Machine Learning (2022) 111:543–573
https://doi.org/10.1007/s10994-021-06127-y

Generalized vec trick for fast learning of pairwise kernel models

Markus Viljanen[1] · **Antti Airola**[1] · **Tapio Pahikkala**[1]

Received: 2 September 2020 / Revised: 28 September 2021 / Accepted: 17 November 2021 /
Published online: 28 January 2022
© The Author(s) 2021

Abstract
Pairwise learning corresponds to the supervised learning setting where the goal is to make predictions for pairs of objects. Prominent applications include predicting drug-target or protein-protein interactions, or customer-product preferences. In this work, we present a comprehensive review of pairwise kernels, that have been proposed for incorporating prior knowledge about the relationship between the objects. Specifically, we consider the standard, symmetric and anti-symmetric Kronecker product kernels, metric-learning, Cartesian, ranking, as well as linear, polynomial and Gaussian kernels. Recently, a $O(nm + nq)$ time generalized vec trick algorithm, where n, m, and q denote the number of pairs, drugs and targets, was introduced for training kernel methods with the Kronecker product kernel. This was a significant improvement over previous $O(n^2)$ training methods, since in most real-world applications $m, q << n$. In this work we show how all the reviewed kernels can be expressed as sums of Kronecker products, allowing the use of generalized vec trick for speeding up their computation. In the experiments, we demonstrate how the introduced approach allows scaling pairwise kernels to much larger data sets than previously feasible, and provide an extensive comparison of the kernels on a number of biological interaction prediction tasks.

Keywords Kernel methods · Pairwise kernels · Pairwise learning · Interaction prediction

Editors: Jean-Philippe Vert.

✉ Antti Airola
ajairo@utu.fi

Markus Viljanen
majuvi@utu.fi

Tapio Pahikkala
aatapa@utu.fi

[1] Department of Computing, University of Turku, Turku, Finland

1 Introduction

The goal of supervised learning is to learn an unknown function $f : \mathcal{X} \rightarrow \mathbb{R}$ from a set of training examples $Z = \{(x_i, y_i)\}_{i=1}^n$ each consisting of an input $x_i \in \mathcal{X}$ and an associated label $y_i \in \mathbb{R}$. The learning algorithm returns a function that approximates the true function on the training set with the aim of generalizing to data unseen during the training phase.

In pairwise learning, each input x is viewed as a pair of objects $x = (d, t)$ that we call here drugs $d \in \mathcal{D}$ and targets $t \in \mathcal{T}$. The task may, for example, then be to predict drug and target interaction $y = f(d, t)$ values to test for novel interactions in drug discovery. This view is not unique and the inputs may be considered as paired in many different applications. For example, recommender system literature deals with ratings given to customer and product pairs (Basilico and Hofmann 2004; Menon and Elkan 2010; Rendle 2010). Information retrieval can be formulated as predicting the relevance of query and document pairs (Liu 2011). Bioinformatics has utilized machine learning for protein-protein (Ben-Hur and Noble 2005; Ruan et al. 2018), protein-RNA (Bellucci et al. 2011) and drug-target (Gönen 2012; Pahikkala et al. 2015a; Cichonska et al. 2017, 2018) interaction prediction. Other applications include image labeling (Romera-Paredes and Torr 2015), and link prediction in social networks (Pieter and Koller 2005) Various terminology and frameworks have been used to describe the general learning problem (see e.g. Waegeman et al. (2019) for overview). These include pairwise (kernel) learning (Ben-Hur and Noble 2005; Park and Chu 2009; Cichonska et al. 2017, 2018), dyadic prediction (Menon and Elkan 2010; Pahikkala et al. 2014; Schäfer and Hüllermeier 2015), pair-input prediction (Park and Marcotte 2012), graph inference (Vert et al. 2007), link prediction (Pieter and Koller 2005; Kashima et al. 2009a), relational learning (Pahikkala et al. 2010; Waegeman et al. 2012; Pahikkala et al. 2013), multi-task (Bonilla et al. 2007; Bernard et al. 2017) and as a special case zero-shot (Romera-Paredes and Torr 2015) learning.

Different fields often consider different but related pairwise prediction tasks. These tasks can be divided into settings where different methods are applicable and which have varying degrees of difficulty. For example, in recommender systems one often assumes that all customers and products belong to the training set and that there are some example interactions for each customer and each product (Basilico and Hofmann 2004). Predictions are needed for (customer, product)-pairs where the rating is missing. In this setting, methods based on factorizing the interaction matrix can be used, and no explicit features are required. However, in cold-start problems the task is to predict an interaction of a new customer and product pair, where we do not have any examples with the same customer or product in the training set. Basic factorization methods do not generalize to such settings, rather methods that make use of customer and product features are needed (sometimes called side information). In this work, we restrict our considerations to methods that can generalize to novel drugs and targets, rather than just imputing missing interactions between known ones.

Kernel methods are a standard method in supervised learning. They provide feature based generalization beyond training drugs and targets, and are used as a competitive method especially in the cold start setting. Kernel methods can be applied when the training data either have explicit feature vectors or implicit feature vectors are defined via positive semidefinite kernel functions. When drugs and targets have separate features or kernels, we can use pairwise kernels to define a kernel for the pair. One simple way to define a pairwise kernel, is to concatenate a feature vector for the drug and the target together, and apply a standard kernel such as polynomial or Gaussian on this feature vector. However, a

large variety of different kernels specifically defined for pairwise data have been introduced in previous literature, starting with the introduction of the standard (Ben-Hur and Noble 2005; Basilico and Hofmann 2004; Oyama and Manning 2004) and symmetric (Ben-Hur and Noble 2005) Kronecker product kernels.

In this work we present a comprehensive review on pairwise kernels, and establish a joint framework under which the most commonly used of them can be expressed as linear combinations of Kronecker products. In particular, we cover the following kernels:

- Linear kernel
- 2nd degreee polynomial kernel
- Gaussian kernel
- Kronecker product kernel (Ben-Hur and Noble 2005; Basilico and Hofmann 2004; Oyama and Manning 2004)
- symmetric Kronecker product kernel (Ben-Hur and Noble 2005)
- Anti-symmetric Kronecker product kernel (Pahikkala et al. 2010)
- Cartesian kernel (Kashima et al. 2009b)
- Metric learning pairwise kernel (Vert et al. 2007)
- Ranking kernel (Herbrich 2000; Waegeman et al. 2012)

In our framework, we represent the linear combinations corresponding to these kernels with specifically designed operator notation. The notation is not only mathematically elegant but, as we show below, enables the analysis of the kernel properties and assumptions, fast computation and easy implementation.

We start by introducing the two most fundamental assumptions. The first, what we call **pairwise data assumption**, is that both the drugs and targets tend to appear several times as parts of different inputs in an observed data set. For example, the same drug d_i may belong to two different examples (d_i, t_j) and (d_i, t_k). In particular, if n is the number of observed data and m and q denote the numbers of unique drugs and targets in the data, then $n >> m + q$. This observation can be used to develop methods with computational shortcuts tailored specifically for the pairwise learning task, as we will elaborate below in more detail.

The second, what we refer to as **non-linearity assumption**, states another property inherent in pairwise learning problems, which is that the functions to be learned are usually not linear combinations of functions depending only of d or t. The opposite case, where the function can be expressed as $f(d,t) = f_d(d) + f_t(t)$ for some f_d and f_t, would indicate that $f(d_1, t_1) > f(d_2, t_1) \Longrightarrow f(d_1, t_2) > f(d_2, t_2)$ for all drugs and targets, that is, if drug d_1 is more effective than drug d_2 against target t_1, then drug d_1 is also more effective than drug d_2 against target t_2. In other words, there would always be a single drug that would be the best choice for all targets (and vice versa). This is illustrated in Fig. 1 with a simple example, where the effect of drug on target is a function of the row and column number parities of both drugs and targets. The 'chessboard' is a true pairwise data set where an interaction exists between even drugs and odd targets, or vice versa, corresponding to a XOR function between their parities, that is unlearnable with linear methods according to the classical result by Minsky and Papert (1969). In contrast, the 'tablecloth' a linear function of interaction strengths of odd drugs and odd targets, which are therefore completely independent of each other.

The runtime and in many cases the memory use of kernel solvers grow at least quadratically with respect to the number of pairs, and hence the use of pairwise kernels becomes infeasible in cases where the number of pairs is large. Faster training algorithms that avoid

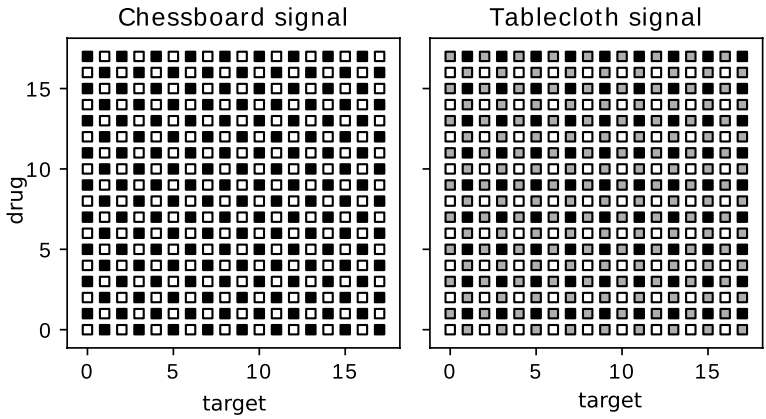

Fig. 1 Illustration of pairwise data. The 'chessboard' is a XOR-function of drug and target parities, whereas 'tablecloth' is a SUM-function of the parities

the costly step of building the pairwise kernel matrix have been previously proposed for certain specific cases. A fast solution to compute closed form solution is known for Kronecker product kernel, when minimizing ridge regression loss on so-called complete data that includes labels between all training drugs and targets (Romera-Paredes and Torr 2015; Pahikkala et al. 2014, 2013; Stock et al. 2018, 2020). Kashima et al. (2009a) show how computations for Cartesian kernel can be made faster, and computational shortcuts for speeding up the use of the ranking kernel are known for the ridge regression (Pahikkala et al. 2009) and support vector machine algorithms (Kuo et al. 2014). Yet thus far there has been no unified approach that would allow to plug in any of the commonly used pairwise kernels to a kernel method training algorithm that guarantees better than $O(n^2)$ scaling.

An exact computationally efficient algorithm has recently been proposed (Airola and Pahikkala 2018) for the special case of Kronecker product kernels when the data is not complete. The computational complexity of multiplying a vector with the kernel matrix is reduced from $O(n^2)$ to $O(nm + nq)$. This improvement has already shown to have major practical relevance, for example, the winning team of a recently held IDG-DREAM Drug-Kinase Binding Prediction Challenge on developing models for predicting unexplored drug-target potencies, used this algorithm (Cichonska et al. 2021).

In this work, we extend this result to present the first general $O(nm + nq)$ approach that simultaneously covers all the widely used pairwise kernels. This is a major improvement if the pairwise assumption holds, and the approach also allows accelerating the computation of the more traditional standard kernels, such as Gaussian and polynomial, if the data can be decomposed as pairwise. This is made possible by the proposed operator framework.

Finally, we perform an experimental comparison of different pairwise kernels on four different biological data sets, in which we compare the prediction performance, training time, number of training iterations and memory usage. The kernels are compared with each other in the following four different prediction tasks: First, prediction of interaction strength between a drug and a target of which both have been observed in the training data as part of some other drug-target pair with a known interaction strength. Second, interaction strength prediction with novel targets that have not been observed in the training data as part of any known drug-target pair. Third, prediction with novel drugs, and fourth, prediction with both novel drugs and targets. As is shown in by the results, the prediction

performances in these four tasks are tremendously different, hence underlying the importance that they should be separately considered in pairwise learning studies. Further, the results indicate that it is not at all self-evident that the expected prediction performance improvements of the nonlinear pairwise kernels over the linear ones implied by the nonlinearity assumption would always translate to practice.

To conclude, the major contributions of this work are as follows:

- Review of the standard kernels for pairwise data that establishes a common operator based framework for analysing and implementing the kernels.
- The framework allows accelerating the computation of matrix-vector products with the pairwise kernels to $O(nm + nq)$ time, leading to considerably faster training methods.
- Comprehensive experimental comparison of the pairwise kernels on biological interaction data sets with four different prediction problems.

2 Pairwise learning problem

Given the spaces of drugs $\mathcal{D}$ and targets $\mathcal{T}$, the possible drug and target pairs are the Cartesian product $\mathcal{X} = \mathcal{D} \times \mathcal{T}$. The label space is denoted $\mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{0, 1\}$ for classification. We further denote the joint space of the pairwise inputs and labels as $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The observed data set consists of n labeled drug-target pairs $Z_{\text{obs}} = (X_{\text{obs}}, \mathbf{y}) \in (\mathcal{D} \times \mathcal{T} \times \mathcal{Y})^n$. We further define $\mathbf{d} \in \mathcal{D}^n$ and $\mathbf{t} \in \mathcal{T}^n$ to be drug and target sequences such that $(d_i, t_i) = x_i$. Finally, we let $\mathcal{D}_{\text{obs}}$ and $\mathcal{T}_{\text{obs}}$ denote the sets of drugs and targets observed in the sample and $\mathcal{Z}_{\text{obs}}$ to denote the set of observed unique drug-target pairs, so that we have $m = |\mathcal{D}_{\text{obs}}|$ unique drugs and $q = |\mathcal{T}_{\text{obs}}|$ unique targets,

Our goal is to learn a prediction function $f : \mathcal{D} \times \mathcal{T} \rightarrow \mathcal{Y}$ from the training set, such that f can correctly predict the labels for a new pair $(d, t) \in \mathcal{D} \times \mathcal{T}$. The drug d and target t in the new pair may or may not belong to drugs $\mathcal{D}_{\text{obs}}$ and targets $\mathcal{T}_{\text{obs}}$ observed during training time. Here, four different settings emerge, as illustrated in Fig. 2:

1. $d \in \mathcal{D}_{\text{obs}}$ and $t \in \mathcal{T}_{\text{obs}}$: prediction for known drugs and targets
2. $d \in \mathcal{D}_{\text{obs}}$ and $t \notin \mathcal{T}_{\text{obs}}$: prediction for novel targets
3. $d \notin \mathcal{D}_{\text{obs}}$ and $t \in \mathcal{T}_{\text{obs}}$: prediction for novel drugs
4. $d \notin \mathcal{D}_{\text{obs}}$ and $t \notin \mathcal{T}_{\text{obs}}$: prediction for novel drugs and targets

In the literature specific settings in Fig. 2 have been sometimes considered separately. For example, Setting 1 can be solved even without drug or target features using matrix

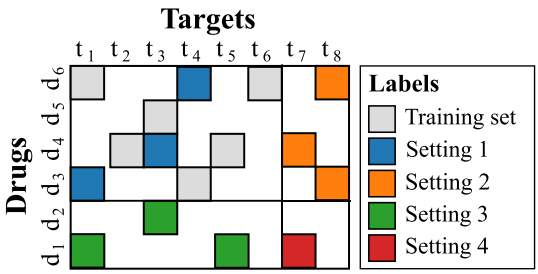

Fig. 2 Illustration of a pairwise data set with (drug,target)-pairs and sparse labels. Different types of test sets corresponding to different settings are illustrated with different colors

factorization methods (Basilico and Hofmann 2004). However, the latent representations learned by matrix factorization methods do not generalize to drugs and targets outside the training set (Settings 2-4). The pairwise kernel learning approach considered in this work is applicable in all of the four settings.

Recent studies have highlighted, that the prediction performance and optimal choice of kernel and hyperparameters for a pairwise learning method crucially depend on the assumption of how the test pairs overlap with training data (Park and Marcotte 2012; Pahikkala et al. 2015a; Stock et al. 2020). An experimental observation made over a large variety of different studies was that Setting 1 is usually the easiest to predict accurately, followed by Settings 2 and 3, whereas making accurate predictions in Setting 4 tends to be very challenging. As recommended in previous studies (Park and Marcotte 2012; Pahikkala et al. 2015a; Stock et al. 2020), we will always generate separate test sets for each of the four settings in the experiments to give a comprehensive view of how the learned prediction functions generalize to different types of test pairs. Depending on the amount of data, this can be implemented either with a single split to training and test sets, or by using cross-validation with repeated splits. The way the data splitting is implemented is defined in Table 1.

3 Learning algorithm

In this section we present a supervised machine learning approach for learning with pairwise kernels. The computational shortcuts presented in this paper can be used to speed up any optimization approach whose computational complexity is dominated by multiplications of a pairwise kernel matrix with a vector, such as the truncated Newton method (Airola and Pahikkala 2018). In this paper we focus on kernel ridge regression, as it is a widely used method that admits a closed form solution and simplifies the following considerations.

To learn a prediction function, we consider the regularized empirical risk minimization problem

Table 1 Training and test set split in different settings

Setting	Task	Validation method
Setting 1	$d \in \mathcal{D}_{\text{obs}}$ and $t \in \mathcal{T}_{\text{obs}}$	Split samples $\mathcal{Z}_{\text{obs}} = \mathcal{Z}_{\text{train}} \cup \mathcal{Z}_{\text{test}}$
Setting 2	$d \in \mathcal{D}_{\text{obs}}$ and $t \notin \mathcal{T}_{\text{obs}}$	Split targets $\mathcal{T}_{\text{obs}} = \mathcal{T}_{\text{train}} \cup \mathcal{T}_{\text{test}}$. Then $(x_i, y_i) \in \begin{cases} \mathcal{Z}_{\text{train}}, & \text{if } t_i \in \mathcal{T}_{\text{train}} \\ \mathcal{Z}_{\text{test}}, & \text{if } t_i \in \mathcal{T}_{\text{test}} \end{cases}$
Setting 3	$d \notin \mathcal{D}_{\text{obs}}$ and $t \in \mathcal{T}_{\text{obs}}$	Split drugs $\mathcal{D}_{\text{obs}} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ Then $(x_i, y_i) \in \begin{cases} \mathcal{Z}_{\text{train}}, & \text{if } d_i \in \mathcal{D}_{\text{train}} \\ \mathcal{Z}_{\text{test}}, & \text{if } d_i \in \mathcal{D}_{\text{test}} \end{cases}$
Setting 4	$d \notin \mathcal{D}_{\text{obs}}$ and $t \notin \mathcal{T}_{\text{obs}}$	Split both targets $\mathcal{T}_{\text{obs}} = \mathcal{T}_{\text{train}} \cup \mathcal{T}_{\text{test}}$ and drugs $\mathcal{D}_{\text{obs}} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ Then $(x_i, y_i) \in \begin{cases} \mathcal{Z}_{\text{train}}, & \text{if } t_i \in \mathcal{T}_{\text{train}} \text{ and } d_i \in \mathcal{D}_{\text{train}} \\ \mathcal{Z}_{\text{test}}, & \text{if } t_i \in \mathcal{T}_{\text{test}} \text{ and } d_i \in \mathcal{D}_{\text{test}} \\ \mathcal{Z}_{\text{ignored}}, & \text{otherwise} \end{cases}$

$$f = \text{argmin}_{f \in \mathcal{H}} \left\{ L(\mathbf{p}, \mathbf{y}) + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}} \right\}$$

where $\mathbf{p} \in \mathbb{R}^n$ are the predicted outputs and $\mathbf{y} \in \mathbb{R}^n$ the correct outputs, L a convex non-negative loss function and $\lambda > 0$ a regularization parameter.

To define a kernel learning problem, let $k_{\mathcal{D},\mathcal{T}} : (\mathcal{D} \times \mathcal{T}) \times (\mathcal{D} \times \mathcal{T}) \to \mathbb{R}$ be a positive semidefinite pairwise kernel function. Denote the kernel matrix containing the kernel evaluations between the drug-target pairs used to train the model as $\mathbf{K} \in \mathbb{R}^{n \times n}$ such that $\mathbf{K}_{i,j} = k_{\mathcal{D},\mathcal{T}}((d_i, t_i), (d_j, t_j))$. Choosing the reproducing kernel Hilbert space (RKHS) associated with $k_{\mathcal{D},\mathcal{T}}$ as the hypothesis space $\mathcal{H}$ for risk minimization, the representer theorem (Schölkopf et al. 2001) implies that the minimizing function can be written as:

$$f(d, t) = \sum_{i=1}^{n} a_i k_{\mathcal{D},\mathcal{T}}((d_i, t_i), (d, t))$$

where $\mathbf{a} \in \mathbb{R}^n$ is the vector of dual coefficients. Accordingly, the predictions for the training data can be written with the kernel matrix as $\mathbf{p} = \mathbf{K}\mathbf{a}$.

Kernel ridge regression (see e.g. (Poggio and Smale 2003)) is a special case of the regularized empirical risk minimization, where the loss function is the squared loss $L(\mathbf{p}, \mathbf{y}) = \|\mathbf{y} - \mathbf{p}\|^2$. The optimization problem then has a direct solution in terms of matrix algebra. The ridge regression problem can be formulated as solving the dual parameter vector $\mathbf{a} \in \mathbb{R}^n$:

$$\mathbf{a} = \text{argmin}_{\mathbf{a} \in \mathbb{R}^n} \|\mathbf{y} - \mathbf{K}\mathbf{a}\|^2 + \lambda \mathbf{a}^T \mathbf{K}\mathbf{a}$$

It can be shown that this corresponds to solving the linear equation:

$$(\mathbf{K} + \lambda \mathbf{I})\mathbf{a} = \mathbf{y} \tag{1}$$

Solving this system with a method that computes $\mathbf{K}$ requires at least $O(n^2)$ time and memory, which is not practical in many pairwise learning problems, where n can be in the range of 10^5 or more. A much more efficient solution can be found, when the kernel matrix can be expressed as a Kronecker product matrix. Assume we have a drug kernel function $k_{\mathcal{D}} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ and a target kernel function $k_{\mathcal{T}} : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$. The Kronecker product kernel is then defined as the product of the drug and target kernels $k_{\mathcal{D},\mathcal{T}}((d, t), (\bar{d}, \bar{t})) = k_{\mathcal{D}}(d, \bar{d}) k_{\mathcal{T}}(t, \bar{t})$.

In the following considerations, we also use the following linear operator notation for the kernels. For the drug kernel, $\mathbf{D} \in \mathbb{R}^{\mathcal{D} \times \mathcal{D}}$ such that $\mathbf{D}_{d,\bar{d}} = k_{\mathcal{D}}(d, \bar{d})$, and for the target kernel $\mathbf{T} \in \mathbb{R}^{\mathcal{T} \times \mathcal{T}}$ such that $\mathbf{T}_{t,\bar{t}} = k_{\mathcal{T}}(t, \bar{t})$. For finite domains of drugs and targets, the operators can be considered as matrices, whose rows and columns are indexed with drugs and targets instead of positive integers. Their addition, scalar multiplication, transpose and Kronecker product of these operators also naturally extend to infinite and continuous domains. For example, the operator corresponding to Kronecker product kernel over drug and target kernels is $\mathbf{D} \otimes \mathbf{T} \in \mathbb{R}^{(\mathcal{D} \times \mathcal{T}) \times (\mathcal{D} \times \mathcal{T})}$ so that $(\mathbf{D} \otimes \mathbf{T})_{(d,t),(\bar{d},\bar{t})} = \mathbf{D}_{d,\bar{d}} \mathbf{T}_{t,\bar{t}}$, and with the parenthesis notation we stress that both the rows and columns of the Kronecker product operator are indexed by drug-target pairs. Extending the matrix product is more involved in general but the products considered in this paper are always well-defined. This is enough for our purposes, and hence we avoid going into further technical details.

Let $\mathbf{R}(\mathbf{d},\mathbf{t}) \in \mathbb{R}^{n\times(\mathcal{D}\times\mathcal{T})}$ denote the Kronecker product indexing operator, whose rows are indexed by a sample of n drug-target pairs and columns by all drug-target pairs in the space $\mathcal{D}\times\mathcal{T}$. Its values, as a function of the sequences $\mathbf{d} \in \mathcal{D}^n$ and $\mathbf{t} \in \mathcal{T}^n$, are defined as follows:

$$\mathbf{R}(\mathbf{d},\mathbf{t})_{i,(d,t)} = \begin{cases} 1 & \text{if } (d,t) = (d_i, t_i) \\ 0 & \text{otherwise} \end{cases} .$$

Below, we omit explicitly writing $\mathbf{d}$ and $\mathbf{t}$ for clarity when they are clear from the context. In the literature, this type of constructs are sometimes called sampling operators, as they select a finite sample from a space of possibilities.

For two samples of data, say $X = (\mathbf{d},\mathbf{t})$ and $\overline{X} = (\overline{\mathbf{d}},\overline{\mathbf{t}})$, the kernel matrix containing all Kronecker product kernel evaluations between data in the first and second sample can then be expressed as $\mathbf{R}(\mathbf{d},\mathbf{t})(\mathbf{D}\otimes\mathbf{T})\mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}})^T$. The second sample can be, for example, a validation sets used for selecting an appropriate value of the regularization parameter, the number of training iterations, or kernel parameter values. It can also be used for prediction performance evaluation of the final model with a separate test set or in general for performing predictions for data with unknown labels.

Substituting the kernel matrix of evaluations between the training data with itself to (1), we end up to the following linear system:

$$(\mathbf{R}(\mathbf{D}\otimes\mathbf{T})\mathbf{R}^T + \lambda\mathbf{I})\mathbf{a} = \mathbf{y} \tag{2}$$

This linear system can be solved iteratively, for example, with the minimal residual method (Saad and Schultz 1986), combined with early stopping. A single training iteration in Equation 2 requires matrix vector products of the form $\mathbf{u} \leftarrow (\mathbf{R}(\mathbf{D}\otimes\mathbf{T})\mathbf{R}^T + \lambda\mathbf{I})\mathbf{u}$. Given a vector of parameters $\mathbf{u}$, predictions for another sample of data not used in training can be computed as a single matrix vector product $\mathbf{v} = \mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}})(\mathbf{D}\otimes\mathbf{T})\mathbf{R}(\mathbf{d},\mathbf{t})^T\mathbf{u}$, where $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^{\overline{n}}$.

Table 2 presents the relevant dimensions associated to the matrix vector products. We next recollect the following result (Airola and Pahikkala 2018) concerning matrix-vector products in which the matrix consists of a Kronecker product that is indexed from both left and right sides. This theorem is a generalization of Roth's column lemma (Roth 1934), often known as the "vec-trick".

Theore.m 1 (Airola and Pahikkala (2018)) *Let*

$$\begin{aligned} \mathbf{R}(\mathbf{d},\mathbf{t}) &\in \mathbb{R}^{n\times(\mathcal{D}\times\mathcal{T})} \\ \mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}}) &\in \mathbb{R}^{\overline{n}\times(\mathcal{D}\times\mathcal{T})} \\ \mathbf{a} &\in \mathbb{R}^{n} \\ \mathbf{p} &\in \mathbb{R}^{\overline{n}} \end{aligned}$$

Then, the operation

Table 2 Notation denoting the numbers of pairs, drugs and targets

n	The number of pairs in the first sample
m	The number of unique drugs in the first sample
q	The number of unique targets in the first sample
$\overline{n}$	The number of pairs in the second sample
$\overline{m}$	The number of unique drugs in the second sample
$\overline{q}$	The number of unique targets in the second sample

$$\mathbf{p} \leftarrow \mathbf{R}(\overline{\mathbf{d}}, \overline{\mathbf{t}})(\mathbf{D} \otimes \mathbf{T})\mathbf{R}(\mathbf{d}, \mathbf{t})^T \mathbf{a}$$

can be carried out in $O(\min(\overline{q}n + m\overline{n}, \overline{m}n + q\overline{n}))$ *time using a sparse Kronecker product multiplication algorithm known as the generalized vec-trick* (*GVT*).

The theorem implies that in training, the Kronecker product kernel matrix can be multiplied with a dual parameter vector in $O(qn + mn)$ time. The cost of computing predictions simultaneously for a set of data not used for training is $O(\min(\overline{q}n + m\overline{n}, \overline{m}n + q\overline{n}))$, where the overlined symbols denote the dimensions of the set for which the predictions are computed. This is much more efficient than the $O(n^2)$ or $O(n\overline{n})$ costs of explicitly forming the kernel matrices, since typically $m, q << n$ and $\overline{m}, \overline{q} << \overline{n}$.

4 Sum of Kronecker products framework for pairwise kernels

In this section we discuss different pairwise kernels presented in the literature and show how they can be expressed as sums of Kronecker products. Each matrix vector product can then be calculated as a sum of individual Knocker product terms. This allows the application of GVT shortcut to all of these kernels, which results in efficient algorithm for both training and making predictions.

Table 4 highlights an important limitation that applies to some of the kernels. These require homogeneous domains, i.e. they assume both objects in the pair belong to the same domain $\mathcal{D} = \mathcal{T}$, so that $x = (d, t) \in \mathcal{D} \times \mathcal{D}$. For the other kernels, we can have heterogeneous domains. Further, the Cartesian kernel is designed to be used in Setting 1 only, as it does not allow generalization to such drugs and targets that are not included in the training data.

The pairwise kernels can be motivated through feature mappings, since different pairwise kernel functions in Table 3 imply different implicit feature mappings for the pair as listed in Table 4. The implicit drug and target feature mappings $\phi_{\mathcal{D}} : \mathcal{D} \rightarrow \mathbb{R}^r$ and $\phi_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{R}^s$ are defined by the drug and target kernels

Table 3 Kernel functions of different pairwise kernels

Kernel	$k_{\mathcal{D},\mathcal{T}}((d,t),(\overline{d},\overline{t}))$ or $k_{\mathcal{D},\mathcal{D}}((d,d'),(\overline{d},\overline{d'}))$
Linear	$k_{\mathcal{D}}(d,\overline{d}) + k_{\mathcal{T}}(t,\overline{t})$
Poly2D	$(k_{\mathcal{D}}(d,\overline{d}) + k_{\mathcal{T}}(t,\overline{t}))^2$
Gaussian	$\exp(-\gamma \left\lVert \phi_{\mathcal{D}}(d) - \phi_{\mathcal{D}}(\overline{d}) \right\rVert) \exp(-\gamma \left\lVert \phi_{\mathcal{T}}(t) - \phi_{\mathcal{T}}(\overline{t}) \right\rVert)$
Kronecker	$k_{\mathcal{D}}(d,\overline{d})k_{\mathcal{T}}(t,\overline{t})$
Symmetric	$k_{\mathcal{D}}(d,\overline{d})k_{\mathcal{D}}(d',\overline{d'}) + k_{\mathcal{D}}(d,\overline{d'})k_{\mathcal{D}}(d',\overline{d})$
Anti-Symmetric	$k_{\mathcal{D}}(d,\overline{d})k_{\mathcal{D}}(d',\overline{d'}) - k_{\mathcal{D}}(d,\overline{d'})k_{\mathcal{D}}(d',\overline{d})$
Ranking	$k_{\mathcal{D}}(d,\overline{d}) - k_{\mathcal{D}}(d,\overline{d'}) - k_{\mathcal{D}}(d',\overline{d}) + k_{\mathcal{D}}(d',\overline{d'})$
MLPK	$(k_{\mathcal{D}}(d,\overline{d}) - k_{\mathcal{D}}(d,\overline{d'}) - k_{\mathcal{D}}(d',\overline{d}) + k_{\mathcal{D}}(d',\overline{d'}))^2$
Cartesian	$k_{\mathcal{D}}(d,\overline{d})\delta(t = \overline{t}) + \delta(d = \overline{d})k_{\mathcal{T}}(t,\overline{t})$

We show that all of these kernels can be expressed as a combination Kronecker products of a separate drug and target kernel

Table 4 Properties of different pairwise kernels

Kernel	$\mathcal{D} \neq \mathcal{T}$	$\phi_{\mathcal{D},\mathcal{T}}((d,t))$ or $\phi_{\mathcal{D},\mathcal{D}}((d,d'))$
Linear	X	$(\phi_{\mathcal{D}}(d), \phi_{\mathcal{T}}(t))$
Poly2D	X	$(\phi_{\mathcal{D}}(d), \phi_{\mathcal{T}}(t)) \otimes (\phi_{\mathcal{D}}(d), \phi_{\mathcal{T}}(t))$
Kronecker	X	$\phi_{\mathcal{D}}(d) \otimes \phi_{\mathcal{T}}(t)$
Symmetric		$\sqrt{1/2}(\phi_{\mathcal{D}}(d) \otimes \phi_{\mathcal{D}}(d') + \phi_{\mathcal{D}}(d') \otimes \phi_{\mathcal{D}}(d))$
Anti-Symmetric		$\sqrt{1/2}(\phi_{\mathcal{D}}(d) \otimes \phi_{\mathcal{D}}(d') - \phi_{\mathcal{D}}(d') \otimes \phi_{\mathcal{D}}(d))$
Ranking		$\phi_{\mathcal{D}}(d) - \phi_{\mathcal{D}}(d')$
MLPK		$(\phi_{\mathcal{D}}(d) - \phi_{\mathcal{D}}(d')) \otimes (\phi_{\mathcal{D}}(d) - \phi_{\mathcal{D}}(d'))$
Cartesian	X	$(\phi_{\mathcal{D}}(d) \otimes e_{\mathcal{T}}, e_{\mathcal{D}} \otimes \phi_{\mathcal{T}}(t))$

The middle column denotes whether the kernel allows heterogenous domains and the last column shows the feature map of the kernel

$k_{\mathcal{D}}(d,\bar{d}) = \langle \phi_{\mathcal{D}}(d), \phi_{\mathcal{D}}(\bar{d}) \rangle$ and $k_{\mathcal{T}}(t,\bar{t}) = \langle \phi_{\mathcal{T}}(t), \phi_{\mathcal{T}}(\bar{t}) \rangle$. Then the feature vector of the pair is defined by the feature mapping $\phi_{\mathcal{D},\mathcal{T}} : \mathcal{D} \times \mathcal{T} \rightarrow \mathbb{R}^p$ corresponding to the pairwise kernel $k_{\mathcal{D},\mathcal{T}}((d,t),(\bar{d},\bar{t})) = \langle \phi_{\mathcal{D},\mathcal{T}}(d,t), \phi_{\mathcal{D},\mathcal{T}}(\bar{d},\bar{t}) \rangle$. The claimed feature maps can be proven simply by computing the inner product and checking that it matches the definition of the kernel function. In the following, we discuss the implied pairwise feature vector $\phi_{\mathcal{D},\mathcal{T}}((d,t)) := (x_1^{d,t}, \ldots, x_p^{d,t}) \in \mathbb{R}^p$ of each pairwise kernel in terms of the drug $\phi_d(d) := (x_1^d, \ldots, x_r^d) \in \mathbb{R}^r$ and the target $\phi_t(t) := (x_1^t, \ldots, x_s^t) \in \mathbb{R}^s$ feature vectors. This motivates the kernels and demonstrates the intuition behind using a specific kernel for a specific task.

4.1 Linear

The pairwise linear kernel is computed as the linear kernel on the concatenated feature vector. The feature vector is the concatenation of the drug and target feature vectors $\mathbf{x}^{d,t} = (\mathbf{x}^d, \mathbf{x}^t)$. The resulting features consists of the union of original drug features $(x_i^d)_{i=1\ldots r}$ and target features $(x_i^t)_{i=1\ldots s}$. In this feature mapping, each feature contributes equally to interaction strength in every drug and target pair. Interaction is predicted simply by the presence or absence of certain features in the drug or the target, regardless of which drug and target pair is being tested. Given drug d and target t, the predicted interaction of the drug on the target is given by $f(d,t) = \langle \mathbf{w}^d, \mathbf{x}^d \rangle + \langle \mathbf{w}^t, \mathbf{x}^t \rangle$. This implies a global ordering of drugs, where drugs and targets are completely decoupled. If drug d_1 is more effective than drug d_2 against target t_1, then drug d_1 is also more effective than drug d_2 against target t_2: $f(d_1,t_1) > f(d_2,t_1) \Longrightarrow (d_1,t_2) > f(d_2,t_2)$. In the resulting model, some drugs and targets simply have more interactions than others, but there are no interactions between drug and target features. The artificial chessboard problem illustrated in Fig. 1 is an example of a data set, that is impossible to model using the pairwise linear kernel.

4.2 Polynomial

The pairwise polynomial kernel is computed as the polynomial kernel on the concatenated feature vector. On a second degree polynomial kernel without bias, the feature vector is the tensor product of the concatenated feature vector with itself $\mathbf{x}^{d,t} = (\mathbf{x}^d, \mathbf{x}^t) \otimes (\mathbf{x}^d, \mathbf{x}^t)$.

The resulting features include three types of terms: self interactions between drug features $(x_i^d x_j^d)_{i=1\ldots r,j=1\ldots r}$, pairwise interactions between drug and target features $(x_i^d x_j^t)_{i=1\ldots r,j=1\ldots s}$, and self interactions between target features $(x_i^t x_j^t)_{i=1\ldots s,j=1\ldots s}$. The self interactions contribute to a global ordering of drugs and targets, similar to the linear kernel. However, the pairwise interactions model actual interactions of drug and target features: a drug and target pair may be interacting if for example the features indicate that a certain chemical structure in a drug binds to a certain receptor on a target.

4.3 Gaussian

The pairwise Gaussian kernel is defined as the Gaussian kernel on the concatenated feature vector. This kernel $\exp(-\gamma\left\|(\mathbf{x}^d,\mathbf{x}^t)-(\overline{\mathbf{x}}^d,\overline{\mathbf{x}}^t)\right\|)=\exp(-\gamma\left\|\mathbf{x}^d-\overline{\mathbf{x}}^d\right\|)\exp(-\gamma\left\|\mathbf{x}^t-\overline{\mathbf{x}}^t\right\|)$ can be expressed as product of Gaussian drug and target kernels. This is a special case of the Kronecker product kernel, and will thus not be considered separately in the following.

4.4 Kronecker product

The Kronecker product kernel (Ben-Hur and Noble 2005; Basilico and Hofmann 2004; Oyama and Manning 2004) is computed as the product of drug and target kernels. The feature vector is given as a tensor product of the drug and target feature vectors $\mathbf{x}^{d,t}=\mathbf{x}^d\otimes\mathbf{x}^t$. The resulting feature vector consists of simply all the pairwise interactions $(x_i^d x_j^t)_{i=1\ldots r,j=1\ldots s}$. These are same as the pairwise interactions in the polynomial kernel with self-interations excluded. The Kronecker product kernel can be motivated as the simplest kernel that models actual pairwise interactions in drug and target features. The Kronecker kernel is an universal kernel, if the drug and target kernels are universal (e.g. Gaussian) (Waegeman et al. 2012).

4.5 Symmetric and anti-symmetric kernels

If we assume homogeneous domains, feature vectors can be written as a sum of symmetric and anti-symmetric parts $\phi_{\mathcal{D},\mathcal{D}}((d,d'))=1/2(\phi_{\mathcal{D},\mathcal{D}}((d,d'))+\phi_{\mathcal{D},\mathcal{D}}((d',d)))+1/2(\phi_{\mathcal{D},\mathcal{D}}((d,d'))-\phi_{\mathcal{D},\mathcal{D}}((d',d)))$. The symmetric Kronecker kernel (Ben-Hur and Noble 2005) is motivated by applying the symmetrization to the Kronecker kernel feature vector. This results in a tensor product of the drug and target feature vectors with only symmetric parts $\mathbf{x}^{d,d'}=1/2(\mathbf{x}^d\otimes\mathbf{x}^{d'}+\mathbf{x}^{d'}\otimes\mathbf{x}^d)$. The resulting features consist of all symmetric pairwise interactions $(1/2(x_i^d x_j^{d'}+x_i^{d'}x_j^d))_{i=1\ldots r,j=1\ldots r}$. When all interactions are known to be symmetric by definition, the symmetric Kronecker kernel is sometimes referred to as the Kronecker kernel in the literature. Several works have analysed the theoretical properties of the symmetric and antisymmetric Kronecker kernels (Pahikkala et al. 2010; Waegeman et al. 2012; Brunner et al. 2012; Pahikkala et al. 2015b; Gnecco 2017, 2018).

4.6 Ranking

The feature vector of the ranking kernel is the difference of drug and target feature vectors $\mathbf{x}^{d,d'}=\mathbf{x}^d-\mathbf{x}^{d'}$, which are assumed to belong to the same domain (Herbrich 2000; Waegeman et al. 2012). The resulting features consist of pairwise differences $(x_i^d-x_i^{d'})_{i=1\ldots r}$.

The ranking kernel can model ranking representable relations, i.e. relations constructed from some utility function h such that $f(d, d') = h(d) - h(d')$. For the ranking kernel $f(d, d') = \langle \mathbf{w}, \mathbf{x}^d \rangle - \langle \mathbf{w}, \mathbf{x}^{d'} \rangle$, which provides a global ranking of drugs based on their feature representation. The ranking kernel can be considered as an anti-symmetric linear kernel as can be observed from the operator notation below.

Pahikkala et al. (2009) show that the ranking kernel matrix can be computed using the oriented incidence operator $\mathbf{M} \in \mathbb{R}^{\mathcal{D} \times n}$ where

$$\mathbf{M}_{d,(d_i,d'_i)} = \begin{cases} 1 & \text{if } d_i = d \\ -1 & \text{if } d'_i = d \\ 0 & \text{otherwise} \end{cases} .$$

as $\mathbf{M}^{\mathrm{T}} \mathbf{D} \mathbf{M}$. Since $\mathbf{M}$ can be implemented with a sparse matrix, this allows efficient kernel matrix vector multiplication in $O(m^2 + n)$ time without need to use GVT.

4.7 MLPK

The MLPK kernel (Vert et al. 2007) is computed as the ranking kernel squared. The feature vector is given by the tensor product of the pairwise difference vector with itself $\mathbf{x}^{d,d'} = (\mathbf{x}^d - \mathbf{x}^{d'}) \otimes (\mathbf{x}^d - \mathbf{x}^{d'})$. The features consists of all pairwise interactions of pairwise differences $((x_i^d - x_i^{d'})(x_j^d - x_j^{d'}))_{i=1\ldots r, j=1\ldots r}$. This models interaction of a pair in the terms of how similar the drug and the target in the pair are. The formulation compares both elementwise differences, and possible interactions between the differences. The MLPK kernel can also be motivated as a distance learning problem by adding an extra parameter constraint to the standard SVM optimization problem (Vert et al. 2007). There, the goal is to learn a linear map such that the function is modelled by the Euclidean distance metric between feature vectors: learn a positive semidefinite matrix $\mathbf{M}$ such that $f(d, d') = (\mathbf{x}^d - \mathbf{x}^{d'})^T \mathbf{M} (\mathbf{x}^d - \mathbf{x}^{d'})$.

4.8 Cartesian

The Cartesian kernel (Kashima et al. 2009b) is computed as the drug kernel when the targets match, and the target kernel when the drugs match. The feature vector is given as a concatenation of the drug feature vector (target specific) and the target feature vector (drug specific) $\mathbf{x}^{d,t} = (\mathbf{x}^d \otimes e_t, e_d \otimes \mathbf{x}^t)$. The resulting features are sparse with nonzero terms $(x_i^d \delta(t = t_j))_{i=1\ldots r, j=1\ldots q}$ and $(\delta(d = d_i) x_j^t)_{i=1\ldots m, j=1\ldots s}$ corresponding to drug and target specific features. The full parameter vector $\mathbf{w}$ can be partitioned into drug specific $(\mathbf{w}^d)_{d \in \mathcal{D}}$ and target specific $(\mathbf{w}^t)_{t \in \mathcal{T}}$ parameters, with separate parameters learned for each drug and target. This means that target features may have different effects, depending on the drug, and vice versa. In this sense the learned model includes pairwise interactions, but it does not utilize information between similar interactions in different pairs and cannot generalize to drugs or targets that have not been seen in the training set. Kashima et al. (2009b) show that the Cartesian kernel can be represented as a Kronecker sum, and thus using the standard vec trick (Roth 1934) kernel matrix multiplication can be done in $O(m^2 q + q^2 m)$ time. In this work, we improve on this result.

4.9 Efficient computation of pairwise kernels

In this section we show how the pairwise kernel matrices of the above described kernels can be conveniently written as sums of Kronecker product matrices. For this purpose, we make the following definitions.

Definition 1 (Commutation and unification operators) The commutation operator $\mathbf{P} \in \mathbb{R}^{(\mathcal{T}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{T})}$ has its values defined as

$$\mathbf{P}_{(t,d),(\bar{d},\bar{t})} = \begin{cases} 1 & \text{if } d = \bar{d} \text{ and } t = \bar{t} \\ 0 & \text{otherwise} \end{cases} .$$

Note that if the domains $\mathcal{D}$ and $\mathcal{T}$ are different, the row indexing of any operator will be changed from $\mathcal{D}\times\mathcal{T}$ to $\mathcal{T}\times\mathcal{D}$ if multiplied from left with $\mathbf{P}$. Its inverse operator $\mathbf{P}^T \in \mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{T}\times\mathcal{D})}$ is defined analogously, by switching the drug and target domains. The values are also defined similarly when $\mathcal{D} = \mathcal{T}$ but in this case we use the notation

$$\mathbf{P}_{(d',d),(\bar{d},\overline{d'})} = \begin{cases} 1 & \text{if } d = \bar{d} \text{ and } d' = \overline{d'} \\ 0 & \text{otherwise} \end{cases} .$$

The unification operator $\mathbf{Q} \in \mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{D}\times\mathcal{D})}$ has its values defined as:

$$\mathbf{Q}_{(d,t),(\bar{d},\overline{d'})} = \begin{cases} 1 & \text{if } d = \bar{d} = \overline{d'} \\ 0 & \text{otherwise} \end{cases} .$$

The corresponding unification operator $\mathbf{Q} \in \mathbb{R}^{(\mathcal{T}\times\mathcal{D})\times(\mathcal{T}\times\mathcal{T})}$ is defined analogously, by switching the drug and target domains. The values are also defined similarly when $\mathcal{D} = \mathcal{T}$ but in this case we use the notation

$$\mathbf{Q}_{(d,d'),(\bar{d},\overline{d'})} = \begin{cases} 1 & \text{if } d = \bar{d} = \overline{d'} \\ 0 & \text{otherwise} \end{cases} .$$

For convenience, we also give the values of the product of operators $\mathbf{PQ} \in \mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{T}\times\mathcal{T})}$:

$$\mathbf{PQ}_{(d,t),(\bar{t},\overline{t'})} = \begin{cases} 1 & \text{if } t = \bar{t} = \overline{t'} \\ 0 & \text{otherwise} \end{cases}$$

as this product is also heavily used in the forthcoming considerations.

In the following example, we illustrate finite dimensional examples of both commutation and unification operators that are, due to their finiteness, representable as matrices.

Example 1 Consider a finite space of drugs of size $|\mathcal{D}| = 3$ and a finite space of targets $|\mathcal{T}| = 2$. Then, the commutation operator $\mathbf{P} \in \mathbb{R}^{(\mathcal{T}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{T})}$ can be represented as the following matrix:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

where rows and columns are arranged according to the natural order of the target-drug and drug-target pairs, respectively. This is in the literature known as the commutation matrix (see e.g. Magnus and Neudecker (1979)). The unification operator $\mathbf{Q} \in \mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{D}\times\mathcal{D})}$ can be represented as the matrix

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

where the rows and columns are arranged in the natural order of the drug-target pairs and drug-drug pairs, respectively.

From the above definition of the commutation and unification operators, we obtain a cheat sheet of rules indicated by the following lemma:

Theorem 2 *For* $\mathbf{P} \in \mathbb{R}^{(\mathcal{T}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{T})}$, *we have*

$$\begin{aligned} \mathbf{P}\mathbf{P}^T &= \mathbf{P}^T\mathbf{P} = \mathbf{I} \\ \mathbf{P}[\mathbf{D}\otimes\mathbf{T}] &= [\mathbf{T}\otimes\mathbf{D}]\mathbf{P} \\ \mathbf{P}[\mathbf{D}\otimes\mathbf{T}]\mathbf{P}^T &= [\mathbf{T}\otimes\mathbf{D}] \end{aligned}$$

And for $\mathbf{P} \in \mathbb{R}^{(\mathcal{D}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{D})}$

$$\begin{aligned} \mathbf{P} &= \mathbf{P}^T \\ \mathbf{P}[\mathbf{D}\otimes\mathbf{D}] &= [\mathbf{D}\otimes\mathbf{D}]\mathbf{P} \\ \mathbf{P}[\mathbf{D}\otimes\mathbf{D}]\mathbf{P} &= [\mathbf{D}\otimes\mathbf{D}] \end{aligned}$$

Further, *for* $\mathbf{Q} \in \mathbb{R}^{(\mathcal{T}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{D})}$, *we have*

$$\mathbf{Q}[\mathbf{D}\otimes\mathbf{D}]\mathbf{Q}^T = [\mathbf{D}^{\odot 2}\otimes\mathbf{1}]\ .$$

where $\mathbf{D}^{\odot 2}$ *denotes the elementwise square of* $\mathbf{D}$, *and* $\mathbf{1} \in \mathbb{R}^{\mathcal{T}\times\mathcal{T}}$ *is an operator with all values equal to one.*

For the values, *we have*

$$[\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T]_{(d,t),(\bar{d},\bar{t})} = (\mathbf{D}\otimes\mathbf{D})_{(d,d),(\bar{d},\bar{d})}\ ,$$

where $\mathbf{Q} \in \mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{D}\times\mathcal{D})}$, *and*

$$[\mathbf{PQ}(\mathbf{T}\otimes\mathbf{T})\mathbf{Q}^T\mathbf{P}^T]_{(d,t),(\bar{d},\bar{t})} = (\mathbf{T}\otimes\mathbf{T})_{(t,t),(\bar{t},\bar{t})}\ ,$$

where $\mathbf{Q}\in\mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{T}\times\mathcal{T})}$.

Finally, if $\mathcal{D}=\mathcal{T}$, *we further have*:

$$\begin{aligned}
(\mathbf{D}\otimes\mathbf{D})_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d',d),(\overline{d'},\bar{d})}\\
(\mathbf{P}(\mathbf{D}\otimes\mathbf{D}))_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d',d),(\bar{d},\overline{d'})}\\
(\mathbf{Q}(\mathbf{D}\otimes\mathbf{D}))_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d,d),(\bar{d},\overline{d'})}\\
((\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T)_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d,d'),(\bar{d},\bar{d})}\\
(\mathbf{PQ}(\mathbf{D}\otimes\mathbf{D}))_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d',d'),(\bar{d},\overline{d'})}\\
((\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T\mathbf{P})_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d,d),(\overline{d'},\overline{d'})}\\
(\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T)_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d,d),(\bar{d},\bar{d})}\\
(\mathbf{PQ}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T)_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d',d'),(\bar{d},\bar{d})}\\
(\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T\mathbf{P})_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d,d),(\overline{d'},\overline{d'})}\\
(\mathbf{PQ}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T\mathbf{P})_{(d,d'),(\bar{d},\overline{d'})} &= (\mathbf{D}\otimes\mathbf{D})_{(d',d'),(\overline{d'},\overline{d'})}
\end{aligned}$$

Proof The listed results are straightforward operator algebraic manipulations based on Definition 1. □

From the above results, we can conclude the following results concerning certain specific pairwise kernels in particular:

Corollary 1 *The kernel matrices of the linear, second order polynomial, Kronecker product, Cartesian, symmetric, anti-symmetric, ranking and metric learning pairwise kernels for two samples of data, say* $X=(\mathbf{d},\mathbf{t})$ *and* $\overline{X}=(\overline{\mathbf{d}},\overline{\mathbf{t}})$, *can be expressed as* $\mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}})\mathbf{K}_{\mathcal{D},\mathcal{T}}\mathbf{R}(\mathbf{d},\mathbf{t})$, *where* $\mathbf{K}_{\mathcal{D},\mathcal{T}}$ *is the corresponding operator of all kernel values as follows*:

Kernel	$\mathbf{K}_{\mathcal{D},\mathcal{T}}\in\mathbb{R}^{(\mathcal{D}\times\mathcal{T})\times(\mathcal{D}\times\mathcal{T})}$ or $\mathbf{K}_{\mathcal{D},\mathcal{D}}\in\mathbb{R}^{(\mathcal{D}\times\mathcal{D})\times(\mathcal{D}\times\mathcal{D})}$
Linear	$\mathbf{D}\otimes\mathbf{1}+\mathbf{1}\otimes\mathbf{T}$
Poly2D	$\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T+2\mathbf{D}\otimes\mathbf{T}+\mathbf{PQ}(\mathbf{T}\otimes\mathbf{T})\mathbf{Q}^T\mathbf{P}$
Kronecker	$\mathbf{D}\otimes\mathbf{T}$
Cartesian	$\mathbf{D}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{T}$
Symmetric	$(\mathbf{P}+\mathbf{I})(\mathbf{D}\otimes\mathbf{D})$
Anti-Symmetric	$(\mathbf{P}-\mathbf{I})(\mathbf{D}\otimes\mathbf{D})$
Ranking	$(\mathbf{I}-\mathbf{P})(\mathbf{D}\otimes\mathbf{1})(\mathbf{I}-\mathbf{P})$
MLPK	$(\mathbf{I}+\mathbf{P})(\mathbf{I}-\mathbf{Q})(\mathbf{D}\otimes\mathbf{D})(\mathbf{I}-\mathbf{Q})^T(\mathbf{I}+\mathbf{P})$

Their products with vectors can be computed with GVT in $O(\min(\overline{q}n+m\overline{n},\overline{m}n+q\overline{n}))$ *time*.

Proof We first show that the kernel matrices over the whole domain of $\mathcal{D}$ and $\mathcal{T}$ can be compactly expressed with the operator notation and show the indexed case afterwards.

$$
\begin{aligned}
\mathbf{K}^{\text{Kronecker}}_{(d,t),(\bar{d},\bar{t})} &= k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{T}}(t,\bar{t}) \\
&= \mathbf{D}_{d,\bar{d}}\mathbf{T}_{t,\bar{t}} \\
&= (\mathbf{D}\otimes\mathbf{T})_{(d,t),(\bar{d},\bar{t})} \\
\mathbf{K}^{\text{Linear}}_{(d,t),(\bar{d},\bar{t})} &= k_{\mathcal{D}}(d,\bar{d}) + k_{\mathcal{T}}(t,\bar{t}) \\
&= (\mathbf{D}\otimes\mathbf{1}+\mathbf{1}\otimes\mathbf{T})_{(d,t),(\bar{d},\bar{t})} \\
\mathbf{K}^{\text{Poly2D}}_{(d,t),(\bar{d},\bar{t})} &= (k_{\mathcal{D}}(d,\bar{d}) + k_{\mathcal{T}}(t,\bar{t}))^2 \\
&= k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d,\bar{d}) + 2k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{T}}(t,\bar{t}) + k_{\mathcal{T}}(t,\bar{t})k_{\mathcal{T}}(t,\bar{t}) \\
&= (\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T + 2\mathbf{D}\otimes\mathbf{T} + \mathbf{P}\mathbf{Q}(\mathbf{T}\otimes\mathbf{T})\mathbf{Q}^T\mathbf{P})_{(d,t),(\bar{d},\bar{t})} \\
\mathbf{K}^{\text{Cartesian}}_{(d,t),(\bar{d},\bar{t})} &= k_{\mathcal{D}}(d,\bar{d})\delta(t,\bar{t}) + \delta(d,\bar{d})k_{\mathcal{T}}(t,\bar{t}) \\
&= (\mathbf{D}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{T})_{(d,t),(\bar{d},\bar{t})} \\
\mathbf{K}^{\text{Symmetric}}_{(d,d'),(\bar{d},\overline{d'})} &= k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d',\overline{d'}) + k_{\mathcal{D}}(d',\bar{d})k_{\mathcal{D}}(d,\overline{d'}) \\
&= (\mathbf{D}\otimes\mathbf{D})_{(d,d'),(\bar{d},\overline{d'})} + (\mathbf{D}\otimes\mathbf{D})_{(d',d),(\bar{d},\overline{d'})} \\
&= ((\mathbf{P}+\mathbf{I})(\mathbf{D}\otimes\mathbf{D}))_{(d,d'),(\bar{d},\overline{d'})} \\
\mathbf{K}^{\text{Anti-Symmetric}}_{(d,d'),(\bar{d},\overline{d'})} &= k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d',\overline{d'}) - k_{\mathcal{D}}(d',\bar{d})k_{\mathcal{D}}(d,\overline{d'}) \\
&= ((\mathbf{P}-\mathbf{I})(\mathbf{D}\otimes\mathbf{D}))_{(d,d'),(\bar{d},\overline{d'})} \\
\mathbf{K}^{\text{Ranking}}_{(d,d'),(\bar{d},\overline{d'})} &= k_{\mathcal{D}}(d,\bar{d}) - k_{\mathcal{D}}(d',\bar{d}) - k_{\mathcal{D}}(d,\overline{d'}) + k_{\mathcal{D}}(d',\overline{d'}) \\
&= [(\mathbf{I}-\mathbf{P})(\mathbf{D}\otimes\mathbf{1})(\mathbf{I}-\mathbf{P})]_{(d,d'),(\bar{d},\overline{d'})} \\
\mathbf{K}^{\text{MLPK}}_{(d,d'),(\bar{d},\overline{d'})} &= \left(k_{\mathcal{D}}(d,\bar{d}) - k_{\mathcal{D}}(d',\bar{d}) - k_{\mathcal{D}}(d,\overline{d'}) + k_{\mathcal{D}}(d',\overline{d'})\right)^2 \\
&= k_{\mathcal{D}}(d,\bar{d})^2 + k_{\mathcal{D}}(d',\bar{d})^2 + k_{\mathcal{D}}(d,\overline{d'})^2 + k_{\mathcal{D}}(d',\overline{d'})^2 \\
&\quad + 2k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d',\overline{d'}) + 2k_{\mathcal{D}}(d',\bar{d})k_{\mathcal{D}}(d,\overline{d'}) \\
&\quad - 2k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d',\bar{d}) - 2k_{\mathcal{D}}(d,\bar{d})k_{\mathcal{D}}(d,\overline{d'}) \\
&\quad - 2k_{\mathcal{D}}(d',\bar{d})k_{\mathcal{D}}(d',\overline{d'}) - 2k_{\mathcal{D}}(d,\overline{d'})k_{\mathcal{D}}(d',\overline{d'}) \\
&= (\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T + \mathbf{P}\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T \\
&\quad + \mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}\mathbf{P} + \mathbf{P}\mathbf{Q}(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T\mathbf{P} + 2(\mathbf{D}\otimes\mathbf{D}) \\
&\quad + 2\mathbf{P}(\mathbf{D}\otimes\mathbf{D}) - 2\mathbf{Q}(\mathbf{D}\otimes\mathbf{D}) - 2\mathbf{P}\mathbf{Q}(\mathbf{D}\otimes\mathbf{D}) - 2(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T \\
&\quad - 2(\mathbf{D}\otimes\mathbf{D})\mathbf{Q}^T\mathbf{P})_{(d,d'),(\bar{d},\overline{d'})} \\
&= [(\mathbf{I}+\mathbf{P})(\mathbf{I}-\mathbf{Q})(\mathbf{D}\otimes\mathbf{D})(\mathbf{I}-\mathbf{Q})(\mathbf{I}+\mathbf{P})]_{(d,d'),(\bar{d},\overline{d'})}
\end{aligned}
$$

Now, recall that if we have two samples of data, say $X = (\mathbf{d}, \mathbf{t})$ and $\overline{X} = (\overline{\mathbf{d}}, \overline{\mathbf{t}})$, and we intend to calculate all kernel evaluations between data in the first sample with the second sample, the matrix consisting of these kernel evaluations is defined as follows:

$$
\mathbf{K} = \mathbf{R}(\overline{\mathbf{d}}, \overline{\mathbf{t}})\mathbf{K}^{\text{kernel}}(\mathbf{D}, \mathbf{T})\mathbf{R}(\mathbf{d}, \mathbf{t})^T
$$

By setting $\mathbf{d} = \overline{\mathbf{d}}$ and $\mathbf{t} = \overline{\mathbf{t}}$ we may as a special case define the kernel matrix for the training data.

We also have the following rules on permuting either of the indexing matrices with the commutation or the unification operator:

$$\mathbf{R}(\mathbf{d},\mathbf{t})\mathbf{P} = \mathbf{R}(\mathbf{t},\mathbf{d})$$
$$\mathbf{R}(\mathbf{d},\mathbf{t})\mathbf{Q} = \mathbf{R}(\mathbf{d},\mathbf{d})$$
$$\mathbf{P}^T\mathbf{R}(\mathbf{d},\mathbf{t})^T = \mathbf{R}(\mathbf{t},\mathbf{d})^T$$
$$\mathbf{Q}^T\mathbf{R}(\mathbf{d},\mathbf{t})^T = \mathbf{R}(\mathbf{d},\mathbf{d})^T$$

To obtain the incomplete data pairwise kernel matrix, we multiply the complete data pairwise kernel matrix $\mathbf{K}^{\text{kernel}}(\mathbf{D},\mathbf{T})$ with the indexing matrix $\mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}})$ and $\mathbf{R}(\mathbf{d},\mathbf{t})^T$ from left and right sides, respectively. The complete data pairwise kernel matrix is a sum of permuted Kronecker product matrices, so these results imply different indexing matrices for each term in the sum. We can then apply GVT to each term separately. □

We have two ways of calculating an identical matrix-vector product given kernel matrices and sample indices $\overline{\mathbf{d}}, \overline{\mathbf{t}}, \mathbf{d}, \mathbf{t}$, with vectors $\mathbf{a} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^t$:

1. Use the standard matrix vector product with the kernel matrix: $\mathbf{u} \leftarrow \mathbf{K}\mathbf{a}$,
2. Use GVT in Theorem (1): $\mathbf{u} \leftarrow$ vectrick $(\mathbf{D}, \mathbf{T}, \overline{\mathbf{d}}, \overline{\mathbf{t}}, \mathbf{d}, \mathbf{t}, \mathbf{a})$.

In computing the pairwise kernel matrix $\mathbf{K} = \mathbf{R}(\overline{\mathbf{d}},\overline{\mathbf{t}})\mathbf{K}^{\text{kernel}}(\mathbf{D},\mathbf{T})\mathbf{R}(\mathbf{d},\mathbf{t})^T$, only elements in the indexing matrices need to be computed. The computational complexity of implementing approach 1 directly is $O(n\overline{n})$. Based on Theorem 1 and Corollary 1, the complexity of approach 2 for any of the kernels listed in Table 4 is $O(\min(\overline{q}n + m\overline{n}, \overline{m}n + q\overline{n}))$. For the training kernel matrix, these complexities can be simplified as $O(n^2)$ and $O(qn + mn)$.

5 Data sets

We apply the pairwise kernel learning framework to four biological data sets. As shown in Table 5, the data sets have quite different characteristics. They vary in the number of samples, ratio of drugs to targets, homogeneity, density, and features. While our data sets belong to the same domain, the different prediction tasks provide an useful benchmark on how pairwise kernels perform over different applications.

Table 5 Data sets used in the experiments. We report for each data set the number of pairs and unique drugs and targets, and whether the data is homogenous. Density is the fraction of drug target pairs that have known labels. We denote the number of drug kernels $|\mathbf{D}|$, target kernels $|\mathbf{T}|$ and pairwise kernels $|\mathbf{K}|$

Data set	Pairs	Drugs	Targ.	Hom.	Dens.	$\lvert\mathbf{D}\rvert$	$\lvert\mathbf{T}\rvert$	$\lvert\mathbf{K}\rvert$
Heterodimer	5497	1526	1526	X	0.2%	3		6
Metz	93 356	156	1421		42%	2	2	4
Merget	167 995	2967	226		25%	10	9	4
Kernel filling	8 803 089	2967	2967	X	100%	10		6

5.1 Heterodimer

Many proteins bind together and form multiprotein structures called protein complexes, which have essential roles in a variety of biological functions. To understand how proteins function, one needs to identify those sets of proteins that form complexes. A significant fraction of known protein complexes are heterodimers, that is, formed by the assembly of only two proteins. Recent research has taken into account information from measured protein-protein interactions and other possible protein information sources in order to develop new methods for predicting complexes, especially for smaller sizes (Ruan et al. 2018; Maruyama 2011; Ruan et al. 2013).

Labels for a heterodimer data set can be generated from databases of curated protein complexes. We created positive and negative examples following a paper which applied Naive Bayes for supervised learning of heterodimers (Maruyama 2011). The labels are based on CYC2008 (Pu et al. 2008), a comprehensive catalogue of 408 manually curated yeast protein complexes, and WI-PHI (Kiemer et al. 2007), a dataset 49607 (protein,protein)-interactions. A positive (negative) example of a heterodimer is a pair of proteins satisfying the following conditions:

1. is (is not) a heterodimeric protein complex in CYC2008
2. is not (is) a proper subset of any other complex in CYC2008
3. WI-PHI includes the PPI corresponding to it

This results in a total of 152 positive examples and 5345 negative examples.

Following research that sought to improve heterodimer predictions (Ruan et al. 2018), we added protein features by considering domain, phylogenetic profile and subcellular localization properties. The idea is that proteins having a similar specification are more likely to form a complex because they are functionally linked. We obtained the domain and subcellular location information from UniProtKB and the phylogenetic profile from KEGG OC (Nakaya et al. 2012). The feature map ϕ for each of the 1526 proteins is one of three binary vectors (length in parenthesis):

1. $\phi_{\text{dom}}(P_i)_j$: the j-th domain occurs in the protein P_i (2554),
2. $\phi_{\text{phylo}}(P_i)_j$: the j-th genome contains the homolog P_i (768),
3. $\phi_{\text{local}}(P_i)_j$: the j-th subcellular localization contains the protein P_i (83).

We computed the protein kernels $\mathbf{D}$ using the Tanimoto kernel on these binary feature vectors. Given binary vectors $\mathbf{v}$ and $\overline{\mathbf{v}}$ of length l, it is defined as the ratio of bits set to 1 in both vs. bits set to 1 in either: $k_d(\mathbf{v}, \overline{\mathbf{v}}) = \sum_{i=1}^{l} \min(v_i, \overline{v}_i) / \sum_{i=1}^{l} \max(v_i, \overline{v}_i)$.

5.2 Metz

Understanding interactions beween chemical compounds and cellular targets is an important research topic in biology. For example, protein kineases control many aspects of the cell life cycle, and drugs that inhibit specific kineases have been developed to treat several diseases. Large-scale bioactivity assays enable the prediction of interactions across wide panels kinease inhibitors and their potential cellular targets. In particular, supervised machine learning is a promising approach of predicting interactions since it can use

structural similarities among the drug compounds and genomic similarities among target proteins.

Labels for an interaction data set were based on biochemical selectivity assays for clinically relevant kinease inhibitors by Metz et al. (2011). The interaction affinity between a ligand molecule (e.g. a drug compound) and a target molecule (e.g. a protein kinease) reflects how tightly the ligand binds to a particular target, quantified using the inhibition constant K_i. The smaller the K_i bioactivity, the higher the interaction affinity between the chemical compond and the protein kinase. We binarized the real valued interactions using a relatively stringent threshold of $K_i < 28.18$ nm into 2798 interacting and 90 558 non-interacting pairs.

Following a study that investigated how well machine learning based methods work in different prediction tasks (Pahikkala et al. 2015a), we extracted features for both drugs and targets. Drug features were based on chemical properties, where structural fingerprint similarity was computed as the two dimensional (2D) Tanimoto coefficient based on the structure clustering server at PubChem. Target features were based on genomic data, where sequence similarities were computed using a normalized version of the Smith-Waterman (SW) score. In total, we have 156 drugs and 1421 targets, with a symmetric 156×156 (drug, drug)-similarity matrix X_d and a symmetric 1421×1421 (target,target)-similarity matrix X_t. Following the previous study, we used the drug and target similarity matrix rows as feature vectors, computing either a linear kernel $k_{\text{Linear}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ or a Gaussian kernel $k_{\text{Gaussian}}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$ with bandwidth $\gamma = 10^{-5}$ [4]. Assuming that target and drug kernels have the same specification, we then have either linear or Gaussian drug kernels $\mathbf{D}$ and target kernels $\mathbf{T}$.

5.3 Merget

A study of similar drug bioactivity prediction appears in Cichonska et al. (2018), where the task was also to predict the interaction affinity between drug compounds and protein kinease targets. The authors evaluated the pairwise kronecker kernel resulting from 3210 different combinations of 10 drug and 320 target kernels. Many of the pairwise kernels were created by using varying choices of target kernel hyperparameters. This study is interesting for our purposes, because we can use these kernels to evaluate how different pairwise kernels compare on different features.

The labels were created by processing the drug-target interactions in Merget (Merget et al. 2016) updated with ChEMBL bioactivities (Sorgenfrei et al. 2018). The authors used only drugs that had more than 1% of bioactivities across kineases measured and only kineases with both domain and ATP binding pocket amino acid subsequences at PROSITE (Sigrist et al. 2012). This resulted in 2967 drugs and 226 protein kinases, with a total of 167 995 binding values.

The features were defined directly through multiple kernel functions for both drugs and targets. Drug kernels $\mathbf{D}$ were based on Tanimoto kernels using 10 different binary molecular fingerprints obtained with rcdk R package (Guha et al. 2007). Given a fixed choice of hyperparameters they had 9 different protein kernels $\mathbf{T}$: three Gaussian kernels based on gene ontology (GO) annotations, three kernels based Smith-Waterman (SW) sequence similarities, and three generic string (GS) kernels. Gaussian kernels were based on three GO profiles: molecular function, biological process and cellular components. The SW kernels and GS kernels are both based on three possible amino acid sequences: full kinase sequences, kinase domain subsequences and ATP binding pocket subsequences. These

kernels used BLOSUM 50 as amino acid descriptors. These 9 protein kernels were originally expanded into 320 different kernels by varying the choice of hyperparameters.

5.4 Kernel filling

In the final experiment, we use the data set in Cichonska et al. (2018) to define a novel prediction task that has an even larger data set, in order to use it for scalability experiments. The authors calculated 10 different drug kernels $(\mathbf{D}^i)_{i=1\ldots10}$, which can be used both as labels and as features in a kernel filling prediction task. Given $n = 2967$ drugs, each drug kernel is a $\mathbf{D}^i \in \mathbb{R}^{2967\times2967}$ matrix. If some of the $2967 \times 2967 = 8803089$ possible entries are missing, they can be predicted using another kernel that has these entries. For a choice of kernels $i \neq j$, denote $Y = \text{ vec }(\mathbf{D}^i)$ as the label vector and $\mathbf{D}^j$ as the drug kernel. The drug kernel is plugged into a pairwise kernel to predict the label vector.

To create a smaller data set, we can sample an $n \times n$ submatrix from both kernel matrices, and split these entries into n_{train} training samples and use remaining samples as setting 1 test samples. The entries outside the submatrix are test samples in settings 2, 3, and 4. The original data set is dense and real-valued; each (drug, drug)-pair has a latent feature vector encoded by the second kernel. Because we are predicting kernel encoded similarities of n drugs that belong to the same domain, the data set is also homogeneous.

6 Experiments

We implemented ridge regression with the minimum residual optimization method, which is an iterative method for the numerical solution of a system of linear equations. The matrix vector products required within the minimum residual method were computed with either of two algorithms. Given a vector to be multiplied with a pairwise Kronecker kernel matrix, the baseline algorithm uses the explicit kernel matrix and the standard matrix vector product, whereas the fast method uses the GVT algorithm. We used the scipy.sparse.linalg.minres method in the SciPy library. The method CGKronRLS in the RLScore software package includes an user friendly implementation of GVT (Pahikkala and Airola 2016), for example. These two methods are identical except for the calculation of the matrix vector products.

Instead of solving the system completely, the minimum residual method can be run up to a given number of iterations. To speed up training, iterations may be stopped before the least squares solution is reached. In practice, a limited number of iterations is often sufficient to reach optimal model performance, where a separate validation set can be used to check whether model performance increases with more iterations. Limiting the number of iterations is also an effective regularization method, known as early-stopping in the literature. A method that includes early stopping therefore has the number of iterations k as a hyperparameter. Regularization can then be performed either by setting the Tikhonov regularization parameter λ to a small constant and limiting the number of iterations k, or finding an optimal λ and stopping iterations when the model has converged. Figure 3 illustrates the effect of early stopping in the Ki data set. The best validation set AUC was reached either by stopping the training early, or by finding the optimal regularization parameter and running the iterations until convergence.

We implemented early stopping ridge regression as follows. The algorithm fits ridge regression ridge $(Z_{\text{obs}}, k_{\mathcal{D}}, k_{\mathcal{T}}, k_{\mathcal{D},\mathcal{T}},$ setting) given a data set Z_{obs}, drug kernel $k_{\mathcal{D}}$, target

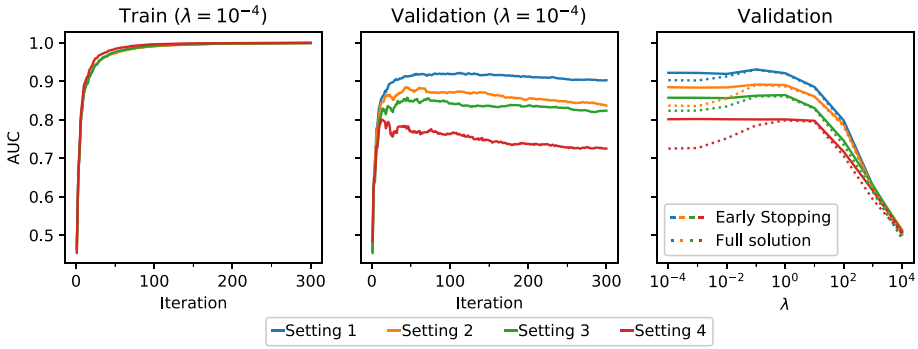

Fig. 3 AUC per iteration and the effect of early stopping in the Ki data set

kernel $k_{\mathcal{T}}$, pairwise kernel $k_{\mathcal{D},\mathcal{T}}$, and setting. We use 9-fold cross-validation, according to the setting (see Table 1), to split the data set into Z_{train} and Z_{test} pair during each round. On each round of cross-validation, the training set Z_{train} is further split into an inner training set Z_{inner} and a validation set $Z_{\text{validation}}$ according to the setting. The optimal hyperparameter for the number of iterations k is then found by running the minimum residual algorithm on Z_{inner} until the AUC stops increasing in $Z_{\text{validation}}$ for a given number of iterations. The number of iterations required and the observed AUC on the validation set is stored. Finally, the model is fit to the full training set Z_{train} using this many iterations. The resulting model is used to make predictions for the test set Z_{test}, for which the AUC is measured.

6.1 Heterodimers

We tested different pairwise kernels, features and settings in the heterodimers data set. The experiment included every combination of following choices:

1. Drug kernel $k_{\mathcal{D}} \in \{k_{\mathcal{D}}^{\text{Domain}}, k_{\mathcal{D}}^{\text{Genome}}, k_{\mathcal{D}}^{\text{Location}}\}$
2. Pairwise kernel $k_{\mathcal{D},\mathcal{T}} \in \{k_{\mathcal{D},\mathcal{T}}^{\text{Linear}}, k_{\mathcal{D},\mathcal{T}}^{\text{Poly2D}}, k_{\mathcal{D},\mathcal{T}}^{\text{Kron.}}, k_{\mathcal{D},\mathcal{T}}^{\text{Cartesian}}, k_{\mathcal{D},\mathcal{T}}^{\text{Symm.}}, k_{\mathcal{D},\mathcal{T}}^{\text{MLPK}}\}$
3. Setting $\in \{1, 2, 3, 4\}$ splits Z_{train} into 75% Z_{inner} and 25% $Z_{\text{validation}}$. We fit ridge regression in Z_{inner} while the AUC in $Z_{\text{validation}}$ is improving.
4. We then train a model $\leftarrow$ ridge ($Z_{\text{train}}, k_{\mathcal{D}}, k_{\mathcal{T}}, k_{\mathcal{D},\mathcal{T}}$, setting) with the optimal number of iterations and calculate the AUC corresponding to Setting $\in \{1, 2, 3, 4\}$ in Z_{test}.

The results in Fig. 4 show that the best pairwise kernel strongly depends on features. For domain features, the MLPK is by far the best pairwise kernel with almost perfect predictions. However, for genome and location features the best kernels are the second degree polynomial and symmetric Kronecker kernels by a notable margin. While the best pairwise kernel depends on the underlying features, using different drug kernel (Min/MinMax/Norm) for the binary feature vectors did not have significant effects, so we report only the Tanimoto, or MinMax, kernel. The best kernel does not seem to vary by the setting, but the later settings are slightly more challenging. The linear kernel that excludes pairwise interactions, and simply models some proteins having more interactions than others, offers suprisingly good results. However, it seems that in this data set there are also significant pairwise interactions that the other kernels are able to capture.

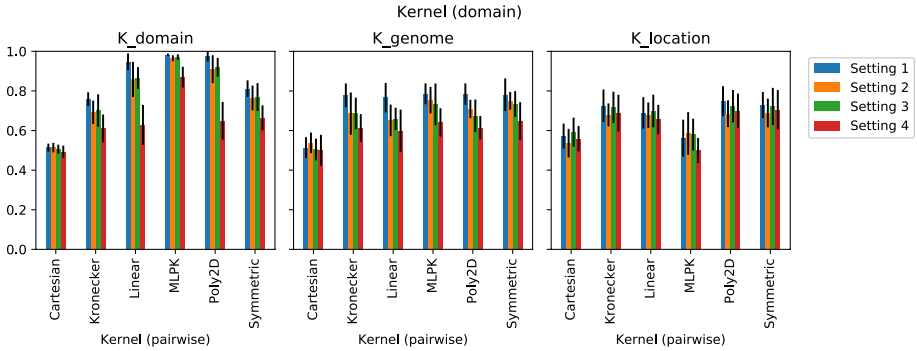

Fig. 4 Heterodimers data set: mean and standard deviation of AUCs in test folds for different kernels and settings

6.2 Metz

We tested different pairwise kernels, features and settings in the Metz data set. The experiment included every combination of following choices:

1. The drug and target kernels $(k_{\mathcal{D}}, k_{\mathcal{T}}) \in \{(k_{\mathcal{D}}^{\text{Linear}}, k_{\mathcal{T}}^{\text{Linear}}), (k_{\mathcal{D}}^{\text{Gaussian}}, k_{\mathcal{T}}^{\text{Gaussian}})\}$
2. The pairwise kernel $k_{\mathcal{D},\mathcal{T}} \in \{k_{\mathcal{D},\mathcal{T}}^{\text{Linear}}, k_{\mathcal{D},\mathcal{T}}^{\text{Poly2D}}, k_{\mathcal{D},\mathcal{T}}^{\text{Kronecker}}, k_{\mathcal{D},\mathcal{T}}^{\text{Cartesian}}\}$
3. Setting $\in \{1, 2, 3, 4\}$ splits Z_{train} into 75% Z_{inner} and 25% $Z_{\text{validation}}$. We fit ridge regression in Z_{inner} while the AUC in $Z_{\text{validation}}$ is improving.
4. We then train a model $\leftarrow$ ridge $(Z_{\text{train}}, k_{\mathcal{D}}, k_{\mathcal{T}}, k_{\mathcal{D},\mathcal{T}},$ setting) with the optimal number of iterations and calculate the AUC corresponding to Setting $\in \{1, 2, 3, 4\}$ in Z_{test}.

The results in Fig. 5 show that for both Linear and Gaussian drug kernels, the second degree polynomial and Kronecker pairwise kernels have the best and comparable performance, because they also include pairwise interactions. The linear kernel offers suprisingly good results, not very far from optimal, but there are also some pairwise interactions that contribute to the prediction task. The cartesian kernel is not much better than random on the task. There seem to be some benefits from using the Gaussian instead of the linear drug

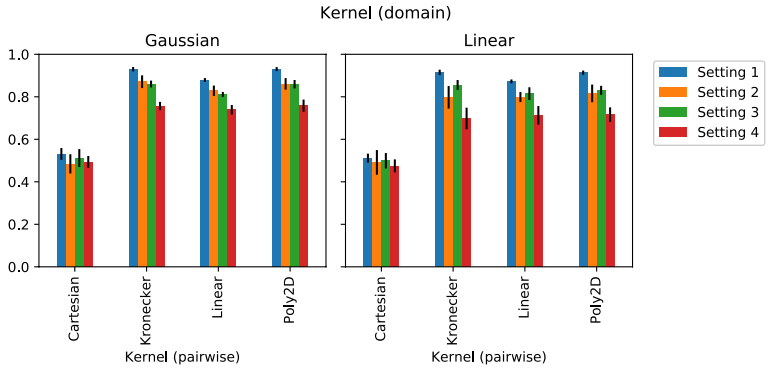

Fig. 5 Metz data set: mean and standard deviation of AUCs in test folds for different kernels and settings

kernel, which are comparable in magnitude to the benefits from modeling pairwise interactions. Regardless of the drug kernels used as features, over different experiments the relative pairwise kernel performance is the same.

6.3 Merget

We tested different pairwise kernels, features and settings in the Merget data set. The experiment included every combination of following choices:

1. The drug and target kernels

$$\begin{aligned}(k_{\mathcal{D}}, k_{\mathcal{T}}) \in \{&(k_{\mathcal{D}}^{\text{sp}}, k_{\mathcal{T}}^{\text{GS-atp-5.4.4}}), (k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{GS-atp-5.4.4}}),\\ &(k_{\mathcal{D}}^{\text{kr}}, k_{\mathcal{T}}^{\text{GS-atp-5.4.4}}), (k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{GS-kindom-5.4.4}}),\\ &(k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{GO-bp-71}}), (k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{GO-cc-19}}),\\ &(k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{SW-kindom}}), (k_{\mathcal{D}}^{\text{circular}}, k_{\mathcal{T}}^{\text{GS-full-5.3.}})\}\end{aligned}$$

2. The pairwise kernel $k_{\mathcal{D},\mathcal{T}} \in \{k_{\mathcal{D},\mathcal{T}}^{\text{Linear}}, k_{\mathcal{D},\mathcal{T}}^{\text{Poly2D}}, k_{\mathcal{D},\mathcal{T}}^{\text{Kronecker}}, k_{\mathcal{D},\mathcal{T}}^{\text{Cartesian}}\}$
3. Setting $\in \{1, 2, 3, 4\}$ splits Z_{train} into 75% Z_{inner} and 25% $Z_{\text{validation}}$. We fit ridge regression in Z_{inner} while the AUC in $Z_{\text{validation}}$ is improving.
4. We then train a model $\leftarrow$ ridge $(Z_{\text{train}}, k_{\mathcal{D}}, k_{\mathcal{T}}, k_{\mathcal{D},\mathcal{T}},$ setting) with the optimal number of iterations and calculate the AUC corresponding to Setting $\in \{1, 2, 3, 4\}$ in Z_{test}.

We obtain close to identical results for different (drug kernel, target kernel)-pairs, so we only present the first two pairs. The results in Fig. 6 closely mirror the Metz data set. Polynomial and Kronecker kernel are the best with comparable performance in all pairs. Linear kernel has almost as good results, even though some pairwise interactions can be found between the drugs and the targets. Cartesian kernel is not much better than random, with an exception in setting 3. Over all possible drug and target kernel pairs, different features do not seem to have much of an effect on prediction performance or relative order of kernels. This is suprising given that the original study was motivated as a method that enables one to use a large mixture of different kernels to improve prediction performance.

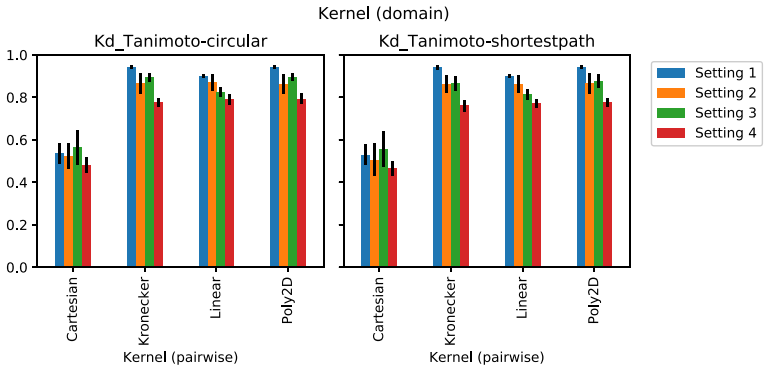

Fig. 6 Merget data set: mean and standard deviation of AUCs in test folds for different kernels and settings

6.4 Kernel filling

We predict the missing labels in a drug kernel matrix $\mathbf{y} = \text{vec}(\mathbf{D}^{\text{circular}})$ using another drug kernel matrix $\mathbf{D} = \mathbf{D}^{\text{estate}}$ as features. Different choices of drug kernels for labels and features result in a drastically different absolute prediction performance. However, not much difference is observed in the relative order of the pairwise kernels. For brevity, we therefore report the experiment on these two kernels, as they offered reasonable but not exceptionally high or low prediction performance.

Because there is so much data available in this task, we used separate test sets. For N samples, the data set Z_{obs} is split into a $(Z_{\text{train}}, Z^{(1)}_{\text{test}}, Z^{(2)}_{\text{test}}, Z^{(3)}_{\text{test}}, Z^{(4)}_{\text{test}})$-partition by taking a subset of k drugs such that approximately 50% of the subset results in Z_{train} with N samples and rest of the subset is $Z^{(1)}_{\text{test}}$, with other drugs defining $Z^{(2)}_{\text{test}}, Z^{(3)}_{\text{test}}, Z^{(4)}_{\text{test}}$. We then tested different drug and pairwise kernels to test how the number of iterations, CPU time, memory usage and the AUC on the test set is affected by the choice of the pairwise kernel. The experiment included every combination of following choices:

1. The pairwise kernel $k_{\mathcal{D},\mathcal{T}} \in \{k^{\text{Linear}}_{\mathcal{D},\mathcal{T}}, k^{\text{Poly2D}}_{\mathcal{D},\mathcal{T}}, k^{\text{Kronecker}}_{\mathcal{D},\mathcal{T}}, k^{\text{Cartesian}}_{\mathcal{D},\mathcal{T}}, k^{\text{Symmetric}}_{\mathcal{D},\mathcal{T}}, k^{\text{MLPK}}_{\mathcal{D},\mathcal{T}}\}$
2. The setting $\in \{1, 2, 3, 4\}$ splits the data set Z_{train} into 75% training set Z_{inner} and 25% validation set $Z_{\text{validation}}$.

We iteratively fit early stopping ridge regression in Z_{inner} while the AUC in $Z_{\text{validation}}$ is improving, and then save the optimal number of iterations. We then train the model on Z_{train} for that many iterations and evaluate the AUC on $Z^{(\text{setting})}_{\text{test}}$.

The number of iterations required to reach an optimal model is shown in Fig. 7. The number of iterations depends on the performance it is possible to achieve in a given setting. More iterations are needed to find a more elaborate model when it is possible to achieve a better prediction performance. Setting 1 requires most iterations, setting 2/3 somewhat less, and setting 4 fewest iterations to reach an optimal solution. Fitting the MLPK and symmetric Kronecker kernel seem to require significantly more iterations than other kernels. Note how the number of iterations is very modest, relative to the total number of samples that is theoretically needed to fully solve the linear system.

The CPU time in seconds and memory usage in GiB are shown in Fig. 7, respectively. The standard method requires significantly more time than the GVT method. At a time when the standard method ran out of memory, the training was taking over an hour whereas GVT completed in a second. The performance of GVT has a small constant term depending on how many summands of Kronecker kernels are required in the pairwise kernel expression. The Kronecker kernel is fastest of these because it has only one term and the MLPK slowest because it has 10 such terms. The naive method requires significantly more memory because it stores the full $O(n^2)$ pairwise kernel matrix whereas GVT only stores the $O(m^2)$ drug and $O(q^2)$ kernel matrices. Here we have $n \approx 0.5q^2$, which implies complexities $O_{\text{naive}}(q^4)$ vs. $O_{\text{GVT}}(q^2)$. The naive method experiments were stopped when N required > 16 GiB memory, which did not become an issue with GVT for the size of this data set.

Prediction performance comparisons, quantified with the AUC in Fig. 7, are quite complicated because they depend on the setting and the size of the data set. We make the following observations in different settings:

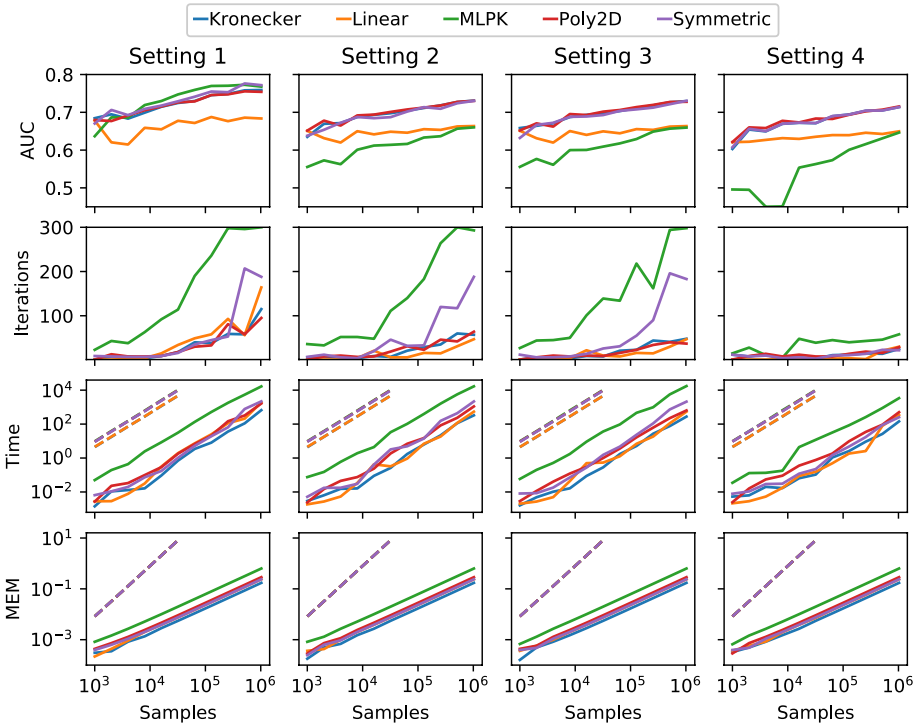

Fig. 7 Kernel filling data set: GVT (solid) versus Baseline (dashed). The AUCs of Kronecker, Poly2D and Symmetric kernels are almost identical and plotted on top of each other. Further, all the baselines have the same memory usage, and their time consumption is close to identical with each other

1. Setting 1: The MLPK kernel has slightly higher performance for larger data sets $N > 10000$. The Kronecker, second degree polynomial, and symmetric Kronecker kernels are comparable to each other and quite close to the MLPK. For medium data sets $N \leq 10000$, they perform better and incorporating prior knowledge via symmetrization may provide a small benefit. The linear kernel is significantly worse except for very small data sets $N \leq 1000$.
2. Setting 2/3: The settings are equivalent because the domain is homogeneous. The MLPK kernel has worst performance for all data set sizes, and the linear kernel is significantly worse except for the smallest $N \leq 1000$ data sets. The Kronecker, second degree polynomial, and symmetric Kronecker kernels have best and almost identical performance. The general prediction accuracy is somewhat lower because the prediction task has become more difficult
3. Setting 4: The results are similar to setting 2/3, but the overall prediction accuracy is even slightly lower.

6.5 Comparison to Nyström approximation with Falkon

The method proposed in this article allows computing efficiently the exact solution to the regularized risk minimization problem for a family of commonly used pairwise kernels. In the following experiments, we compare the approach to a standard approximation

method that speeds up training by using only a random subset of the training data to represent the learned function. Specifically, we compare the proposed method, implemented in RLScore package (Pahikkala and Airola 2016), to the Nyström-method based training algorithm implemented in the Falkon package (Rudi et al. 2017; Meanti et al. 2020). The Nyström approximation allows speeding up kernel methods on large data sets, though there is a trade-off in accuracy if the approximation is not sufficient. This introduces an additional hyperparameter: the number of basis vectors N used as an approximation. The method then computes the kernel $\widetilde{\mathbf{K}} \in \mathbb{R}^{n \times N}$ between data points and basis vectors, and Falkon solves the resulting linear system using a preconditioned conjugate gradient optimizer. Our pairwise kernel implementation inherits the falkon.kernels.kernel.Kernel class from Falkon, and is implemented as c-language extension using Cython to guarantee efficiency. The comparison was implemented on the kernel filling task described in the previous section.

Our experiments in Fig. 8 collaborate theoretic results, which state that increasing the number of basis vectors will result in a higher accuracy when properly regularized. The solution converges to the full solution as the number of basis vectors approaches the number of data points. We also saw that limiting the number of basis vectors effectively regularizes the problem, as the model converges quickly and early stopping results in an identical solution. However, a kernel matrix with 1 024 000 samples and 2048 basis vectors already consumes 16GiB memory so we use this as the best approximation. To align the results with RLScore, we use a regularization parameter $\lambda = 1e-5$ and early stopping based on a validation set. In Fig. 9, we perform experiments to compare the Falkon method with $N = 32, 128, 512, 2048$ basis vectors against RLScore that computes the full solution using GVT, both with the Kronecker product kernel. The experiments are otherwise identical to the previous experiment with the standard kernel method versus the GVT based method. We see that the quality of the approximation increases with the number of basis vectors, almost reaching the AUC of the RLScore implementation. RLScore has lower runtime and lower memory requirement. We conclude that both methods provide a drastically smaller runtime and memory use compared to the standard method, but are quite comparable to each other in computational requirements. RLScore provides slightly better AUC with less computational resources, especially in Setting 1.

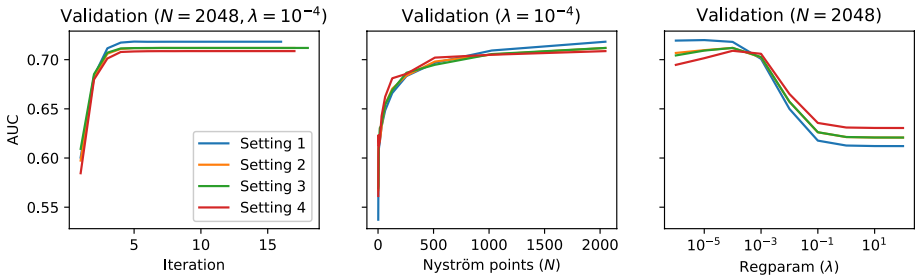

Fig. 8 Tuning the hyperparameters of Falkon package with 64 000 data points: the number of basis vectors (middle) and regularization (right). Only a few iterations are required to reach optimal validation AUC (left)

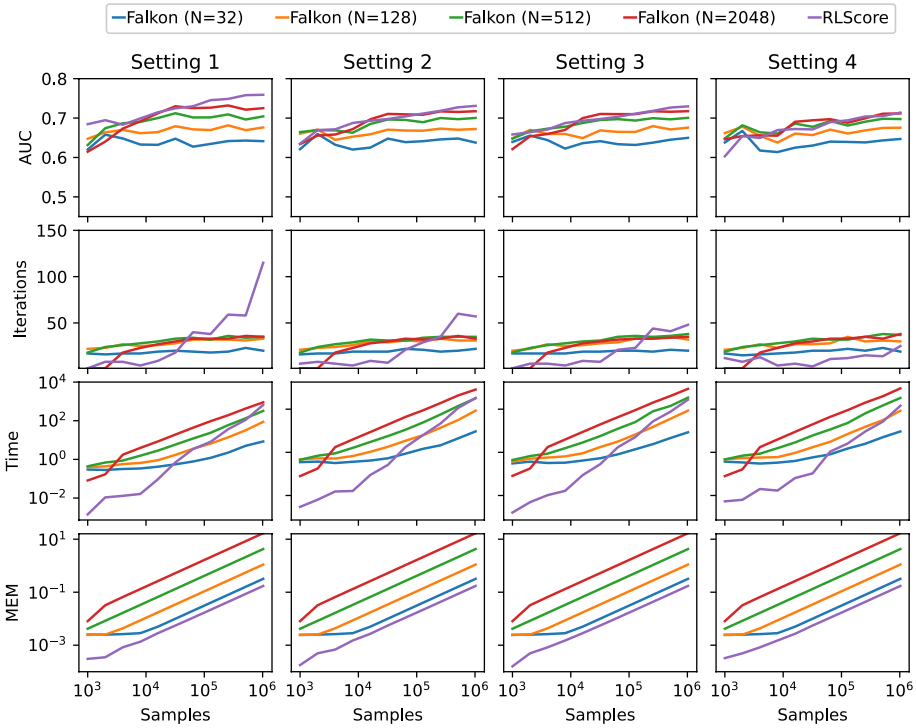

Fig. 9 Kronecker kernel: Nyström approximation given a number of basis vectors implemented by the 'Falkon' package vs. full solution implemented with GVT by 'RLScore'

7 Discussion and conclusion

In this work we reviewed the most commonly used pairwise kernels and introduced an operator based framework for analysing and implementing the kernels. The framework allows applying the generalized vec-trick algorithm (Airola and Pahikkala 2018) for speeding up matrix-vector products for the kernels, allowing much faster training and prediction than with explicit computation of the kernel matrix. As a specific use case we considered the ridge regression method, but the approach can be also used for speeding up the (sub) gradient computations for other types of regularized kernel methods, such as kernel logistic regression or support vector machines. Our experiments on drug-target data show that the approach allows scaling to much larger problem sizes than without the computational short cuts, and provides better predictive performance with the same computational resources than Falkon (Rudi et al. 2017), a the state-of-the art method for training large-scale kernel machines. Further, the choice of optimal kernel is seen to be highly dependant on both the problem domain and the type of prediction task considered.

An interesting observation that can be made from the experimental results is that in many cases the linear pairwise kernel produces results that are competitive with those obtained using the non-linear kernels. This is surprising in the sense that the kernel allows only expressing functions of the form $f(d,t) = f_d(d) + f_t(t)$ that score the drugs and targets separately without truly modeling interactions between them. It seems implausible that the

non-linearity assumption would not hold in the domain of drug-target interaction prediction, or other similar interaction prediction tasks, since this would imply the existence of "universal drug" that would be the optimal choice for all targets. We observed from our experimental results that, with larger sample sizes (see Fig. 7), the nonlinear kernels were better able to capture the nonlinear components of the underlying signal, and the relative strength of the nonlinear part likely determines how large training sample is needed to capture it. An example showing the extreme cases containing either nonlinear or linear signal components only is given in Fig. 1. The nonlinear component also may easily "drown" with high dimensional data, as the number of interaction terms increases quickly as its function.

We make the GVT code publicly available as part of the open source RLScore machine learning library (Pahikkala and Airola 2016), allowing other researchers and developers to make use of the described kernel matrix multiplication short cuts. Our work considers the specific case of pairwise data, an open question remains under what conditions similar efficient methods can be derived in general to nth order tensorial data, which could be a Kronecker product of more than two kernel matrices. For example, the data may consist of triplets (drug, target, cell line) where each object in the triplet has its own kernel.

Acknowledgements The authors would like to thank Dr. Anna Cichonska for her generous help in providing us with the Merget and Kernel filling data sets.

Author Contributions MV wrote the initial draft of the manuscript, with contributions from AA and TP. All the authors edited together and approved the final version. MV was responsible for data processing and experiments, with experimental design done jointly by all the authors. All the authors contributed to writing code. AA and TP jointly planned the study deriving the joint framework for pairwise kernels, with contributions from MV.

Funding Open Access funding provided by University of Turku (UTU) including Turku University Central Hospital. This work was supported by Academy of Finland (Grants 311273, 313266, 340140, 340182).

Availability of data and material The data used in the Metz experiments can be downloaded from http://staff.cs.utu.fi/~aatapa/data/DrugTarget/ and Merget and kernel filling data from https://github.com/aalto-ics-kepaco/pairwiseMKL).

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Ethics approval Not Applicable

Consent to participate Not Applicable

Consent for publication Not Applicable

Code availability The GVT algorithm for fast training of pairwise kernel methods is made publicly available as part of RLScore package under open access license at https://github.com/aatapa/RLScore.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Airola, A., & Pahikkala, T. (2018). Fast Kronecker product kernel methods via generalized vec trick. *IEEE Transactions on Neural Networks and Learning Systems, 29*(8), 3374–3387.

Basilico, J., & Hofmann, T. (2004). Unifying collaborative and content-based filtering. In: Brodley CE (ed) Proceedings of the twenty-first international conference on Machine learning. ACM, pp 65–72.

Bellucci, M., Agostini, F., Masin, M., & Tartaglia, G. G. (2011). Predicting protein associations with long noncoding rnas. *Nature Methods, 8*(6), 444.

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

Bernard, E., Jiao, Y., Scornet, E., Stoven, V., Walter, T., & Vert, J. P. (2017). Kernel multitask regression for toxicogenetics. *Molecular Informatics, 36*(10), 1700053.

Bonilla, E. V., Agakov, F. V., & Williams, C. K. I. (2007). Kernel multi-task learning using task-specific features. In: Meila M, Shen X (eds) 11th International Conference on Artificial Intelligence and Statistics, JMLR.org, JMLR Proceedings, vol 2, pp 43–50.

Brunner, C., Fischer, A., Luig, K., & Thies, T. (2012). Pairwise support vector machines and their application to large scale problems. *Journal of Machine Learning Research, 13*(1), 2279–2292.

Cichonska, A., Ravikumar, B., Parri, E., Timonen, S., Pahikkala, T., Airola, A., et al. (2017). Computational-experimental approach to drug-target interaction mapping: A case study on kinase inhibitors. *PLoS Computational Biology, 13*(8), e1005678.

Cichonska, A., Pahikkala, T., Szedmak, S., Julkunen, H., Airola, A., Heinonen, M., Aittokallio, T., & Rousu, J. (2018). Learning with multiple pairwise kernels for drug bioactivity prediction. *Bioinformatics, 34*(13), i509–i518.

Cichonska, A., Ravikumar, B., Allaway, R. J., Park, S., Wan, F., Isayev, O., Li, S., Mason, M., Lamb, A., Tanoli, Z. U. R., Jeon, M., Kim, S., Popova, M., Capuzzi, S., Zeng, J., Dang, K., Koytiger, G., Kang, J., Wells, C. I., Willson, T. M., Consortium, T. I. D. D. K. B. P. C., Oprea, T. I., Schlessinger, A., Drewry, D. H., Stolovitzky, G., Wennerberg, K., Guinney, J., & Aittokallio, T. (2021). Crowdsourced mapping of unexplored target space of kinase inhibitors. *Nature Communications*, *12*(1), 3307.

Gnecco, G. (2017). Symmetry and antisymmetry properties of optimal solutions to regression problems. *Optimization Letters, 11*(7), 1427–1442.

Gnecco, G. (2018). Symmetric and antisymmetric properties of solutions to kernel-based machine learning problems. *Neurocomputing, 306*, 141–159.

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

Guha, R., et al. (2007). Chemical informatics functionality in r. *Journal of Statistical Software, 18*(5), 1–16.

Herbrich, R. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers, 88,* 115–132.

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

Kashima, H., Oyama, S., Yamanishi, Y., & Tsuda, K. (2009b). On pairwise kernels: An efficient alternative and generalization analysis. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp. 1030–1037.

Kiemer, L., Costa, S., Ueffing, M., & Cesareni, G. (2007). Wi-phi: A weighted yeast interactome enriched for direct physical interactions. *Proteomics, 7*(6), 932–943.

Kuo, T. M., Lee, C. P., & Lin, C. J. (2014). Large-scale kernel ranksvm. In: Proceedings of the 2014 SIAM International Conference on Data Mining. SIAM, pp. 812–820.

Liu, T. Y. (2011). *Learning to rank for information retrieval*. Springer.

Magnus, J. R., & Neudecker, H. (1979). The commutation matrix: Some properties and applications. *The annals of statistics, 7*(2), 381–394.

Maruyama, O. (2011). Heterodimeric protein complex identification. In: Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine. ACM, pp. 499–501.

Meanti, G., Carratino, L., Rosasco, L., & Rudi, A. (2020). Kernel methods through the roof: Handling billions of points efficiently. arXiv:200610350

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

Merget, B., Turk, S., Eid, S., Rippmann, F., & Fulle, S. (2016). Profiling prediction of kinase inhibitors: Toward the virtual assay. *Journal of Medicinal Chemistry, 60*(1), 474–485.

Metz, J. T., Johnson, E. F., Soni, N. B., Merta, P. J., Kifle, L., & Hajduk, P. J. (2011). Navigating the kinome. *Nature Chemical Biology, 7*(4), 200–202.

Minsky, M., & Papert, S. A. (1969). *Perceptrons*. MIT Press.

Nakaya, A., Katayama, T., Itoh, M., Hiranuka, K., Kawashima, S., Moriya, Y., et al. (2012). Kegg oc: A large-scale automatic construction of taxonomy-based ortholog clusters. *Nucleic Acids Research, 41*(D1), D353–D357.

Oyama, S., Manning, C. D. (2004). Using feature conjunctions across examples for learning pairwise classifiers. In: Boulicaut, J. F, Esposito, F., Giannotti, F., Pedreschi, D. (eds) Machine learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy, September 20–24, 2004. Proceedings. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 322–333.

Pahikkala, T., & Airola, A. (2016). Rlscore: Regularized least-squares learners. *Journal of Machine Learning Research, 17*, 1–5.

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

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

Pahikkala, T., Airola, A., Stock, M., Baets, B. D., & 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: Calders T, Esposito F, Hüllermeier E, Meo R (eds) Machine Learning and Knowledge Discovery in Databases. Springer, Lecture Notes in Computer Science, vol. 8725, pp. 517–532.

Pahikkala, T., Airola, A., Pietilä, S., Shakyawar, S., Szwajda, A., Tang, J., & Aittokallio, T. (2015). Toward more realistic drug-target interaction predictions. *Briefings in Bioinformatics, 16*(2), 325–337. https://doi.org/10.1093/bib/bbu010

Pahikkala, T., Viljanen, M., Airola, A., & Waegeman, W. (2015b). Spectral analysis of symmetric and anti-symmetric pairwise kernels. arXiv:150605950

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

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

Pieter, B. T. M. F. W., & Koller, A. D. (2005). Link prediction in relational data. Learning Statistical Patterns in Relational Data Using Probabilistic Relational Models p. 7.

Poggio, T., & Smale, S. (2003). The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS), 50*(5), 537–544.

Pu, S., Wong, J., Turner, B., Cho, E., & Wodak, S. J. (2008). Up-to-date catalogues of yeast protein complexes. *Nucleic Acids Research, 37*(3), 825–831.

Rendle, S. (2010). Factorization machines. In: 2010 IEEE 10th International Conference on Data Mining (ICDM). IEEE, pp. 995–1000.

Romera-Paredes, B., & Torr, P. H. S. (2015). An embarrassingly simple approach to zero-shot learning. In: Bach, F. R., Blei, & D. M. (eds) Proceedings of the 32nd International Conference on Machine Learning, JMLR.org, JMLR Proceedings, vol 37, pp 2152–2161.

Roth, W. E. (1934). On direct product matrices. *Bulletin of the American Mathematical Society, 40*, 461–468.

Ruan, P., Hayashida, M., Maruyama, O., & Akutsu, T. (2013). Prediction of heterodimeric protein complexes from weighted protein-protein interaction networks using novel features and kernel functions. *PLoS ONE, 8*(6), e65265.

Ruan, P., Hayashida, M., Akutsu, T., & Vert, J. P. (2018). Improving prediction of heterodimeric protein complexes using combination with pairwise kernel. *BMC Bioinformatics, 19*(1), 39.

Rudi, A., Carratino, L., & Rosasco, L. (2017). Falkon: An optimal large scale kernel method. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 30.

Saad, Y., & Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing, 7*(3), 856–869.

Schäfer, D., & Hüllermeier, E. (2015). Dyad ranking using a bilinear plackett-luce model. In: Appice, A., Rodrigues, P. P., Santos Costa, V., Gama, J., Jorge, A., Soares, C. (eds) Machine learning and knowledge discovery in databases: European conference. ECML PKDD 2015, Porto, Portugal, September 7–11, 2015, Proceedings, Part II. Springer International Publishing, pp. 227–242.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In: International conference on computational learning theory. Springer, pp. 416–426.

Sigrist, C. J., De Castro, E., Cerutti, L., Cuche, B. A., Hulo, N., Bridge, A., et al. (2012). New and continuing developments at prosite. *Nucleic Acids Research, 41*(D1), D344–D347.
Sorgenfrei, F. A., Fulle, S., & Merget, B. (2018). Kinome-wide profiling prediction of small molecules. *ChemMedChem, 13*(6), 495–499.
Stock, M., Pahikkala, T., Airola, A., De Baets, B., & Waegeman, W. (2018). A comparative study of pairwise learning methods based on kernel ridge regression. *Neural Computation, 30*(8), 2245–2283.
Stock, M., Pahikkala, T., Airola, A., Waegeman, W., & De Baets, B. (2020). Algebraic shortcuts for leave-one-out cross-validation in supervised network inference. *Briefings in Bioinformatics, 21*(1), 262–271.
Vert, J. P., Qiu, J., & Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics, 8*(10), S8.
Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., Stock, M., & De Baets, B. (2012). A kernel-based framework for learning graded relations from data. *IEEE Transactions on Fuzzy Systems, 20*(6), 1090–1101.
Waegeman, W., Dembczyński, K., & Hüllermeier, E. (2019). Multi-target prediction: A unifying view on problems and methods. *Data Mining and Knowledge Discovery, 33*(2), 293–324.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.