning problems can be formulated as predicting labels for a pair of objects. Problems of that kind are often referred to as pairwise learning, dyadic prediction, or network inference problems. During the past decade, kernel methods have played a dominant role in pairwise learning. They still obtain a state-of-the-art predictive performance, but a theoretical analysis of their behavior has been underexplored in the machine learning literature. In this work we review and unify kernel-based algorithms that are commonly used in different pairwise learning settings, ranging from matrix filtering to zero-shot learning. To this end, we focus on closed-form efficient instantiations of Kronecker kernel ridge regression. We show that independent task kernel ridge regression, two-step kernel ridge regression, and a linear matrix filter arise naturally as a special case of Kronecker kernel ridge regression, implying that all these methods implicitly minimize a squared loss. In addition, we analyze universality, consistency, and spectral filtering properties. Our theoretical results provide valuable insights into assessing the advantages and limitations of existing pairwise learning methods.

Neural Computation **30, 2245–2283** (2018)
doi:10.1162/neco_a_01096

1 Introduction to Pairwise Learning

1.1 Settings in Pairwise Learning. Many real-world machine learning problems can naturally be represented as pairwise learning or dyadic prediction problems. In contrast to more traditional learning settings, the goal here consists of making predictions for pairs of objects $u \in \mathcal{U}$ and $v \in \mathcal{V}$, as elements of two universes $\mathcal{U}$ and $\mathcal{V}$. Such an ordered pair (u, v) is often referred to as a dyad, and both elements in the dyad are usually equipped with a feature representation. In contrast to many statistical settings, these dyads are not independently and identically distributed, as the same objects tend to appear many times as part of different pairs.

Applications of pairwise learning often arise in the life sciences, such as predicting various types of interactions in all sorts of biological networks (e.g., drug-target networks, gene regulatory networks, and species interaction networks). Similarly, pairwise learning methods are also used to extract novel relationships in social networks, such as author-citation networks. Other popular applications include recommender systems (predicting interactions between users and items) and information retrieval (predicting interactions between search queries and search results).

Formally speaking, in pairwise learning, one attempts to learn a function of the form $f(u, v)$, that is, a function to predict properties of two objects. Such functions are fitted using a set of n labeled examples: the training set $S = \{(u_h, v_h, y_h) \mid h = 1, \ldots, n\}$. Further on, $U = \{u_i \mid i = 1, \ldots, m\}$ and $V = \{v_j \mid j = 1, \ldots, q\}$ will denote the sets of distinct objects of both types, later referred to as instances and tasks, respectively, in the training set with $m = |U|$ and $q = |V|$.

Pairwise learning holds strong connections with many other machine learning settings. Especially a link with multitask learning can be advocated by calling the first object of a dyad an "instance" and the second object a "task." The underlying idea for making the distinction between instances and tasks is that the feature description of the instances is often considered as more informative, while the feature description of the tasks is mainly used to steer learning in the right direction. Albeit less common in traditional multitask learning formulations, feature representations for tasks play a crucial role in recent paradigms such as zero-shot learning (see, e.g., Palatucci, Hinton, Pomerleau, & Mitchell, 2009; Lampert, Nickisch, & Harmeling, 2014).

The connection between pairwise learning and multitask learning allows one to distinguish different prediction settings that are crucial in the context of this letter. Formally, four settings for predicting the label of the dyad (u, v) can be distinguished in pairwise learning, based on whether testing objects are in-sample (appear in the training data) or out-of-sample (do not appear in the training data):

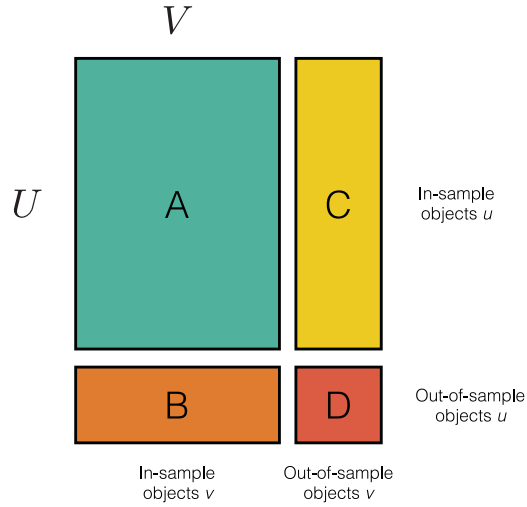

Figure 1: The different prediction settings in pairwise learning, depending on whether objects in a dyad occurred in the training set. The four settings are always referred to as settings A, B, C, and D, respectively, in this letter.

- *Setting A*: Both u and v are observed during training, as parts of different dyads, but the label of the dyad (u, v) must be predicted.
- *Setting B*: Only v is known during training, while u is not observed in any training dyad, and the label of the dyad (u, v) must be predicted.
- *Setting C*: Only u is known during training, while v is not observed in any training dyad, and the label of the dyad (u, v) must be predicted.
- *Setting D*: Neither u nor v occurs in any training dyad, and the label of the dyad (u, v) must be predicted.

Figure 1 shows data of the four settings graphically in four matrix representations. Setting A resembles a matrix completion or matrix filtering scenario, as typically encountered in collaborative filtering problems. In principle, feature representations are not needed if the structure of the matrix is exploited to generate predictions, but additional information might be helpful. Setting B resembles a classical multitask learning scenario, where the columns represent instances and the rows tasks. For a predefined set of tasks, one aims for predicting the labels of novel instances. Setting C then considers the converse setting, where the instances are all known during training and some tasks are unobserved. This setting is in essence identical to setting B if one interchanges the notions of task and instance.

Setting D is the most difficult prediction setting of the four. In the multitask learning literature, this setting is known as zero-shot learning, as one aims for predicting the labels of tasks with zero training data.

In pairwise learning, it is extremely important to distinguish these four prediction scenarios. Without bearing them in mind, one might select the wrong model for the given scenario or obtain an under- or overestimation of the generalization error. For example, a pairwise recommender system that can generalize well to new users might perform poorly for new items. In a large-scale metastudy about biological network identification, it was found that these concepts are vital to correctly evaluate pairwise learning models (Park & Marcotte, 2012). Certain properties of different models discussed in this work hold only for certain settings.

1.2 Kernel Methods for Pairwise Learning with Complete Data Sets. During the past decade, various types of methods for pairwise learning have been proposed in the literature. Kernel methods in particular have been extensively used (see, e.g., Vert & Yamanishi, 2005; Zaki, Lazarova-Molnar, El-Hajj, & Campbell, 2009; Huynh-Thu, Irrthum, Wehenkel, & Geurts, 2010; van Laarhoven, Nabuurs, & Marchiori, 2011; Cao et al., 2012; Liu & Yang, 2015). Especially in bioinformatics applications, they have been popular because biological entities are often easier to represent in terms of similarity scores than feature representations (Ben-Hur & Noble, 2005; Shen et al., 2007; Vert, Qiu, & Noble, 2007).[1]

In this work, we focus on kernel methods for pairwise learning. We believe that kernel methods have a number of appealing properties:

First, the methods that we analyze in this letter are general-purpose methods. They can be applied to a wide range of settings, including settings A to D and a wide range of application domains. More recent methods might outperform kernel methods in specific situations, but they are usually not applicable to settings A to D at the same time, or they are mainly developed for specific application domains with very specific types of data sets (e.g., computer vision and text mining data sets).

Second, the methods that we analyze often form an essential building block of more recent (and more complicated) methods. This is, for example, the case for zero-shot learning methods in computer vision. It is therefore important to provide a theoretical analysis of older methods in order to gain a better understanding of more recent methods that are often black-box engineering approaches. More details on this aspect are given section 4.

[1] Recent advances in convolutional neural networks, however, have resulted in intriguing ways to generate representations for molecules (Duvenaud et al., 2015), proteins (Jo, Hou, Eickholt, & Cheng, 2015) and nucleic acids (Alipanahi, Delong, Weirauch, & Frey, 2015). Such feature representations, obtained by pretraining on large data sets, will likely replace kernel methods in the future, at least to some extent.

Third, the methods that we analyze in this paper are still clear winners for specific scenarios. One of those scenarios is cross-validation in pairwise learning, for which kernel methods outperform other methods substantially with regard to computational scalability. Furthermore, scalable and exact algorithms can be derived to learn a model online or when the data set is not complete (see definition 1). For more information on these aspects, we refer readers to our complementary work (Stock, De Baets, & Waegeman, 2017; Stock, 2017; Stock, Pahikkala, Airola, Waegeman, & De Baets, 2018).

These three reasons are the key motivations for studying kernel-based pairwise learning methods from a theoretical perspective. The key idea of extending kernel methods to pairwise learning is to construct so-called pairwise kernels, which measure the similarity between two dyads (u, v) and $(\bar{u}, \bar{v})$. Kernels of that kind can be used in tandem with any conventional kernelized learning algorithm such as support vector machines, kernel ridge regression (KRR), and kernel Fisher discriminant analysis. In this letter, we particularly focus on pairwise learning methods that are inspired by kernel ridge regression. Due to the algebraic properties of such methods, they are especially useful when analyzing so-called complete data sets in pairwise learning.

Definition 1 *(complete dataset). A training set is called complete if it contains exactly one labeled example for every dyad* $(u, v) \in U \times V$.

If the label matrix contains only a few missing labels, matrix imputation methods can be applied to render the matrix complete (Mazumder, Hastie, & Tibshirani, 2010; Stekhoven & Bühlmann, 2012; Zachariah & Sundin, 2012). Complete data sets, however, occur frequently, for example, in biological networks such as drug-protein interactions or species interactions. Here, screenings or field studies generate a set of observed interactions, while interactions that are not observed are either interactions not occurring or false negatives (Schrynemackers, Küffner, & Geurts, 2013; Jordano, 2016). In such cases, the positive instances are labeled 1 and the negatives are labeled 0. Theoretical work by Elkan and Noto (2008) has shown that models can still be learned from such data sets. Outside of biological network inference, complete datasets occur in recommender systems with implicit feedback; for example, buying a book can be seen as a proxy for liking a book (Isinkaye, Folajimi, & Ojokoh, 2015). Setting A (reestimating labels) is still relevant for such data sets if the labels are noisy or contain false positives or false negatives. A pairwise learning model can be used to detect and curate such errors.

For a complete training set, 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 U and the columns by the objects in V. Furthermore, we use $\mathbf{Y}_{i.}$ (resp. $\mathbf{Y}_{.j}$) to denote the ith row (resp. jth column) of $\mathbf{Y}$. The vectorization of the matrix $\mathbf{Y}$ by stacking its columns in one long vector will be denoted $\mathbf{y}$.

1.3 Scope and Objectives of This Letter. The goal of this letter is to provide theoretical insight into the working of existing pairwise learning methods that are based on kernel ridge regression. To this end, we focus on scenarios with complete training data sets while analyzing the behavior for settings A to D. More specifically, we provide an in-depth discussion of the following four methods:

- *Kronecker kernel ridge regression*: Adopting a least-squares formulation, this method is representative for many existing systems based on pairwise kernels.
- *Two-step kernel ridge regression*: This recent method has some interesting properties such as simplicity and computational efficiency. The method has been independently proposed in Pahikkala et al. (2014) and Romera-Paredes and Torr (2015). In a variant of it, tree-based methods replace kernel ridge regression as base learners (Schrynemackers, Wehenkel, Babu, & Geurts, 2015). In a statistical context, similar models have been developed for structural equation modelling (Bollen, 1996; Bollen & Bauer, 2004; Jung, 2013).
- *Linear matrix filtering*: This recently proposed method provides predictions in setting A without the need for object features, similar to collaborative filtering methods. Though simple, this linear filter was found to perform very well in predicting interactions in a variety of species-species and protein-ligand interaction datasets (Stock, Poisot, Waegeman, & De Baets, 2017; Stock, 2017). On these data sets, it outperforms standard matrix factorization methods and is very tolerant of a large number of false negatives in the label matrices.
- *Independent-task kernel ridge regression*: This method serves as a baseline and a building block for some of the other methods. This approach resembles the traditional kernel ridge regression method, applied to each task (i.e., each column of $\mathbf{Y}$) separately. When the method is applied to a single task, we speak of single-task kernel ridge regression.

We review these four models in section 2. All can be represented using two positive-semidefinite kernel functions, one for each type of object—$k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ and $g : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$. These capture the similarity between two objects of the same types. We will with prediction functions of the form

$$f(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij} k(u, u_i) g(v, v_j), \tag{1.1}$$

with $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m \times q}$ the dual parameters. Such a model can, for instance, be obtained by the pairwise Kronecker kernel in a kernel-based learning algorithm such as support vector machines (e.g., Vert et al., 2007; Brunner

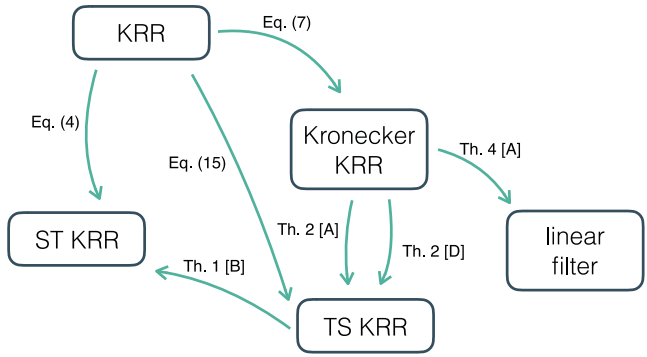

Figure 2: Overview of the methods discussed in this work and their relation to one other. KRR = kernel ridge regression; ST KRR = single-task kernel ridge regression; TS KRR = two-step kernel ridge regression. The letters in brackets indicate the settings for which the theorem holds, as shown in Figure 1.

& Fischer, 2012). In this work, we limit ourselves to models where the dual parameters can be written as a linear combination of the label matrix:

$$\text{vec}(\mathbf{A}) = \mathbf{B}\text{vec}(\mathbf{Y}). \tag{1.2}$$

Here, $\mathbf{B} \in \mathbb{R}^{mq \times mq}$ is a matrix constructed based on the training objects U and V, the kernel functions and the learning algorithm, but not the labels of the pairs. Similarly, the matrix containing the predictions $\mathbf{F}$ associated with the labels can be obtained by

$$\text{vec}(\mathbf{F}) = \mathbf{H}\text{vec}(\mathbf{Y}), \tag{1.3}$$

where $\mathbf{H} \in \mathbb{R}^{mq \times mq}$ is the so-called hat matrix which maps observations to predictions (Hastie, Tibshirani, & Friedman, 2001). Although $\mathbf{B}$ and $\mathbf{H}$ are huge matrices for problems of even modest size (e.g., if $|U|$ and $|V|$ are on the order of thousands, these matrices have a cardinality of millions); for several methods, the parameters and predictions can be computed efficiently. More specifically, the learning algorithms discussed in this work scale with the number of objects rather than the number of labels.

The learning properties of the four methods are theoretically analyzed in section 3. In a first series of results, we establish equivalences using special kernels and algebraic operations. We discuss several links that are specific for settings A, B, C, or D. Figure 2 gives an overview of what readers might expect to learn. In a second series of results, we prove the universality of Kronecker product pairwise kernels, and we analyze the consistency of the algorithms that can be derived from such kernels. To this end, we provide a

spectral interpretation of Kronecker and two-step kernel ridge regression. This will give further insight into the behavior of these methods.

2 Pairwise Learning with Methods Based on Kernel Ridge Regression

In this section we formally review the four methods outlined in section 1. We start by explaining a baseline multitask learning formulation that will be needed to understand more complicated methods. We call this method independent-task kernel ridge regression, since it constructs independent models for the different tasks, that is, the different columns of $\mathbf{Y}$. Subsequently, we elaborate on Kronecker kernel ridge regression as an instantiation of a method that employs pairwise kernels. In sections 2.3 and 2.4, we review two-step kernel ridge regression and the linear matrix filter. In what follows we adopt a multitask learning formulation in which the objects of $\mathcal{U}$ and $\mathcal{V}$ are referred to as instances and tasks, respectively.

2.1 Independent-Task Kernel Ridge Regression. Suppose that only features of objects of type $\mathcal{U}$ are available, but not of type $\mathcal{V}$. Since there is no information available on how the tasks are related, a separate model for each task is trained. Let $\mathbf{Y}_{\cdot j} \in \mathbb{R}^m$ be the labels of task v_j and $k(\cdot, \cdot)$ be a suitable kernel function that quantifies the similarity of the different instances. Since a separate and independent model is trained for each task, we denote this setting as independent task (IT) kernel ridge regression. For each task v_j, one would like to learn a function of the form

$$f_j^{\mathrm{IT}}(u) = \sum_{i=1}^{m} a_{ij}^{\mathrm{IT}} k(u, u_i),$$

with a_{ij}^{IT} parameters that minimize a suitable objective function. In the case of KRR, this objective function is the squared loss with an L_2-complexity penalty. The parameters for the individual tasks using KRR can be found jointly by minimizing the following objective function (Wahba, 1990; Bishop, 2006):

$$J(\mathbf{A}^{\mathrm{IT}}) = \mathrm{tr}[(\mathbf{K}\mathbf{A}^{\mathrm{IT}} - \mathbf{Y})^\top (\mathbf{K}\mathbf{A}^{\mathrm{IT}} - \mathbf{Y})] + \lambda_u \mathrm{tr}[\mathbf{A}^{\mathrm{IT}\top} \mathbf{K}\mathbf{A}^{\mathrm{IT}}], \tag{2.1}$$

with $\mathrm{tr}(\cdot)$ the trace, $\mathbf{A}^{\mathrm{IT}} = [a_{ij}^{\mathrm{IT}}] \in \mathbb{R}^{m \times q}$ and $\mathbf{K} \in \mathbb{R}^{m \times m}$ the Gram matrix associated with the kernel function $k(\cdot, \cdot)$ for the instances and λ_u a regularization parameter. For simplicity, we assume the same regularization parameter λ_u for each task v, though extensions to different penalties for different tasks are straightforward. This basic setting assumes no cross talk between the tasks, as each model is fitted independently. The optimal

coefficients that minimize equation 2.1 can be found by solving the following linear system:

$$(\mathbf{K} + \lambda_u \mathbb{I})\, \mathbf{A}^{\mathrm{IT}} = \mathbf{Y}. \tag{2.2}$$

Using the singular value decomposition of the Gram matrix, this system can be solved for any value of λ_u with a time complexity of $\mathcal{O}(m^3 + m^2 q)$.

2.2 Pairwise and Kronecker Kernel Ridge Regression. Suppose one has prior knowledge about which tasks are more similar, quantified by a kernel function $g(\cdot, \cdot)$ defined over the tasks. Several authors (see Álvarez, Rosasco, & Lawrence, 2012; Baldassarre, Rosasco, Barla, & Verri, 2012) have extended KRR to incorporate task correlations via matrix-valued kernels. However, most of this literature concerns kernels for which the tasks are fixed at training time. An alternative approach, allowing for the generalization to new tasks more straightforwardly by means of such a task kernel, is to use a pairwise kernel $\Gamma\left((u, v), (\bar{u}, \bar{v})\right)$. Pairwise kernels provide a prediction function of the type

$$f(u, v) = \sum_{h=1}^{n} \alpha_h \Gamma\left((u, v), (u_h, v_h)\right), \tag{2.3}$$

where $\boldsymbol{\alpha} = [\alpha_h]$ are parameters that minimize the same objective function as in equation 2.1,

$$J(\boldsymbol{\alpha}) = (\boldsymbol{\Gamma}\boldsymbol{\alpha} - \mathbf{y})^{\top}(\boldsymbol{\Gamma}\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^{\top}\boldsymbol{\Gamma}\boldsymbol{\alpha}, \tag{2.4}$$

with $\boldsymbol{\Gamma}$ the pairwise Gram matrix. The minimizer can also be found by solving a system of linear equations:

$$(\boldsymbol{\Gamma} + \lambda \mathbb{I})\, \boldsymbol{\alpha} = \mathbf{y}. \tag{2.5}$$

The most commonly used pairwise kernel is the Kronecker product pairwise kernel (Basilico & Hofmann, 2004; Oyama & Manning, 2004; Ben-Hur & Noble, 2005; Park & Chu, 2009; Hayashi, Takenouchi, Tomioka, & Kashima, 2012; Bonilla, Agakov, & Williams, 2007; Pahikkala, Airola, Stock, De Baets, & Waegeman, 2013). This kernel is defined as

$$\Gamma^{\mathrm{KK}}\left((u, v), (\bar{u}, \bar{v})\right) = k(u, \bar{u})\, g(v, \bar{v}), \tag{2.6}$$

a product of the data kernel $k(\cdot, \cdot)$ and the task kernel $g(\cdot, \cdot)$. Many other variations of pairwise kernels have been considered to incorporate prior knowledge on the nature of the relations (e.g., Vert et al., 2007; Pahikkala, Waegeman, Tsivtsivadze, Salakoski, & De Baets, 2010; Waegeman et al.,

2012; Pahikkala et al., 2013) or for more efficient calculations in certain settings (e.g., Kashima, Oyama, Yamanishi, & Tsuda, 2010).

Let $\mathbf{G} \in \mathbb{R}^{q \times q}$ be the Gram matrix for the tasks. Then, for a complete training set, the Gram matrix for the instance-task pairs is the Kronecker product $\mathbf{\Gamma} = \mathbf{G} \otimes \mathbf{K}$. Often it is infeasible to use this kernel directly due to its large size. The prediction function 2.3 can be written as

$$f^{\mathrm{KK}}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{KK}} k(u, u_i) g(v, v_j). \tag{2.7}$$

The matrix $\mathbf{F}$ containing the predictions for the training data using a pairwise kernel can be obtained by a linear transformation of the training labels:

$$\begin{aligned} \mathrm{vec}(\mathbf{F}) &= \mathbf{\Gamma} \mathrm{vec}(\mathbf{A}^{\mathrm{KK}}) && (2.8) \\ &= \mathbf{\Gamma} \left(\mathbf{\Gamma} + \lambda \mathbb{I}\right)^{-1} \mathrm{vec}(\mathbf{Y}) && (2.9) \\ &= \mathbf{H}^{\Gamma} \mathrm{vec}(\mathbf{Y}). && (2.10) \end{aligned}$$

As a special case of Kronecker KRR, we also retrieve ordinary Kronecker kernel least squares (OKKLS) when the objective function of equation 2.4 has no regularization term (i.e., $\lambda = 0$).

Several authors have pointed out that while the size of the system in equation 2.5 is considerably large, its solutions for the Kronecker product kernel can be found efficiently via tensor algebraic optimization (Van Loan, 2000; Martin & Van Loan, 2006; Kashima, Kato, Yamanishi, Sugiyama, & Tsuda, 2009; Raymond & Kashima, 2010; Pahikkala et al., 2013; Álvarez et al., 2012). This is because the eigenvalue decomposition of a Kronecker product of two matrices can easily be computed from the eigenvalue decomposition of the individual matrices. The time complexity scales roughly with $\mathcal{O}(m^3 + q^3)$, which is required for computing the singular value decomposition of $\mathbf{K}$ and $\mathbf{G}$ (see property 2 in the appendix), but the complexities can be scaled down even further by using sparse kernel matrix approximation (Mahoney, 2011; Gittens & Mahoney, 2013).

However, these computational shortcuts concern only the case in which the training set is complete. If some of the instance-task pairs in the training set are missing or if there are several occurrences of certain pairs, one has to resort, for example, to gradient-descent-based training approaches (Park & Chu, 2009; Pahikkala et al., 2013; Kashima et al., 2009; Airola & Pahikkala, 2018). While the training can be accelerated via tensor algebraic optimization, such techniques still remain considerably slower than the approach based on eigenvalue decomposition.

2.3 Two-Step Kernel Ridge Regression. Clearly, independent-task ridge regression can generalize to new instances, but not to new tasks as no

dependence between these tasks is encoded in the model. Kronecker KRR, on the other hand, can be used for all four prediction settings depicted in Figure 1. But since our definition of *instances* and *tasks* is purely conventional, nothing is preventing us from building a model using the kernel function $g(\cdot, \cdot)$ to generalize to new tasks for the same instances. By combining two ordinary KRRs, one for generalizing to new instances and one that generalizes to new tasks, one can indirectly predict for new dyads.

More formally, suppose one wants to make a prediction for the dyad (u, v). Let $\mathbf{k} \in \mathbb{R}^m$ denote the vector of instance kernel evaluations between the instances in the training set and an instance in the test set: $\mathbf{k}(u) = (k(u, u_1), \ldots, k(u, u_m))^\top$. Likewise, $\mathbf{g} \in \mathbb{R}^q$ represents the vector of task kernel evaluations between the target task and the auxiliary tasks: $\mathbf{g}(v) = \big(g(v, v_1), \ldots, g(v, v_q)\big)^\top$. Based on the parameters found by solving equation 2.2, we can make a prediction for the new instance u for all the auxiliary tasks,

$$\mathbf{f}_V(u) = \mathbf{k}^\top (\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y}, \tag{2.11}$$

with λ_u the specific regularization parameter for the instances. This vector of predictions $\mathbf{f}_V(u)$ can be used as a set of labels in an intermediate step to train a second model for generalizing to new tasks for the same instance. Thus, using the task kernel and a regularization parameter for the tasks λ_v, one obtains

$$f^{\mathrm{TS}}(u, v) = \mathbf{g}^\top (\mathbf{G} + \lambda_v \mathbb{I})^{-1} \mathbf{f}_V(u)^\top,$$

or, by making use of equation 2.11, the prediction is given by

$$f^{\mathrm{TS}}(u, v) = \mathbf{k}^\top (\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\mathbf{G} + \lambda_v \mathbb{I})^{-1} \mathbf{g} \tag{2.12}$$

$$= \mathbf{k}^\top \mathbf{A}^{\mathrm{TS}} \mathbf{g}, \tag{2.13}$$

with $\mathbf{A}^{\mathrm{TS}}$ the dual parameters. The concept of two-step KRR is illustrated in Figure 3. Two-step KRR can be used for any of the prediction settings discussed in section 1.1. Note that in practice, there is no need to explicitly calculate $\mathbf{f}_V$, nor does it matter if in the first step, one uses a model for new tasks and in the second step for instances, or the other way around.

This model can be cast in a similar form as the pairwise prediction function of equation 2.7 by making use of property 1 in the appendix. Thus, for two-step KRR, the parameters are given by

$$\mathbf{A}^{\mathrm{TS}} = (\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\mathbf{G} + \lambda_v \mathbb{I})^{-1}. \tag{2.14}$$

The time complexity for two-step KRR is the same as for Kronecker KRR: $\mathcal{O}(m^3 + q^3)$. The parameters can also be found by computing the eigenvalue

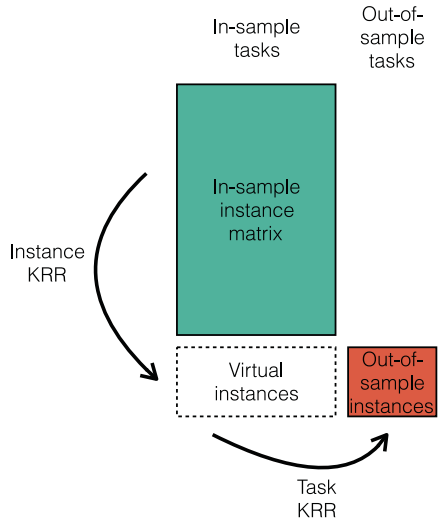

Figure 3: Principle of two-step KRR. In a first step, a virtual prediction is made for the out-of-sample tasks for new instances using a first KRR model. A second KRR model is trained using these data; and this model is used to make predictions for new tasks.

decomposition of the two Gram matrices. Starting from these eigenvalue decompositions, it is possible to directly obtain the dual parameters for any values of the regularization hyperparameters λ_u and λ_v. Because of its conceptual simplicity, it is quite straightforward to use two-step KRR for certain situations when the label matrix is not complete. The algebraic simplicity of two-step KRR can lead to some interesting algorithmic shortcuts for training and validating models. We refer to our other work for a theoretical and experimental overview (Stock, 2017).

2.4 Linear Filter for Matrices. Single-task KRR uses a feature description only for the objects u, while Kronecker and two-step KRR incorporate feature descriptions of both objects u and v. Is it possible to make predictions without any features at all? Obviously this would be possible only for setting A, where both objects are known during training. The structure of the label matrix $\mathbf{Y}$ (e.g., being low rank) often contains enough information to successfully make predictions for this setting. In recommender systems, methods that do not take side features into account are often categorized as collaborative filtering methods (Su & Khoshgoftaar, 2009).

In order to use our framework, we have to construct some feature description in the form of a kernel function. An object u (resp. v) can be described by the observed labels of the dyads that contain the object. In the

context of item recommendation, this seems reasonable: users are described by the ratings they have given to items, and items are described by users' ratings. For example, Basilico and Hofmann (2004) use a kernel based on the Pearson correlation of rating vectors of users to obtain a kernel description of users for collaborative filtering. In bioinformatics, van Laarhoven and colleagues (2011) predict drug-target interactions using so-called gaussian interaction profile kernels, that is, the classical radial basis kernel applied to the corresponding row or column of the label matrix. There is nothing inherently wrong with using the labels to construct feature descriptions or kernels for the object. One should be cautious only when taking a holdout set for model selection or model evaluation; the omitted labels should also be removed from the feature description to prevent overfitting.

Kernels that take observed labels into account, such as the gaussian interaction profile kernel, are in theory quite powerful. Because they can be used to learn nonlinear associations, they lead to more expressive models than matrix factorization. The advantage of using these kernels compared to other collaborative filtering techniques such as matrix factorization, k-nearest neighbors, or restricted Boltzmann machines is that side features can elegantly be incorporated into the model. To this end, one only has to combine the collaborative and content-based kernel matrices, for example, by computing a weighted sum or element-wise multiplication.

Recently, a different method was proposed to make predictions without object features (Stock, Poisot et al., 2017). This method makes a prediction for a couple (u_i, v_j) by aggregating the observed value, the row and column average, and the total average of the label matrix. By analogy with an image filter, this method was called a linear filter (LF) for matrices. The prediction matrix, equation 1.3, is obtained as the following weighted average of averages:

$$\mathbf{F}^{\mathrm{LF}}_{ij} = \alpha_1 \mathbf{Y}_{ij} + \alpha_2 \frac{1}{n} \sum_{k=1}^{n} \mathbf{Y}_{kj} + \alpha_3 \frac{1}{m} \sum_{l=1}^{m} \mathbf{Y}_{il} + \alpha_4 \frac{1}{nm} \sum_{k=1}^{n} \sum_{l=1}^{m} \mathbf{Y}_{kl}, \tag{2.15}$$

where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in [0, 1]^4$. The first term is proportional to the label, and the last term is proportional to the average of all labels. The second (resp. third) term is proportional to the average label in the corresponding column (resp. row). The parameters α_1, α_2, α_3, and α_4 act as weighing coefficients.

As mentioned in section 1, this linear filter can outcompete standard methods such as matrix factorization, and it was observed to be particularly tolerant to a large number of false negatives in the label matrices. An attractive property of the linear filter is that it is possible to derive a computational shortcut for leave-one-pair-out (LOO) cross validation:

$$\mathbf{F}^{\mathrm{LOO}}_{ij} = \frac{\mathbf{F}_{ij} - \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm}\right) \mathbf{Y}_{ij}}{1 - \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm}\right)}. \tag{2.16}$$

This allows one to efficiently compute the prediction value $\mathbf{F}_{ij}^{\mathrm{LOO}}$ using the label matrix except for the value $\mathbf{Y}_{ij}$.

In section 3.2 we will show that this linear filter is a special instance of Kronecker KRR. This filter can hence be written in the form of equation 1.1 with the parameters obtained by solving a system of the form 1.2. In practice, however, one would always prefer to work directly using equation 2.15. The parameters α_1, α_2, α_3, and α_4 can be set by means of leave-one-pair-out cross validation using equation 2.16.

3 Theoretical Considerations

In sections 3.1 and 3.2 we show how the four methods of section 2 are related via special kernels and algebraic equivalences. We establish several links that are specific for settings A, B, C, or D. Therefore, each result is formulated as a theorem that indicates the setting to which it applies in its header. In section 3.3, the universality of the Kronecker product pairwise kernels is proven. This result provides a theoretical justification for the observation that Kronecker-based systems often obtain very satisfactory performance in empirical studies. The universality is also used to prove the consistency of the methods that we analyze. This is done in section 3.4 via a spectral interpretation. In addition, this interpretation also allows us to illustrate that two-step kernel ridge regression adopts a special decomposable filter.

3.1 Equivalence between Two-Step and Other Kernel Ridge Regression Methods.

The relation between two-step kernel ridge regression and independent-task ridge regression is given in the following theorem.

Theorem 1 *(setting B). When the Gram matrix of the tasks $\boldsymbol{G}$ is full rank and λ_v is set to zero, independent-task KRR and two-step KRR return the same predictions for any given training task:*

$$f_j^{IT}(\cdot) \equiv f^{TS}(\cdot, v_j).$$

Proof. The prediction for the independent-task KRR is given by

$$f_j^{\mathrm{IT}}(u) = [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}]_j \,.$$

For two-step KRR, it follows from equation 2.12 that

$$\begin{aligned} f_j^{\mathrm{TS}}(u) &= [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}\mathbf{G}^{-1}\mathbf{G}]_j \\ &= [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}]_j. \end{aligned}$$

□

When $\mathbf{G}$ is singular, the q outputs for the different tasks are projected on a lower-dimensional subspace by two-step KRR. This means that a dependence between the tasks is enforced even when $\lambda_v = 0$.

The connection between two-step and Kronecker KRR is established by the following results.

Theorem 2 *(setting A). Consider the following pairwise kernel matrix:*

$$\boldsymbol{\Xi} = \boldsymbol{G} \otimes \boldsymbol{K} \left(\lambda_u \lambda_v \mathbb{I} \otimes \mathbb{I} + \lambda_v \mathbb{I} \otimes \boldsymbol{K} + \lambda_u \boldsymbol{G} \otimes \mathbb{I}\right)^{-1}.$$

The predictions for the training data $\mathbf{F}$ using pairwise KRR (see equation 2.9) with the above pairwise kernel and regularization parameter $\lambda = 1$ correspond to those obtained with two-step KRR using the kernel matrices $\boldsymbol{K}$, $\boldsymbol{G}$ with respective regularization parameters λ_u and λ_v.

Proof. We will formulate the corresponding empirical risk minimization of equation 2.4 from the perspective of value regularization. Since setting A is an imputation setting, we directly search for the optimal predicted label matrix $\mathbf{F}$ rather than the optimal parameter matrix. Starting from the objective function for Kronecker KRR, the predictions for the training data are obtained through minimizing the following variational function:

$$\begin{aligned}
J(\mathbf{F}) &= \mathrm{vec}(\mathbf{F} - \mathbf{Y})^\top \mathrm{vec}(\mathbf{F} - \mathbf{Y}) + \mathrm{vec}(\mathbf{F})^\top \boldsymbol{\Xi}^{-1} \mathrm{vec}(\mathbf{F}) \qquad (3.1)\\
&= \mathrm{vec}(\mathbf{F} - \mathbf{Y})^\top \mathrm{vec}(\mathbf{F} - \mathbf{Y})\\
&\quad + \mathrm{vec}(\mathbf{F})^\top (\mathbf{G} \otimes \mathbf{K} \left(\lambda_u \lambda_v \mathbb{I} \otimes \mathbb{I} + \lambda_v \mathbb{I} \otimes \mathbf{K} + \lambda_u \mathbf{G} \otimes \mathbb{I}\right)^{-1})^{-1} \mathrm{vec}(\mathbf{F})\\
&= \mathrm{vec}(\mathbf{F} - \mathbf{Y})^\top \mathrm{vec}(\mathbf{F} - \mathbf{Y})\\
&\quad + \mathrm{vec}(\mathbf{F})^\top \left(\mathbf{G}^{-1} \otimes \mathbf{K}^{-1} \left(\lambda_u \lambda_v \mathbb{I} \otimes \mathbb{I} + \lambda_v \mathbb{I} \otimes \mathbf{K} + \lambda_u \mathbf{G} \otimes \mathbb{I}\right)\right) \mathrm{vec}(\mathbf{F})\\
&= \mathrm{vec}(\mathbf{F} - \mathbf{Y})^\top \mathrm{vec}(\mathbf{F} - \mathbf{Y})\\
&\quad + \mathrm{vec}(\mathbf{F})^\top \left(\lambda_u \lambda_v \mathbf{G}^{-1} \otimes \mathbf{K}^{-1} + \lambda_u \mathbb{I} \otimes \mathbf{K}^{-1} + \lambda_v \mathbf{G}^{-1} \otimes \mathbb{I}\right) \mathrm{vec}(\mathbf{F})\\
&= \mathrm{tr}((\mathbf{F} - \mathbf{Y})^\top (\mathbf{F} - \mathbf{Y}) + \lambda_u \lambda_v \mathbf{F}^\top \mathbf{K}^{-1} \mathbf{F} \mathbf{G}^{-1}\\
&\quad + \lambda_u \mathbf{F}^\top \mathbf{K}^{-1} \mathbf{F} + \lambda_v \mathbf{F}^\top \mathbf{F} \mathbf{G}^{-1}).
\end{aligned}$$

The derivative with respect to $\mathbf{F}$ is given by

$$\begin{aligned}
\frac{\partial J(\mathbf{F})}{\partial \mathbf{F}} &= 2(\mathbf{F} - \mathbf{Y} + \lambda_u \lambda_v \mathbf{K}^{-1} \mathbf{F} \mathbf{G}^{-1} + \lambda_u \mathbf{K}^{-1} \mathbf{F} + \lambda_v \mathbf{F} \mathbf{G}^{-1})\\
&= 2(\lambda_u \mathbf{K}^{-1} + \mathbb{I}) \mathbf{F} + 2(\lambda_u \mathbf{K}^{-1} + \mathbb{I})(\lambda_v \mathbf{F} \mathbf{G}^{-1}) - 2\mathbf{Y}\\
&= 2(\lambda_u \mathbf{K}^{-1} + \mathbb{I}) \mathbf{F} (\lambda_v \mathbf{G}^{-1} + \mathbb{I}) - 2\mathbf{Y}.
\end{aligned}$$

Setting it to zero and solving with respect to $\mathbf{F}$ yields

$$\begin{aligned}\mathbf{F} &= (\lambda_u \mathbf{K}^{-1} + \mathbb{I})^{-1} \mathbf{Y} (\lambda_v \mathbf{G}^{-1} + \mathbb{I})^{-1} \\ &= \mathbf{K}(\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\mathbf{G} + \lambda_v \mathbb{I})^{-1} \mathbf{G}.\end{aligned}$$

Comparing with equation 2.14, we note that $\mathbf{F} = \mathbf{K}\mathbf{A}^{\mathrm{TS}}\mathbf{G}$, which proves the theorem. □

Here, we have assumed that $\mathbf{K}$ and $\mathbf{G}$ are invertible. Note that the kernel $\boldsymbol{\Xi}$ can always be obtained as long as $\mathbf{K}$ and $\mathbf{G}$ are positive semidefinite. The relevance of theorem 2 is that it formulates two-step KRR as an empirical risk minimization problem for setting A (see equation 3.1). It is important to note that the pairwise kernel matrix $\boldsymbol{\Xi}$ appears only in the regularization term of this variational problem. The loss function is dependent on only the predicted values $\mathbf{F}$ and the label matrix $\mathbf{Y}$. Using two-step KRR for setting A when dealing with incomplete data is thus well defined. The empirical risk minimization problem of equation 3.1 can be modified so that the squared loss takes only the observed dyads into account:

$$J'(\mathbf{F}) = \sum_{(u,v,y)\in S} (y - f(u,v)))^2 + \mathrm{vec}(\mathbf{F})^\top \boldsymbol{\Xi}^{-1} \mathrm{vec}(\mathbf{F}), \tag{3.2}$$

with S the training set of labeled dyads. In this case, one ends up with a transductive setting. This explains why setting A is the easiest setting to predict, for, as in transductive learning, one only has to predict for a finite number of dyads known during training, in contrast to inductive learning, where the model has to make predictions for any new dyad, a harder problem (Chapelle, Schölkopf, & Zien, 2006). (See Rifkin & Lippert, 2007, and Johnson & Zhang, 2008 for a more in-depth discussion.)

Two-step and Kronecker KRR also coincide in an interesting way for setting D (e.g., the special case in which there is no labeled data available for the target task). This in turn will allow us to show the consistency of two-step KRR via its universal approximation and spectral regularization properties. Theorem 3 shows the relation between two-step KRR and ordinary Kronecker KRR for setting D.

Theorem 3 *(setting D). Consider a setting with a complete training set. Let* $f^{TS}(\cdot,\cdot)$ *be a model trained with two-step KRR and* $f^{OKKLS}(\cdot,\cdot)$ *be a model trained with ordinary Kronecker kernel least-squares regression (OKKLS) using the following pairwise kernel function on* $\mathcal{U} \times \mathcal{V}$*:*

$$\Upsilon((u,v),(\bar{u},\bar{v})) = (k(u,\bar{u}) + \lambda_u \delta(u,\bar{u}))(g(v,\bar{v}) + \lambda_v \delta(v,\bar{v}))), \tag{3.3}$$

where δ is the delta kernel whose value is 1 if the arguments are equal and 0 otherwise. Then for making predictions for instances $u \in \mathcal{U} \setminus U$ and tasks $v \in \mathcal{V} \setminus V$ not seen in the training set, it holds that $f^{TS}(u, v) = f^{OKKLS}(u, v)$.

Proof. From equation 2.12, we have the following dual model for prediction:

$$f^{\mathrm{TS}}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{TS}} k(u, u_i) g(v, v_j),$$

with $\mathbf{A}^{\mathrm{TS}} = [a_{ij}^{\mathrm{TS}}]$ the matrix of parameters. Similarly, the dual representation of the OKKLS (see equation 2.7) using a parameterization $\mathbf{A}^{\mathrm{OKKLS}} = [a_{ij}^{\mathrm{OKKLS}}]$ is given by

$$\begin{aligned} f^{\mathrm{OKKLS}}(u, v) &= \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} \Upsilon\left((u, v), (u_i, v_j)\right) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} (k(u, u_i) + \lambda_u \delta(u, u_i))(g(v, v_j) + \lambda_v \delta(v, v_j))) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} k(u, u_i) g(v, v_j). \end{aligned}$$

In the last step, we used the fact that $u \neq u_i$ and $v \neq v_j$ to drop the delta kernels. Hence, we need to show that $\mathbf{A}^{\mathrm{TS}} = \mathbf{A}^{\mathrm{OKKLS}}$.

By equation 2.14 and denoting $\widetilde{\mathbf{G}} = (\mathbf{G} + \lambda \mathbb{I})^{-1}$ and $\widetilde{\mathbf{K}} = (\mathbf{K} + \lambda \mathbb{I})^{-1}$, we observe that the model parameters $\mathbf{A}^{\mathrm{TS}}$ of the two-step model can also be obtained in the following closed form:

$$\mathbf{A}^{\mathrm{TS}} = \widetilde{\mathbf{K}} \mathbf{Y} \widetilde{\mathbf{G}}. \tag{3.4}$$

The kernel matrix of Υ for setting D can be expressed as $\boldsymbol{\Upsilon} = (\mathbf{G} + \lambda_v \mathbb{I}) \otimes (\mathbf{K} + \lambda_u \mathbb{I})$. The OKKLS problem with kernel Υ being

$$\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}}) = \underset{\mathbf{A} \in \mathbb{R}^{m \times q}}{\arg\min} \left(\mathrm{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon} \mathrm{vec}(\mathbf{A})\right)^{\top} \left(\mathrm{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon} \mathrm{vec}(\mathbf{A})\right),$$

its minimizer can be expressed as

$$\begin{aligned} \mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}}) &= \boldsymbol{\Upsilon}^{-1} \mathrm{vec}(\mathbf{Y}) = \left((\mathbf{G} + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\ &= \mathrm{vec}\left((\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\mathbf{G} + \lambda_v \mathbb{I})^{-1}\right) = \mathrm{vec}\left(\widetilde{\mathbf{K}} \mathbf{Y} \widetilde{\mathbf{G}}\right). \end{aligned} \tag{3.5}$$

Here we again make use of property 1 in the appendix. From equation 3.5, it then follows that $\mathbf{A}^{\text{TS}} = \mathbf{A}^{\text{OKKLS}}$, which proves the theorem. □

3.2 Smoother Kernels Lead to the Linear Filter. Here, we show that using Kronecker KRR in tandem with certain kernels results in the linear filter of section 2.4. When no good features of the objects are available, we propose to use different kernels, agnostic of the true objects:

$$k^{\text{SM}}(u, \bar{u}) = 1 + \theta_u \delta(u, \bar{u}),$$
$$g^{\text{SM}}(v, \bar{v}) = 1 + \theta_v \delta(v, \bar{u}),$$

or, equivalently, as Gram matrices:

$$\mathbf{K} = \mathbb{J}_{m \times m} + \theta_u \mathbb{I}_m \quad \text{and} \quad \mathbf{G} = \mathbb{J}_{q \times q} + \theta_v \mathbb{I}_q. \tag{3.6}$$

Here, θ_u and θ_v are two hyperparameters of the kernels, $\mathbb{J}_{m \times m}$ is an $m \times m$ matrix filled with ones, and $\mathbb{J}_{q \times q}$ is an $q \times q$ matrix filled with ones. We call these kernels smoother kernels for reasons that will become clear. The rationale behind these kernels is quite simple: a kernel that is the identity matrix would imply that all objects are unique and independent; there is no similarity between them. However, using the all-ones matrix encodes all objects being exactly the same; no distinction between two objects can be made. Hence, the kernels of equation 3.6 represent a trade-off between all objects being similar (first part) and all objects being unique (second part). This is controlled by the hyperparameters θ_u and θ_v.

Using these kernels in the Kronecker-based models has an interesting interpretation: the predictions can be written as a weighted sum of averages.

Theorem 4 *(smoother kernels). Predictions using Kronecker KRR for setting A using the Gram matrices, equation 3.6, are of the form*

$$f(u_i, v_j) = \alpha_1 \mathbf{Y}_{ij} + \alpha_2 \frac{1}{q} \sum_{l=1}^{q} \mathbf{Y}_{il} + \alpha_3 \frac{1}{m} \sum_{k=1}^{m} \mathbf{Y}_{kj} + \alpha_4 \frac{1}{mq} \sum_{k=1}^{m} \sum_{l=1}^{q} \mathbf{Y}_{kl},$$

with $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{R}^4$.

Proof. For setting A, the hat matrix of equation 1.3 transforms the label matrix in the prediction matrix. The hat matrix $\mathbf{H}^{\text{SM}}$ for Kronecker KRR using these smoother kernels can be obtained by applying equation 2.10:

$$\mathbf{H}^{\text{SM}} = a_1 \mathbb{I}_q \otimes \mathbb{I}_m + a_2 \mathbb{J}_q \otimes \mathbb{I}_m + a_3 \mathbb{I}_q \otimes \mathbb{J}_m + a_4 \mathbb{J}_q \otimes \mathbb{J}_m. \tag{3.7}$$

This can easily be seen because the pairwise Gram matrix will be of the form 3.7 and properties 3 and 4 state that multiplying or inverting matrices

of the form 3.7 results in a matrix of the same form. From these properties, it follows that the hat matrix will also be of this form.

The prediction for dyad (u_i, v_j) is given by $[\mathbf{H}^{\mathrm{SM}}\mathrm{vec}(\mathbf{Y})]_{jq+i}$. Using the relation between the Kronecker product and the vectorization operation, each term of equation 3.7 can be rewritten using property 1 as follows:

$$a_1[(\mathbb{I}_q \otimes \mathbb{I}_m)\mathrm{vec}(\mathbf{Y})]_{jm+i} = a_1[\mathbb{I}_m \mathbf{Y} \mathbb{I}_q]_{ij} = a_1 \mathbf{Y}_{ij},$$

$$a_2[(\mathbb{J}_{q\times q} \otimes \mathbb{I}_m)\mathrm{vec}(\mathbf{Y})]_{jm+i} = a_2[\mathbb{I}_m \mathbf{Y} \mathbb{J}_{q\times q}]_{ij} = a_2 \sum_{l=1}^{q} \mathbf{Y}_{il},$$

$$a_3[(\mathbb{I}_q \otimes \mathbb{J}_{m\times m})\mathrm{vec}(\mathbf{Y})]_{jm+i} = a_3[\mathbb{J}_{m\times m} \mathbf{Y} \mathbb{I}_q]_{ij} = a_3 \sum_{k=1}^{m} \mathbf{Y}_{kj},$$

$$a_4[(\mathbb{J}_{q\times q} \otimes \mathbb{J}_{m\times m})\mathrm{vec}(\mathbf{Y})]_{jm+i} = a_4[\mathbb{J}_{m\times m} \mathbf{Y} \mathbb{J}_{q\times q}]_{ij} = a_4 \sum_{l=1}^{q} \sum_{k=1}^{m} \mathbf{Y}_{kl},$$

which proves the theorem. □

The smoother kernel is thus quite restrictive in the type of models that can be learned. It can only exploit the fact that some rows or columns have a larger average value (e.g., in item recommendation, some items in collaborative filtering have a high average rating, independent for the user). Nevertheless, it can lead to good baseline predictions for setting A and is particularly useful for small data sets with no side features, such as species interaction networks.

3.3 Universality of the Kronecker Product Pairwise Kernel. Here we consider the universal approximation properties of Kronecker KRR and, by theorems 2 and 3, of two-step KRR. This is a necessary step in showing the consistency of the latter method. First, recall the concept of universal kernel functions.

Definition 2 *(Steinwart, 2002). A continuous kernel $k(\cdot, \cdot)$ 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 $k(\cdot, \cdot)$ 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 a universal kernel can approximate any continuous function on the input space $\mathcal{X}$ 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 this approximation from the hypothesis space (Steinwart, 2002). In other words, the learning algorithm is consistent in the sense that, informally put, the hypothesis learned by it gets closer to the function to be

learned while the size of the training set gets larger. The consistency properties of two-step KRR are considered in more detail in section 3.4.

Next, we consider the universality of the Kronecker product pairwise kernel. The following result is a straightforward modification of some of the existing results in the literature (e.g., Waegeman et al., 2012), but it is presented here for self-containedness. This theorem is mainly related to setting D but also covers the other settings as special cases.

Theorem 5. *The kernel* $\Gamma^{KK}((\cdot,\cdot),(\cdot,\cdot))$ *on* $\mathcal{U} \times \mathcal{V}$ *defined in equation 2.6 is universal if the instance kernel* $k(\cdot,\cdot)$ *on* $\mathcal{U}$ *and the task kernel* $g(\cdot,\cdot)$ *on* $\mathcal{V}$ *are both universal.*

Proof. Let us define

$$\mathcal{A} \otimes \mathcal{B} = \left\{ t \mid t(u,v) = a(u)b(v), a \in \mathcal{A}, b \in \mathcal{B} \right\} \tag{3.8}$$

for compact metric spaces $\mathcal{U}$ and $\mathcal{V}$ and sets of functions $\mathcal{A} \subset C(\mathcal{U})$ and $\mathcal{B} \subset C(\mathcal{V})$. We observe that the RKHS of kernel Γ can be written as $\mathcal{H}(k) \otimes \mathcal{H}(g)$, where $\mathcal{H}(k)$ and $\mathcal{H}(g)$ are the RKHS of kernels $k(\cdot,\cdot)$ and $g(\cdot,\cdot)$, respectively.

Let $\epsilon > 0$, and let $t \in C(\mathcal{U}) \otimes C(\mathcal{V})$ be an arbitrary function that, according to equation 3.8, can be written as $t(u,v) = a(u)b(v)$, where $a \in C(\mathcal{U})$ and $b \in C(\mathcal{V})$. By definition of the universality property, $\mathcal{H}(k)$ and $\mathcal{H}(g)$ are dense in $C(\mathcal{U})$ and $C(\mathcal{V})$, respectively. Therefore, there exist functions $\bar{a} \in \mathcal{H}(k)$ and $\bar{b} \in \mathcal{H}(g)$ such that

$$\max_{u \in \mathcal{U}} \left| \bar{a}(u) - a(u) \right| \leq \bar{\epsilon}, \quad \max_{v \in \mathcal{V}} \left| \bar{b}(v) - b(v) \right| \leq \bar{\epsilon},$$

where $\bar{\epsilon}$ is a constant for which it holds that

$$\max_{u \in \mathcal{U}, v \in \mathcal{V}} \left\{ \left| \bar{\epsilon}\, a(u) \right| + \left| \bar{\epsilon}\, b(v) \right| + \bar{\epsilon}^2 \right\} \leq \epsilon.$$

Note that according to the extreme value theorem, the maximum exists due to the compactness of $\mathcal{U}$ and $\mathcal{V}$ and the continuity of the functions $a(\cdot)$ and $b(\cdot)$. Now we have

$$\begin{aligned}
&\max_{u \in \mathcal{U}, v \in \mathcal{V}} \left| t(u,v) - \bar{a}(u)\bar{b}(v) \right| \\
&\leq \max_{u \in \mathcal{U}, v \in \mathcal{V}} \left| t(u,v) - a(u)b(v) \right| + \left| \bar{\epsilon}\, a(u) \right| + \left| \bar{\epsilon}\, b(v) \right| + \bar{\epsilon}^2 \\
&= \max_{u \in \mathcal{U}, v \in \mathcal{V}} \left| \bar{\epsilon}\, a(u) \right| + \left| \bar{\epsilon}\, b(v) \right| + \bar{\epsilon}^2 \leq \epsilon,
\end{aligned}$$

which confirms the density of $\mathcal{H}(k) \otimes \mathcal{H}(g)$ in $C(\mathcal{U}) \otimes C(\mathcal{V})$.

The space $\mathcal{U} \times \mathcal{V}$ is compact if both $\mathcal{U}$ and $\mathcal{V}$ are compact according to Tikhonov's theorem. It is straightforward to see that $C(\mathcal{U}) \otimes C(\mathcal{V})$ is a subalgebra of $C(\mathcal{U} \times \mathcal{V})$; separates points in $\mathcal{U} \times \mathcal{V}$, vanishes at no point of

$C(\mathcal{U} \times \mathcal{V})$, and is therefore dense in $C(\mathcal{U} \times \mathcal{V})$ due to the Stone-Weierstraß theorem. Thus, $\mathcal{H}(k) \otimes \mathcal{H}(g)$ is also dense in $C(\mathcal{U} \times \mathcal{V})$, and Γ is a universal kernel on $\mathcal{U} \times \mathcal{V}$. □

3.4 Spectral Interpretation and Consistency. In this section we study the difference between independent task, Kronecker, and two-step KRR from the point of view of spectral regularization. The universal approximation properties of this kernel, already shown, are also connected to the consistency properties of two-step KRR, as we elaborate in more detail in this section.

Learning by spectral regularization originates from the theory of ill-posed problems. This paradigm is well studied in domains such as image analysis (Bertero & Boccacci, 1998) and, more recently, machine learning (e.g., Lo Gerfo, Rosasco, Odone, De Vito, & Verri, 2008). Here, one wants to find the parameters $\boldsymbol{\alpha}$ of the data-generating process given a set of noisy measurements $\mathbf{y}$ such that

$$\boldsymbol{\Gamma}\boldsymbol{\alpha} \approx \mathbf{y}, \tag{3.9}$$

with $\boldsymbol{\Gamma}$ a Gram matrix with eigenvalue decomposition $\boldsymbol{\Gamma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^\top$. At first glance, one can find the parameters $\boldsymbol{\alpha}$ by inverting $\boldsymbol{\Gamma}$:

$$\begin{aligned}\boldsymbol{\alpha} &= \boldsymbol{\Gamma}^{-1}\mathbf{y} \\ &= \mathbf{W}\boldsymbol{\Lambda}^{-1}\mathbf{W}^\top\mathbf{y}.\end{aligned}$$

If $\boldsymbol{\Gamma}$ has small eigenvalues, the inverse becomes highly unstable: small changes in the feature description of the label vector will lead to huge changes in $\boldsymbol{\alpha}$. Spectral regularization deals with this problem by generalizing the inverse by a so-called filter function to make solving equation 3.9 well posed. The following definition of a spectral filter–based regularizer is standard in the machine learning literature (see, e.g., Lo Gerfo et al., 2008). Note that we assume $\Gamma((\cdot,\cdot),(\cdot,\cdot))$ being bounded with $\kappa > 0$ such that $\sup_{\mathbf{x}\in\mathcal{X}} \sqrt{\Gamma(\mathbf{x},\mathbf{x})} \leq \kappa$, ensuring that the eigenvalues of the Gram matrix $\boldsymbol{\Gamma}$ are in $[0, \kappa^2]$.

Definition 3 *(admissible regularizer). A function $\varphi_\lambda : [0,\kappa^2] \to \mathbb{R}$, parameterized by $0 < \lambda \leq \kappa^2$, is an admissible regularizer if there exist constants $D, B, \gamma \in \mathbb{R}$, and $\bar{\nu}, \gamma_\nu > 0$ such that*

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

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

where the constant γ_ν does not depend on λ.

In the literature, the constant $\bar{\nu}$ is called the qualification of the regularizer, and it is related to the consistency properties of the learning method, as we will describe in more detail.

The spectral filter is a matrix function that acts as a stabilized generalization of a matrix inverse. Hence, equation 3.9 can be solved by

$$\begin{aligned}\boldsymbol{\alpha} &= \varphi_\lambda(\boldsymbol{\Gamma})\mathbf{y} \\ &= \mathbf{W}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top \mathrm{vec}(\mathbf{Y}).\end{aligned}$$

Similarly, the noisy measurements can be filtered to obtain a better estimation of the true labels:

$$\begin{aligned}\mathbf{f} &= \boldsymbol{\Gamma}\boldsymbol{\alpha} \\ &= \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^\top\mathbf{W}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top \mathrm{vec}(\mathbf{Y}) \\ &= \mathbf{W}\boldsymbol{\Lambda}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top \mathrm{vec}(\mathbf{Y}).\end{aligned}$$

The spectral interpretation allows for using a more general form of the hat matrix (see equation 2.10):

$$\mathbf{H}^{\boldsymbol{\Gamma}} = \mathbf{W}\boldsymbol{\Lambda}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top.$$

For example, the filter function corresponding to the Tikhonov regularization, as used for independent-task KRR, is given by

$$\varphi_\lambda^{\mathrm{TIK}}(\sigma) = \frac{1}{\sigma + \lambda},$$

with the ordinary least-squares approach corresponding to $\lambda = 0$. Several other learning approaches, such as spectral cutoff, iterated Tikhonov, and *L*2 boosting, can also be expressed as filter functions, but they cannot be expressed as a penalized empirical error minimization problem analogous to equation 2.4 (Lo Gerfo et al., 2008). The spectral interpretation can also be used to motivate novel learning algorithms.

Many authors have expanded this framework to multitask settings (e.g., Baldassarre et al., 2012; Argyriou, Micchelli, Pontil, & Ying, 2007; Argyriou, Micchelli, Pontil, & Massimiliano, 2010). We translate the pairwise learning methods from section 2 to this spectral regularization context. Let us denote the eigenvalue decomposition of the instance and task kernel matrices as

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^\top \quad \text{and} \quad \mathbf{G} = \mathbf{V}\mathbf{S}\mathbf{V}^\top.$$

Let $\mathbf{u}_i$ denote the ith eigenvector of $\mathbf{K}$ and $\mathbf{v}_j$ the jth eigenvector of $\mathbf{G}$. The eigenvalues of the kernel matrix obtained with the Kronecker product

pairwise kernel on a complete training set can be expressed as the Kronecker product $\mathbf{\Lambda} = \mathbf{S} \otimes \mathbf{\Sigma}$ of the eigenvalues $\mathbf{\Sigma}$ and $\mathbf{S}$ of the instance and task kernel matrices. For the models in this letter, it is opportune to define a pairwise filter function over the representation of the instances and tasks.

Both of the factor kernels are assumed to be bounded, and hence we can write that all the eigenvalues ς of the Kronecker product kernel can be factorized as the product of the eigenvalues of the instance and task kernels as follows:

$$\varsigma = \sigma s \quad \text{with } 0 \leq \sigma, s \leq a\sqrt{\varsigma} \text{ and } 1 \leq a < \infty, \tag{3.10}$$

where σ, s denote the eigenvalues of the factor kernels and a the constant determined as the product of $\sup_{u \in \mathcal{U}} \sqrt{k(u,u)}$ and $\sup_{v \in \mathcal{V}} \sqrt{g(v,v)}$.

Definition 4 *(pairwise spectral filter). We say that a function $\varphi_\lambda : [0, \kappa^2] \to \mathbb{R}$, parameterized by $0 < \lambda \leq \kappa^2$, is a pairwise spectral filter if it can be written as*

$$\varphi_\lambda(\varsigma) = \vartheta_\lambda(\sigma, s)$$

for some function $\vartheta_\lambda : [0, a\sqrt{\varsigma}]^2 \to \mathbb{R}$ with $1 \leq a < \infty$, and it is an admissible regularizer for all possible factorizations of the eigenvalues as in equation 3.10.

Since the eigenvalues of a Kronecker product of two matrices are just the scalar product of the eigenvalues of the matrices, the filter function for Kronecker KRR is given by

$$\vartheta_\lambda^{\mathrm{KK}}(s, \sigma) = \varphi_\lambda^{\mathrm{TIK}}(\sigma s) = \frac{1}{(\sigma s + \lambda)}, \tag{3.11}$$

where σ and s are the eigenvalues of $\mathbf{K}$ and $\mathbf{G}$, respectively. The admissibility of this filter is a well-known result, since it is simply the Tikhonov regularizer for the pairwise Kronecker product kernel.

Instead of considering two-step KRR from the kernel point of view, one can also cast it into the spectral filtering regularization framework. We start from equation 2.14, in vectorized form:

$$\begin{aligned}
\mathrm{vec}(\mathbf{A}) &= \left((\mathbf{G} + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V}\mathbf{S}\mathbf{V}^\top + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V}\varphi_{\lambda_v}^{\mathrm{TIK}}(\mathbf{S})\mathbf{V}^\top) \otimes (\mathbf{U}\varphi_{\lambda_u}^{\mathrm{TIK}}(\mathbf{\Sigma})\mathbf{U}^\top)\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V} \otimes \mathbf{U})(\varphi_{\lambda_v}^{\mathrm{TIK}}(\mathbf{S}) \otimes \varphi_{\lambda_u}^{\mathrm{TIK}}(\mathbf{\Sigma}))(\mathbf{V} \otimes \mathbf{U})^\top\right) \mathrm{vec}(\mathbf{Y}).
\end{aligned}$$

Hence, one can interpret two-step KRR with a complete training set for setting D as a spectral filtering regularization–based learning algorithm that

uses the pairwise Kronecker product kernel with the following filter function:

$$\begin{aligned}\vartheta_{\lambda}^{\mathrm{TS}}(s,\sigma) &= \varphi_{\lambda_v}^{\mathrm{TIK}}(s)\varphi_{\lambda_u}^{\mathrm{TIK}}(\sigma) \\ &= \frac{1}{(\sigma+\lambda_u)(s+\lambda_v)} \\ &= \frac{1}{\sigma s+\lambda_v\sigma+\lambda_u s+\lambda_v\lambda_u}. \end{aligned} \tag{3.12}$$

The validity of this filter is characterized by the following theorem:

Theorem 6. *The filter function $\vartheta_{\lambda}^{TS}(\cdot,\cdot)$ is admissible with $D = B = \gamma = 1$, $\gamma_\nu = 2ab$, and has qualification $\bar{\nu} = \frac{1}{2}$ for all factorizations of ς and λ as*

$$\varsigma = \sigma s \text{ and } \lambda = \lambda_v\lambda_u \quad \text{with } 0 \le \sigma, s \le a\sqrt{\varsigma} \text{ and } 0 < \lambda_v, \lambda_u \le b\sqrt{\lambda}, \tag{3.13}$$

where $1 \le a, b < \infty$ are constants that do not depend on λ or ς.

Proof. Let us recollect the last condition in definition 3:

$$\sup_{0<\varsigma\le\kappa^2} \frac{\varsigma^\nu}{\lambda^\nu}|1-\varsigma\varphi_\lambda(\varsigma)| \le \gamma_\nu, \quad \text{for any } \nu \in]0,\bar{\nu}],$$

where γ_ν does not depend on λ. In order to show this for all cases covered by equation 3.13, we rewrite the condition by taking the supremum with respect to the factorizations of ς and λ given the constants a and b:

$$\sup_{\substack{0<\varsigma\le\kappa^2 \\ 0<\lambda_v,\lambda_u\le b\sqrt{\lambda} \\ 0<\sigma,s\le a\sqrt{\varsigma}}} \frac{\varsigma^\nu}{\lambda^\nu}\left(1-\frac{\varsigma}{\varsigma+\lambda_v\sigma+\lambda_u s+\lambda}\right) \le \gamma_\nu, \quad \text{for any } \nu \in]0,\bar{\nu}].$$

The left-hand side then becomes

$$\sup_{0<\varsigma\le\kappa^2} \frac{\varsigma^\nu}{\lambda^\nu}\left(1-\frac{\varsigma}{\varsigma+2ab\sqrt{\lambda}\sqrt{\varsigma}+\lambda}\right) = \sup_{0<\varsigma\le\kappa^2}\left(\frac{2ab\lambda^{\frac{1}{2}-\nu}\varsigma^{\nu+\frac{1}{2}}+\lambda^{1-\nu}\varsigma^\nu}{\varsigma+2ab\sqrt{\lambda}\sqrt{\varsigma}+\lambda}\right).$$

By checking the extreme values of the latter expression with respect to $(\varsigma, \lambda, \nu)$ using standard differential calculus, we observe that it is bounded by $\gamma_\nu = 2ab$ if $\nu \in]0, \frac{1}{2}]$. With values of $\bar{\nu}$ larger than $\frac{1}{2}$, the term $2ab\lambda^{\frac{1}{2}-\nu}\varsigma^{\nu+\frac{1}{2}}$ in the numerator grows arbitrarily while $\lambda \to 0$, and hence the qualification

is $\bar{v} = \frac{1}{2}$. The other conditions in definition 3 can be verified by direct computation. □

Thus, equation 3.12 can be positioned within the spectral filtering regularization-based framework with separate regularization parameter values for instances and tasks. In contrast to equation 3.11, the filter of two-step KRR can be factorized into a component for the tasks and instances separately:

$$\vartheta_\lambda(s, \sigma) = \varphi_{\lambda_u}(\sigma)\varphi_{\lambda_v}(s). \tag{3.14}$$

Providing a different regularization for instances and tasks also makes sense from a learning point of view. It is easy to imagine a setting in which the instance has a much larger influence in determining the label compared to the task, or vice versa. For example, consider a collaborative filtering setting with the goal of recommending books for customers. Suppose that the sales of a book are for a very large part determined simply by being a best-seller novel or not, and less by an individual customer's taste. When building a predictive model, one would give more freedom to the part concerning the books (hence a lower regularization). Lesser degrees of freedom are given to the inference of the user's personal task, as this is harder to learn and explains less of the variance in the preferences. This can be extended even further by choosing specific filter functions separately for the instances and tasks tuned to the application at hand. In a pairwise setting, the filter function to perform independent-task KRR arises as a special case with $\lambda_v = 0$,

$$\vartheta^{\mathrm{IT}}_{\lambda_v}(s, \sigma) = \frac{1}{(\sigma + \lambda_u)s},$$

when the task kernel is full rank (see theorem 1).

Next, we analyze the consistency properties of two-step KRR in setting D, given the above results about the universality of the pairwise Kronecker product kernel and the spectral filtering interpretation of the method. Let $R(\cdot)$ denote the expected prediction error of a hypothesis f 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,

$$R(f) = \int_{\mathcal{X}\times\mathbb{R}} (f(\mathbf{x}) - y)^2 \mathrm{d}\rho(\mathbf{x}, y).$$

Given the input space $\mathcal{X}$, the minimizer of the error is the so-called regression function:

$$f_\rho(\mathbf{x}) = \int_{\mathbb{R}} y \, \mathrm{d}\rho(y \mid \mathbf{x}).$$

Following Baldassarre et al. (2012), Lo Gerfo et al. (2008), and Bauer, Pereverzev, and Rosasco (2007), we characterize the quality of a learning algorithm via its consistency properties, in particular, by considering whether the learning algorithm is consistent in the sense of definition 5:

Definition 5. *A learning algorithm is consistent if the following holds with high probability,*

$$\lim_{n\to\infty} \int_{\mathcal{X}} \left(\hat{f}_n^\lambda(\boldsymbol{x}) - f_\rho(\boldsymbol{x}) \right)^2 d\rho(\boldsymbol{x}) = 0,$$

where $\hat{f}_n^\lambda$ denotes the hypothesis inferred by the learning algorithm from a training set having n independent and identically drawn training examples.

The following result is assembled from the existing literature concerning spectral filtering-based regularization methods, and we present it here only in a rather abstract form. (For the details and further elaboration, we refer to Baldassarre et al., 2012; Lo Gerfo et al., 2008; and Bauer et al., 2007.)

Theorem 7. *If the filter function is admissible and the kernel function is universal, then the learning algorithm is consistent in the sense of definition 5. Furthermore, if the regularization parameter is set as $\lambda = \frac{1}{n^{2\upsilon+1}}$, where n denotes the number of independent and identically drawn training examples, the following holds with high probability:*

$$R(\hat{f}^\lambda) - R(f_\rho(\boldsymbol{x})) = \mathcal{O}\left(n^{-\frac{\upsilon}{2\upsilon+1}} \right). \tag{3.15}$$

Intuitively put, the universality of the kernel ensures that the regression function belongs to the hypothesis space of the learning algorithm, and the admissibility of the regularizer ensures that $R(\hat{f}^\lambda)$ converges to it when the size of the training set approaches infinity and the rate of convergence is reasonable.

Corollary 1. *Two-step KRR is consistent, and the hypothesis it infers from the training set of size $n = mq$ converges to the underlying regression function with a rate at least proportional to*

$$R(\hat{f}^\lambda) - R(f_\rho(u, v)) = \mathcal{O}\left(min(m, q)^{-\frac{\upsilon}{2\upsilon+1}} \right). \tag{3.16}$$

Proof. The result follows from the admissibility of the pairwise filter function, the universality of the pairwise Kronecker product kernel and the fact that the training set consists of at least $\min(m, q)$ independent and identically drawn training examples. □

Hence, it is proven that the two-step KRR is not only a universal method (can approximate any pairwise prediction function) but will also converge to the prediction function that generated the data when provided with enough training examples.

4 Related Work

In section 1, we argued that it remains important to study the theoretical properties of kernel methods for three reasons: (1) kernel methods are general-purpose instruments, (2) they often serve as building blocks for more complicated methods, and (3) they clearly outperform other methods for specific scenarios such as cross-validation. As such observations have been reported in other papers, including quantitative results on real-world data sets, we see no merit in providing additional experimental evidence. We refer to other work that pairwisely compares the kernel methods discussed in this letter with other machine learning methods (e.g., Ding, Takigawa, Mamitsuka, & Zhu, 2013; Romera-Paredes & Torr, 2015; Schrynemackers, Wehenkel, Babu, & Geurts, 2015; Stock, Poisot, Waegeman, & De Baets, 2017). However, it remains important to outline the commonalities and differences with other methods. In what follows, we subdivide these methods according to their applicability to settings A, B, C, or D.

4.1 Methods That Are Applicable to Setting A. In this section, we review methods for setting A—matrix completion methods. In section 2, such methods were claimed to be useful for a pairwise learning setting with partially observed matrices $\mathbf{Y}$. Both u and v are observed, but not for all instance-target combinations. In setting A, side information about instances or targets is not required per se. We hence distinguish between methods that ignore side information and methods that also exploit such information, in addition to analyzing the matrix $\mathbf{Y}$.

Inspired by the Netflix challenge in 2006, the former type of methods has been mainly popular in the area of recommender systems. Those methods often impute missing values by computing a low-rank approximation of the sparsely filled matrix $\mathbf{Y}$, and many variants exist in the literature, including algorithms based on nuclear norm minimization (Candes & Recht, 2008), gaussian processes (Lawrence & Urtasun, 2009), probabilistic methods (Shan & Banerjee, 2010), spectral regularization (Mazumder et al., 2010), nonnegative matrix factorization (Gaujoux & Seoighe, 2010), graph-regularized nonnegative matrix factorization (Cai, He, Han, & Huang, 2011), and alternating least-squares minimization (Jain, Netrapalli, & Sanghavi, 2013). In addition to recommender systems, matrix factorization methods are commonly applied to social network analysis (Menon & Elkan, 2010), biological network inference (Gönen, 2012; Liu, Sun, Guan, Zheng, & Zhou, 2015), and travel time estimation in car navigation systems (Dembczyński, Kotłowski, Gawel, Szarecki, & Jaszkiewicz, 2013).

In addition to matrix factorization, a few other methods exist for setting A. Historically, memory-based collaborative filtering has been popular, and corresponding methods are very easy to implement. They make predictions for the unknown cells of the matrix by modeling a similarity measure between either rows or columns (see, e.g., Takács, Pilászy, Németh, & Tikk, 2008). For example, when rows and columns correspond to users and items, respectively, one can predict novel items for a particular user by searching for other users with similar interests. To this end, different similarity measures are commonly used, including cosine similarity, Tanimoto similarity, and statistical similarity measures (Basnou, Vicente, Espelta, & Pino, 2015).

Many variants of matrix factorization and other collaborative methods have been presented in which side information of rows and columns is considered during learning, in addition to exploiting the structure of the matrix $\mathbf{Y}$ (see, e.g., Basilico and Hofmann, 2004; Abernethy, Bach, Evgeniou, & Vert, 2008; Adams, Dahl, & Murray, 2010; Fang & Si, 2011; Zhou, Chen, & Ye, 2011; Menon & Elkan, 2011; Zhou, Shan, Banerjee, & Sapiro, 2012; Gönen, 2012; Liu & Yang, 2015; Ezzat, Zhao, Wu, Li, & Kwoh, 2017). One simple but effective method is to extract latent feature representations for instances and targets in a first step, and combine those latent features with explicit features in a second step (Volkovs & Zemel, 2012). To this end, the methods that have been described in this letter could be used, as well as other pairwise learning methods that depart from explicit feature representations.

4.2 Methods That Are Applicable to Settings B and C. When side information is available for the objects u and v, it would be pointless to ignore this information. The hybrid filtering methods from the previous paragraph seek to combine the best of both worlds by simultaneously modeling side information and the structure of $\mathbf{Y}$. In addition to setting A, they can often be applied to settings B and C, which coincide, respectively, with a novel user and a novel item in recommender systems. In that context, one often speaks of cold-start recommendations.

However, when focusing on settings B and C only, a large bunch of machine learning methods is closely connected to pairwise learning. In fact, many multitarget prediction problems can be interpreted as specific pairwise learning problems. All multitask learning problems, and multilabel classification and multivariate regression problems as special cases, can be seen as pairwise learning problems by calling u an instance and v a label (target/output/task). (We refer readers to Waegeman, Dembczynski, & Hüllermeier, 2018), for a recent review on connections between multitarget prediction problems and pairwise learning.

Multitask learning, multilabel classification, and multivariate regression are huge research fields, so it is beyond the scope of this letter to give an in-depth review of all methods developed in those fields. Moreover, not all multitarget prediction methods are relevant for the discussion we intend to provide. Roughly speaking, simple multitarget prediction methods

consider side information for only one type of object, let's say the objects u, which represent the instances. No side information is available for the targets, which could then be denoted v. Since no side information is available for the targets, simple multitarget prediction methods can be applied to only settings B and C. We note that u and v are interchangeable, so settings B and C are identical settings from a theoretical point of view.

The situation changes when side information in the form of relations or feature representations becomes available for both instances and targets. In such a scenario, multitarget prediction methods that process side information about targets are more closely related to the pairwise learning methods that are analyzed in this letter. We will therefore provide a thorough review of such methods in the next section. Furthermore, the availability of side information on both instance and target level implies that setting D can now be covered, in addition to settings B and C. Exploiting side information about targets has two main purposes: it might boost the predictive performance in settings B and C, and it is essential for generalizing to novel targets in setting D.

4.3 Methods That Are Applicable to Settings B, C, and D. In setting D, side information for both u and v is essential for generalizing to zero-shot problems, such as a novel target molecule in drug discovery, a novel tag in document tagging, or a novel person in image classification. In this area, kernel methods have played a prominent role in the past, but tree-based methods are also commonly used (Geurts, Touleimat, Dutreix, & D'Alché-Buc, 2007; Schrynemackers et al., 2015). In bioinformatics, a subdivision is usually made between global methods, which construct one predictive model, and local methods, which separate the problem into several subproblems (Vert, 2008; Bleakley & Yamanishi, 2009; Schrynemackers et al., 2013).

Factorization machines (Rendle, 2010; Steffen, 2012) deserve special mention here, as they can be seen as an extension of matrix factorization methods toward settings B, C, and D. They work by simultaneously learning a lower-dimensional feature embedding and a polynomial (usually of degree two) predictive model. Factorization machines can effectively cope with large, sparse data sets frequently encountered in collaborative and content-based filtering. For such problems, they are expected to outperform kernel methods. Their main drawback, however, is that training them becomes a nonconvex problem and requires relatively large data sets to train. The relation between factorization machines, polynomial networks, and kernel machines was recently explored by Blondel, Ishihata, Fujino, and Ueda (2016).

In recent years, specific zero-shot learning methods based on deep learning have become extremely popular in image classification applications. The central idea in all those methods is to construct semantic feature representations for class labels, for which various techniques might work. One

class of methods constructs binary vectors of visual attributes (Lampert, Nickisch, & Harmeling, 2009; Palatucci et al., 2009; Liu, Kuipers, & Savarese, 2011; Fu, Hospedales, Xiang, & Gong, 2013). Another class of methods rather considers continuous word vectors that describe the linguistic context of images (Mikolov, Chen, Corrado, & Dean, 2013; Frome et al., 2013; Socher et al., 2013).

Many zero-shot learning methods for image classification adopt principles that originate from kernel methods. The model structure can often be formalized as follows:

$$f(u, v) = \mathbf{w}^T \big(\phi(u) \otimes \psi(v)\big), \tag{4.1}$$

with $\mathbf{w}$ a parameter vector and ϕ an embedding of an object in a high-dimensional feature space. This model in fact coincides with the primal formulation of equation 2.3 with Γ the Kronecker product pairwise kernel. Different optimization problems with this model have been proposed (Frome et al., 2013; Akata, Reed, Walter, Lee, & Schiele, 2015; Akata, Perronnin, Harchaoui, & Schmid, 2016), and related methods provide nonlinear extensions (Socher et al., 2013; Xian et al., 2016). Most of these optimization problems do not minimize squared error loss, and they should rather be seen as structured output prediction methods. Indeed, a representation such as equation 4.1 is in fact commonly used in structured output prediction methods. These methods additionally have inference procedures that allow for finding the best-scoring targets in an efficient manner.

Some of the zero-shot learning methods from computer vision also turn out to be useful for the related field of text classification. For documents, it is natural to model a latent representation for both the (document) instances and class labels in a joint space (Nam, Loza Mencía, & Fürnkranz, 2016). Nonetheless, many of those approaches are tailor-made for particular application domains. In contrast to kernel methods, they do not provide general purpose tools for analyzing general data types.

5 Conclusion

In this work we have studied several models derived from kernel ridge regression. First, we independently derived single-task kernel ridge regression, Kronecker kernel ridge regression, and two-step kernel ridge regression and the linear filter. Subsequently, we have shown that they are all related; two-step kernel ridge regression and the linear filter are a special case of pairwise kernel ridge regression, itself being merely kernel ridge regression with a specific pairwise kernel. From this, universality and consistency results could be derived, motivating the general use of these methods.

Pairwise learning is a broadly applicable machine learning paradigm. It can be applied to problems as diverse as multitask learning, content and collaborative filtering, transfer learning, network inference, and zero-shot learning. This work offers a general tool kit to tackle such problems. Despite being easy to implement and computationally efficient, kernel methods have been found to attain an excellent performance on a wide variety of problems. We believe that the intriguing algebraic properties of the Kronecker product will serve as a basis for developing novel learning algorithms, and we hope that the results of this work will be helpful in that regard.

Appendix: Matrix Properties

The trick of pairwise learning is transforming a matrix in a vector. This can be done by the vectorization operation.

Definition 6 *(vectorization). The vectorization operator $vec(\cdot)$ is a linear operator that transforms an $n \times m$ matrix $\boldsymbol{A}$ in a column vector of length nm by stacking the columns of $\boldsymbol{A}$ on top of each other.*

Further, the Kronecker product is defined as follows:

Definition 7 *(Kronecker product). If $\boldsymbol{A} = [a_{ij}]$ is an $n \times m$ matrix and $\boldsymbol{B} = [ij]$ is an $p \times q$ matrix, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $mp \times nq$ block matrix:*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1m}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\boldsymbol{B} & \dots & a_{nm}\boldsymbol{B} \end{bmatrix}.$$

For instance, if

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},$$

then

$$\text{vec}(\mathbf{A}) = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}.$$

The relation between vectorization and the Kronecker product is given by the following property:

Property 1. For any conformable matrices $\mathbf{N}$, $\mathbf{M}$ and $\mathbf{X}$, it holds that

$$(\mathbf{N}^\top \otimes \mathbf{M})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{MXN}).$$

Computing the Kronecker product of two reasonably large matrices results in a huge matrix, often too large to fit in computer memory. If the Kronecker product is needed only in an intermediary step, the above identity can be used to dramatically reduce computation time and memory requirement.

Using the eigenvalue decomposition of matrices, a large system of equations using the Kronecker product can be solved efficiently.

Property 2 (Pahikkala et al., 2013). Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ be diagonalizable matrices, that is, matrices that can be eigendecomposed as

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \text{ and } \mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{-1},$$

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

$$(\mathbf{A} \otimes \mathbf{B} + \lambda\mathbb{I})\mathbf{a} = \text{vec}(\mathbf{Y}), \tag{A.1}$$

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

Proof. By multiplying both sides of equation A.1 by $(\mathbf{A} \otimes \mathbf{B} + \lambda\mathbb{I})^{-1}$, it is relatively straightforward to show that

$$\mathbf{a} = \text{vec}(\mathbf{V}(\mathbf{C} \odot \mathbf{E})(\mathbf{U}^\top)^{-1}), \tag{A.2}$$

with $\odot$ the Hadamard product (element-wise matrix multiplication),

$$\mathbf{E} = \mathbf{U}^{-1}\mathbf{Y}(\mathbf{V}^{-1})^\top$$

and

$$\text{diag}_m(\text{vec}(\mathbf{C})) = (\mathbf{\Lambda} \otimes \mathbf{\Sigma} + \lambda\mathbb{I})^{-1}.$$

The eigendecompositions of $\mathbf{A}$ and $\mathbf{B}$, as well as all matrix multiplications in equation A.2, can be computed in $\mathcal{O}(n^3)$ time. □

Finally, we present two matrix identities that are useful in deriving the linear filter of section 3.2. Consider two matrices of the form

$$\mathbf{A} = a_1\mathbb{I}_m \otimes \mathbb{I}_q + a_2\mathbb{J}_m \otimes \mathbb{I}_q + a_3\mathbb{I}_m \otimes \mathbb{J}_q + a_4\mathbb{J}_m \otimes \mathbb{J}_q$$

and

$$\mathbf{B} = b_1\mathbb{I}_m \otimes \mathbb{I}_q + b_2\mathbb{J}_m \otimes \mathbb{I}_q + b_3\mathbb{I}_m \otimes \mathbb{J}_q + b_4\mathbb{J}_m \otimes \mathbb{J}_q.$$

Two properties can easily be deduced.

Property 3. The product $\mathbf{C} = \mathbf{AB}$ is given by

$$\mathbf{C} = c_1\mathbb{I}_m \otimes \mathbb{I}_q + c_2\mathbb{J}_m \otimes \mathbb{I}_q + c_3\mathbb{I}_m \otimes \mathbb{J}_q + c_4\mathbb{J}_m \otimes \mathbb{J}_q,$$

with

$$\begin{aligned}
c_1 &= a_1b_1,\\
c_2 &= a_1b_2 + a_2b_1 + a_2b_2m,\\
c_3 &= a_1b_3 + a_3b_1 + a_3b_3q,\\
c_4 &= a_1b_4 + a_2b_3 + a_2b_4m + a_3b_2 + a_3b_4q + a_4b_1 + a_4b_2m + a_4b_3q + a_4b_4mq.
\end{aligned}$$

Property 4. The inverse $\mathbf{D} = \mathbf{A}^{-1}$ is given by

$$\mathbf{D} = d_1\mathbb{I}_m \otimes \mathbb{I}_q + d_2\mathbb{J}_m \otimes \mathbb{I}_q + d_3\mathbb{I}_m \otimes \mathbb{J}_q + d_4\mathbb{J}_m \otimes \mathbb{J}_q,$$

with

$$\begin{aligned}
d_1 &= \frac{1}{a_1},\\
d_2 &= \frac{-a_2}{a_1(a_1 + a_2m)},\\
d_3 &= \frac{-a_3}{a_1(a_1 + a_3q)},\\
d_4 &= (a_2(a_1 + a_3q)(a_3 + a_4m) + a_3(a_1 + a_2m)(a_2 + a_3q + a_4q)\\
&\quad - a_4(a_1 + a_2m)(a_1 + a_3q))(a_2(a_1 + a_2m)(a_1 + a_3q)(a_1 + a_2m\\
&\quad + a_3q + a_4mq))^{-1}.
\end{aligned}$$

Acknowledgments

M. S. is supported by the Research Foundation–Flanders (FWO17/PDO/067). This work was supported by the Academy of Finland (grants 311273 and 313266 to T.P. and grant 289903 to A.A.).

References

Abernethy, J., Bach, F., Evgeniou, T., & Vert, J.-P. (2008). A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, *10*, 803–826.

Adams, R. P., Dahl, G. E., & Murray, I. (2010). Incorporating side information into probabilistic matrix factorization using gaussian processes. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*. Corvallis, OR: AUAI Press.

Airola, A., & Pahikkala, T. (2018). Fast Kronecker product kernel methods via generalized vec trick. *IEEE Transactions on Neural Networks and Learning Systems* (forthcoming).

Akata, Z., Perronnin, F., Harchaoui, Z., & Schmid, C. (2016). Label-embedding for image classification. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *38*(7), 1425–1438.

Akata, Z., Reed, S. E., Walter, D., Lee, H., & Schiele, B. (2015). Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2927–2936). Piscataway, NJ: IEEE.

Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, *33*(8), 831–838.

Álvarez, M., Rosasco, L., & Lawrence, N. (2012). Kernels for vector-valued functions: A review. *Foundation and Trends in Machine Learning*, *4*(3), 195–266.

Argyriou, A., Micchelli, C. A., Pontil, M., & Massimiliano, Y. (2010). On spectral learning. *Journal of Machine Learning Research*, *11*, 935–953.

Argyriou, A., Micchelli, C. A., Pontil, M., & Ying, Y. (2007). A spectral regularization framework for multi-task structure learning. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems, 20* (pp. 25–32). Cambridge, MA: MIT Press.

Baldassarre, L., Rosasco, L., Barla, A., & Verri, A. (2012). Multi-output learning via spectral filtering. *Machine Learning*, *87*(3), 259–301.

Basilico, J., & Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning* (pp. 9–16). New York: ACM.

Basnou, C., Vicente, P., Espelta, J. M., & Pino, J. (2015). A network approach for inferring species associations from co-occurrence data. *Ecography*, *39*(12), 1139–1150.

Bauer, F., Pereverzev, S., & Rosasco, L. (2007). On regularization algorithms in learning theory. *Journal of Complexity*, *23*(1), 52–72.

Ben-Hur, A., & Noble, W. S. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, *21*(Suppl. 1), i38–i46.

Bertero, M., & Boccacci, P. (1998). *Introduction to inverse problems in imaging*. Boca Raton, FL: CRC Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer-Verlag.

Bleakley, K., & Yamanishi, Y. (2009). Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics*, *25*(18), 2397–2403.

Blondel, M., Ishihata, M., Fujino, A., & Ueda, N. (2016). Polynomial networks and factorization machines: New insights and efficient training algorithms. In *Proceedings of the 33th International Conference on Machine Learning*.

Bollen, K. A. (1996). An alternative two stage least squares (2SLS) estimator for latent variable equations. *Psychometrika*, *61*(1), 109–121.

Bollen, K. A., & Bauer, D. J. (2004). Automating the selection of model-implied instrumental variables. *Sociological Methods and Research*, *32*(4), 425–452.

Bonilla, E. V., Agakov, F., & Williams, C. (2007). Kernel multi-task learning using task-specific features. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics* (pp. 43–50).

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

Cai, D., He, X., Han, J., & Huang, T. S. (2011). Generalized graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(8), 1548–1560.

Candes, E., & Recht, B. (2008). Exact low-rank matrix completion via convex optimization. *Foundations of Computational Mathematics*, *9*, 717–772.

Cao, D.-S., Liu, S., Xu, Q.-S., Lu, H.-M., Huang, J.-H., Hu, Q.-N., & Liang, Y.-Z. (2012). Large-scale prediction of drug-target interactions using protein sequences and drug topological structures. *Analytica Chimica Acta*, *752*, 1–10.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Dembczyński, K., Kotłowski, W., Gawel, P., Szarecki, A., & Jaszkiewicz, A. (2013). Matrix factorization for travel time estimation in large traffic networks. In *Lecture Notes in Computer Science: Vol. 7895. Proceedings of the Artificial Intelligence and Soft Computing 12th International Conference* (pp. 500–510). Berlin: Springer-Verlag.

Ding, H., Takigawa, I., Mamitsuka, H., & Zhu, S. (2013). Similarity-based machine learning methods for predicting drug-target interactions: A brief review. *Briefings in Bioinformatics*, *14*(5), 734–747.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems* (vol. 2, pp. 2224–2232).

Elkan, C., & Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 213–220). New York: ACM.

Ezzat, A., Zhao, P., Wu, M., Li, X. L., & Kwoh, C. K. (2017). Drug-target interaction prediction with graph regularized matrix factorization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *14*(3), 646–656.

Fang, Y., & Si, L. (2011). Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems* (pp. 65–69). New York: ACM.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., & Mikolov, T. (2013). Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 2121–2129). Red Hook, NY: Curran.

Fu, Y., Hospedales, T., Xiang, T., & Gong, S. (2013). Learning multimodal latent attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(2), 303–316.

Gaujoux, R., & Seoighe, C. (2010). A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, *11*, 367.

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

Gittens, A., & Mahoney, M. W. (2013). Revisiting the Nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, *28*(3), 567–575.

Gönen, M. (2012). Predicting drug-target interactions from chemical and genomic kernels using Bayesian matrix factorization. *Bioinformatics*, *28*(18), 2304–2310.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.

Hayashi, K., Takenouchi, T., Tomioka, R., & Kashima, H. (2012). Self-measuring similarity for multi-task gaussian process. *Transactions of the Japanese Society for Artificial Intelligence*, *27*, 103–110.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., & Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, *5*(9), 1–10.

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273.

Jain, P., Netrapalli, P., & Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (pp. 665–674). New York: ACM.

Jo, T., Hou, J., Eickholt, J., & Cheng, J. (2015). Improving protein fold recognition by deep learning networks. *Scientific Reports*, *5*(1), 17573.

Johnson, R., & Zhang, T. (2008). Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, *54*(1), 275–288.

Jordano, P. (2016). Sampling networks of ecological interactions. *Functional Ecology*, *30*(12), 1883–1893.

Jung, S. (2013). Structural equation modeling with small sample sizes using two-stage ridge least-squares estimation. *Behavior Research Methods*, *45*(1), 75–81.

Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., & Tsuda, K. (2009). Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SIAM International Conference on Data Mining* (pp. 1099–1110). Philadelphia: SIAM.

Kashima, H., Oyama, S., Yamanishi, Y., & Tsuda, K. (2010). Cartesian kernel: An efficient alternative to the pairwise kernel. *IEICE Transactions on Information and Systems*, *93*(10), 2672–2679.

Lampert, C. H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by betweenclass attribute transfer. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Washington, DC: IEEE Computer Society.

Lampert, C. H., Nickisch, H., & Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(3), 453–465.

Lawrence, N., & Urtasun, R. (2009). Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*. New York: ACM.

Liu, H., Sun, J., Guan, J., Zheng, J., & Zhou, S. (2015). Improving compound-protein interaction prediction by building up highly credible negative samples. *Bioinformatics*, *31*(12), i221–i229.

Liu, H., & Yang, Y. (2015). Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of the 32nd International Conference on Machine Learning* (vol. 37, pp. 1880–1888). New York: ACM.

Liu, J., Kuipers, B., & Savarese, S. (2011). Recognizing human actions by attributes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3337–3344). Washington, DC: IEEE Computer Society.

Lo Gerfo, L., Rosasco, L., Odone, F., De Vito, E., & Verri, A. (2008). Spectral algorithms for supervised learning. *Neural Computation*, *20*(7), 1873–1897.

Mahoney, M. M. W. (2011). Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, *3*(2), 123–224.

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

Mazumder, R., Hastie, T., & Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, *11*, 2287–2322.

Menon, A., & Elkan, C. (2010). A log-linear model with latent features for dyadic prediction. In *Proceedings of the 10th IEEE International Conference on Data Mining* (pp. 364–373). Piscataway, NJ: IEEE.

Menon, A., & Elkan, C. (2011). Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery in Databases*, *6912*, 437–452.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. CoRR, abs/1301.3781.

Nam, J., Loza Mencía, E., & Fürnkranz, J. (2016). All-in text: Learning document, label, and word representations jointly. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 1948–1954). Palo Alto, CA: AAAI Press.

Oyama, S., & Manning, C. (2004). Using feature conjunctions across examples for learning pairwise classifiers. In *Lecture Notes in Computer Science: Vol. 3201. Proceedings of the European Conference on Machine learning and Knowledge Discovery in Databases* (pp. 322–333). Berlin: Springer-Verlag.

Pahikkala, T., Airola, A., Stock, M., De Baets, B., & Waegeman, W. (2013). Efficient regularized least-squares algorithms for conditional ranking on relational data. *Machine Learning*, *93*(2–3), 321–356.

Pahikkala, T., Stock, M., Airola, A., Aittokallio, T., De Baets, B., & Waegeman, W. (2014). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. In *Lecture Notes in Computer Science: Vol. 8725* (pp. 517–532). Berlin: Springer.

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

Palatucci, M., Hinton, G., Pomerleau, D., & Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems, 22* (pp. 1410–1418). Cambridge, MA: MIT Press.

Park, S.-T., & Chu, W. (2009). Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems* (pp. 21–28). New York: ACM.

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

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. European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 131–147). Berlin: Springer.

Rendle, S. (2010). Factorization machines. In *Proceedings of the IEEE International Conference on Data Mining* (pp. 995–1000). Piscataway, NJ: IEEE.

Rifkin, R., & Lippert, R. (2007). Value regularization and Fenchel duality. *Journal of Machine Learning Research*, *8*, 441–479.

Romera-Paredes, B., & Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *Proceedings of the 32nd International Conference on Machine Learning* (vol. 37, pp. 2152–2161). New York: ACM.

Schrynemackers, M., Küffner, R., & Geurts, P. (2013). On protocols and measures for the validation of supervised methods for the inference of biological networks. *Frontiers in Genetics*, *4*, 262.

Schrynemackers, M., Wehenkel, L., Babu, M. M., & Geurts, P. (2015). Classifying pairs with trees for supervised biological network inference. *Molecular Biosystems*, *11*(8), 2116–2125.

Shan, H., & Banerjee, A. (2010). Generalized probabilistic matrix factorizations for collaborative filtering. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, & X. Wu (Eds.), *The 10th IEEE International Conference on Data Mining* (pp. 1025–1030). Washington, DC: IEEE Computer Society.

Shen, J., Zhang, J., Luo, X., Zhu, W., Yu, K., Chen, K., . . . Jiang, H. (2007). Predicting protein-protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences of the United States of America*, *104*(11), 4337–4341.

Socher, R., Ganjoo, M., Sridhar, H., Bastani, O., Manning, C. D., & Ng, A. Y. (2013). *Zero-shot learning through cross-modal transfer*. CoRR, abs/1301.3666.

Steffen, R. (2012). Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, *3*(3), 1–22.

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

Stekhoven, D., & Bühlmann, P. (2012). MissForest-nonparametric missing value imputation for mixed-type data. *Bioinformatics*, *28*(2001), 1–7.

Stock, M. (2017). *Exact and efficient algorithms for pairwise learning*. Ph.D. diss., Ghent University.

Stock, M., De Baets, B., & Waegeman, W. (2017). An exact iterative algorithm for transductive pairwise prediction. In *Proceedings of the Twenty-Sixth Benelux Conference on Machine Learning* (pp. 98–101). Eindhoven: Technische Universiteit Eindhoven.

Stock, M., Pahikkala, T., Airola, A., Waegeman, W., & De Baets, B. (2018). *Algebraic shortcuts for leave-one-out cross-validation in supervised network inference*. Manuscript submitted for publication. https://doi.org/10.1101/24232.

Stock, M., Poisot, T., Waegeman, W., & De Baets, B. (2017). Linear filtering reveals false negatives in species interaction data. *Scientific Reports*, *7*(45908), 1–8.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, *2009*, art. 4.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008). Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (pp. 267–274), New York: ACM Press.

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.

Vert, J.-P. (2008). Reconstruction of biological networks by supervised machine learning approaches. In H. M. Lodhi & S. H. Muggleton (Eds.), *Elements of computational systems biology* (pp. 165–188). New York: Wiley.

Vert, J.-P., Qiu, J., & Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, *8*(S–10), 1–10.

Vert, J.-P., & Yamanishi, Y. (2005). Supervised graph inference. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, *17* (pp. 1433–1440). Cambridge, MA: MIT Press.

Volkovs, M., & Zemel, R. S. (2012). Collaborative ranking with 17 parameters. In F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 2303–2311). Red Hook, NY: Curran.

Waegeman, W., Dembczynski, K., & Hüllermeier, E. (2018). *Multi-target prediction: A unifying view on problems and methods*. Manuscript submited for publication.

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.

Wahba, G. (1990). *Spline models for observational data*. Philadelphia: SIAM.

Xian, Y., Akata, Z., Sharma, G., Nguyen, Q. N., Hein, M., & Schiele, B. (2016). Latent embeddings for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 69–77). Piscataway, NJ: IEEE.

Zachariah, D., & Sundin, M. (2012). Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Processing Letters*, *19*(4), 231–234.

Zaki, N., Lazarova-Molnar, S., El-Hajj, W., & Campbell, P. (2009). Protein-protein interaction based on pairwise similarity. *BMC Bioinformatics*, *10*(150), 1–12.

Zhou, J., Chen, J., & Ye, J. (2011). Clustered multi-task learning via alternating structure optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *24*. Red Hook, NY: Curran.

Zhou, T., Shan, H., Banerjee, A., & Sapiro, G. (2012). Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 12th SIAM International Conference on Data Mining* (pp. 403–414). Philadelphia: SIAM.

Received August 23, 2017; accepted February 16, 2018.