OXFORD

Briefings in Bioinformatics, 21(1), 2020, 262–271

doi: 10.1093/bib/bby095
Advance Access Publication Date: 16 October 2018
Problem solving protocol

Algebraic shortcuts for leave-one-out cross-validation in supervised network inference

Michiel Stock, Tapio Pahikkala, Antti Airola, Willem Waegeman and Bernard De Baets

Corresponding author: Michiel Stock, Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium. Tel: (+32) 9 264 60 18; Fax: (+32) 9 264 62 20; E-mail: michiel.stock@ugent.be

Abstract

Supervised machine learning techniques have traditionally been very successful at reconstructing biological networks, such as protein–ligand interaction, protein–protein interaction and gene regulatory networks. Many supervised techniques for network prediction use linear models on a possibly nonlinear pairwise feature representation of edges. Recently, much emphasis has been placed on the correct evaluation of such supervised models. It is vital to distinguish between using a model to either predict new interactions in a given network or to predict interactions for a new vertex not present in the original network. This distinction matters because (i) the performance might dramatically differ between the prediction settings and (ii) tuning the model hyperparameters to obtain the best possible model depends on the setting of interest. Specific cross-validation schemes need to be used to assess the performance in such different prediction settings.

In this work we discuss a state-of-the-art kernel-based network inference technique called two-step kernel ridge regression. We show that this regression model can be trained efficiently, with a time complexity scaling with the number of vertices rather than the number of edges. Furthermore, this framework leads to a series of cross-validation shortcuts that allow one to rapidly estimate the model performance for any relevant network prediction setting. This allows computational biologists to fully assess the capabilities of their models. The machine learning techniques with the algebraic shortcuts are implemented in the RLScore software package: https://github.com/aatapa/RLScore.

Key words: network inference; biological networks; cross-validation; kernel methods

Introduction

Biological systems can be understood as large collections of interacting parts, such as genes, proteins, nucleic acids and small organic molecules. Computational techniques are required to model such biological networks, since the number of possible interactions is simply too large to explore experimentally. Furthermore, high-throughput screenings are often noisy and not always reproducible [1–3]. Supervised machine learning techniques have been successfully used for biological network inference for over a decade [4–8]. Such methods depart from an experimentally determined network, a set of observed interactions, from which a statistical model is learned. This

Michiel Stock is a post-doctoral researcher at the Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium. His research interests include machine learning and computational biology.
Tapio Pahikkala is an assistant professor of machine learning at the Department of Future Technologies, University of Turku, Finland. His research interests include machine learning methods and applications.
Antti Airola is adjunct professor and university lecturer at the Department of Future Technologies, University of Turku, Finland. His research interests include machine learning methods and applications.
Willem Waegeman is an associate professor at the Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium. His research interests include machine learning and its applications.
Bernard De Baets is a senior full professor at the Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium. His research interests include knowledge-based, predictive and spatio-temporal modelling.
Submitted: 27 June 2018; **Received (in revised form):** 21 August 2018

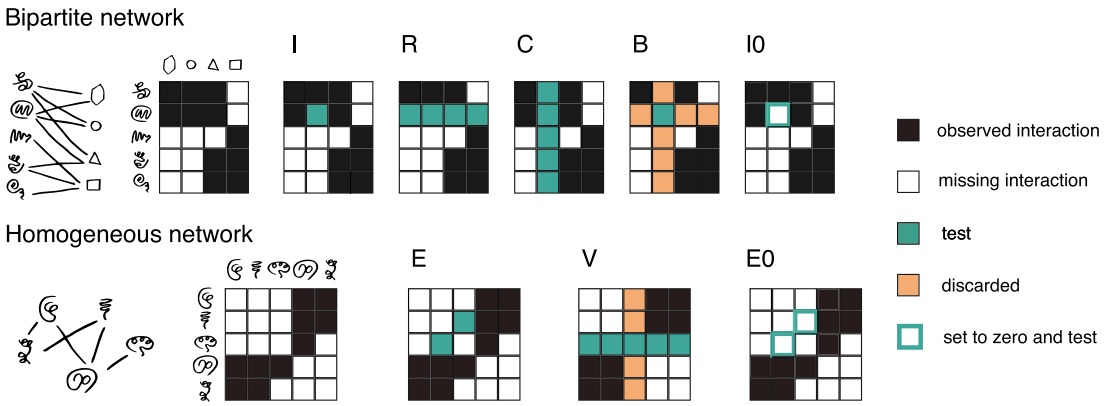

Figure 1. Overview of the network inference problems discussed in this work. (top) Bipartite network prediction, where the vertices are of a different type, e.g. predicting interactions between proteins and ligands. A toy network between five proteins and four ligands with the interaction network is shown. Here, we distinguish four prediction settings: I (interactions), R (rows), C (columns) and B (both). (bottom) Network prediction, where the vertices are of the same type, e.g. protein–protein interaction prediction. A toy network of interactions between five proteins is shown. Here, two prediction settings are distinguished: E (edges) and V (vertices). We also present variants of both Settings I and E: Settings I0 and E0. Here, the value of an interaction is set to zero rather than being discarded. See main text for details.

model can subsequently be used to suggest missing interactions in the given network or to predict interactions with new vertices.

The network used to build the model can be represented as an adjacency matrix, with the rows and columns representing the vertices and the matrix elements the values of the edges between the vertices. If the network is only characterized by the presence or absence of an interaction, the values in the matrix are binary and the matrix is often sparse. If the interaction strength is measured, for example in the form of a binding affinity, the elements of the adjacency matrix are real-valued. In supervised network inference, one also uses a description of the vertices, typically in the form of numerical features (e.g. a molecular fingerprint) or a similarity matrix (e.g. obtained by sequence alignment). Using the example network and these vertex descriptors, a function $f(u, v)$ is learned to predict the interaction value for two given vertices u and v.

Despite the fact that supervised network inference is arguably only a specific application of standard regression or classification algorithms, there are some specific challenges in correctly estimating the performance of a learned model [7, 9, 10]. This is because the set of labeled edges serving as the training data is not independent and identically distributed; the same vertices participate in several edges. A major problem in assessing the performance of models for supervised network inference is that it is not unambiguously clear how to choose an independent test set. Following the work of [9], we distinguish different prediction settings, dependent on whether one is interested in detecting new interactions between the vertices of the trained network or predicting for one or two new vertices. The performance for each of those settings should be assessed accordingly. For each of the settings, we present computational shortcuts to perform suitable leave-one-out (LOO) cross-validation, allowing for extremely rapidly tuning and validating models for such settings. These shortcuts relate to a kernel-based method for network inference, namely two-step kernel ridge regression (KRR) [11–13]. Kernel methods are popular in computational biology for their ability to learn nonlinear associations and to represent complex structured objects such as sequences, graphs and trees [14–16]. As will be discussed later, two-step KRR is a specific way of finding the parameters of a popular class of kernel-based models commonly used for biological network inference.

The remainder of this work is structured as follows. First, we outline the different prediction settings for which we developed specialized cross-validation shortcuts. Subsequently, we discuss the two-step kernel method for network inference and how it relates to other methods. In the next section we provide the computational shortcuts for cross-validation in network inference. The basic shortcuts for KRR are presented first, with the proofs in the appendix. These are combined into the shortcuts for bipartite and homogeneous networks, respectively. The shortcuts are illustrated on some benchmark biological networks in the experimental section. Finally, we end this work with a short discussion and provide some pointers for future research.

Supervised network prediction settings

The different prediction settings for which we present cross-validation shortcuts are depicted in Figure 1. We categorize the cross-validation schemes in two ways. We distinguish between bipartite networks (e.g. protein–ligand interaction networks) and homogeneous networks (e.g. protein–protein networks). The latter case might induce additional structure in the network, for example if protein u' interacts with protein v', then protein v interacts with protein u. Such symmetries need to be taken into account to obtain a fair estimate of the model performance. A second issue is whether the training network can have false negative or missing interactions. If we want to assess whether the model can find such missing interactions in the network, we can delete interactions and see whether the model can recover. This setting is called the zero-one-out (ZOO) setting.

Cross-validation for bipartite network prediction

The first set of cross-validation schemes applies to bipartite networks, i.e. graphs for which the vertices can be subdivided in two disjoint sets and edges only occur between vertices of different sets. Examples of bipartite network inference include protein–ligand interaction prediction [6, 17–21], mRNA–miRNA interaction prediction [22], nucleic acid–protein affinity prediction [23] and drug sensitivity prediction [24]. Suppose that one wants to build a classifier to predict protein–ligand interaction, i.e. a function $f(\cdot, \cdot)$ that takes as input a protein u and a

ligand v and returns a positive value if the pair shows a binding interaction and a negative value otherwise. For a given protein–ligand pair, four predictions settings can be distinguished: (1) the protein and the ligand both occur in the training network, (2) only the protein occurs in the training network, (3) only the ligand occurs in the training network or (4) both vertices are new. For these four settings, we define four respective LOO cross-validation settings. In Setting I, one interaction or value of the adjacency matrix is withheld at a time. In Setting R, every row of the adjacency matrix is withheld once. Similarly, in Setting C, every column is withheld one-by-one. Finally, in Setting B, every element of the adjacency matrix is withheld once and the model is trained using the adjacency matrix with both the row and column containing that element discarded. All bipartite settings are depicted at the top part of Figure 1. Note that even though the toy example illustrates network inference as a binary classification task (predict presence/absence of an interaction), our settings can also be used for regression tasks such as predicting binding affinities between molecules.

Cross-validation for homogeneous network prediction

Some slightly different prediction settings arise for homogeneous networks, i.e. networks for which the vertices are of the same kind. Inference problems for these types of networks arise for protein–protein interaction (ppi) networks [25–27], gene regulatory networks [28–30] and metabolic networks [31–34]. Homogeneous networks are represented by square adjacency matrices. In addition, the adjacency matrix is often symmetric (in case of ppi networks) or, more rarely, skew-symmetric (in metabolic networks, an ingoing flux is always accompanied by a negative outgoing flux of equal magnitude). To accommodate for these properties, we suggest two additional prediction settings for homogeneous networks: predicting for edges within the network or predicting how new vertices will interact with the existing network. We refer to the former setting as Setting E. Here, we remove one edge of the network at a time and predict for this edge. This edge is represented by two values in the adjacency matrix: one above (i.e. the element at position (i,j)) and one below the diagonal (i.e. the element at position (j,i)). To evaluate how new vertices interact with the network, we suggest Setting V. Here, every vertex is removed once from the network and the interaction values of the remaining vertices with this left-out vertex are predicted. Leaving out one vertex corresponds to removing the corresponding row and column of the adjacency matrix. These two cross-validation settings are depicted at the bottom part of Figure 1. Note that when there is no symmetric or skew-symmetric structure in the adjacency matrix, it makes more sense to use the cross-validation schemes developed for bipartite networks.

False negative interactions: LOO versus zero-one-out

Biological networks have one more peculiarity to keep in mind: the occurrence of false negatives. When networks are constructed by experimentally determining all the pairwise interactions, e.g. an assay of kinase inhibition, all interactions are assumed to be correct within experimental error. Usually, however, biological networks are obtained by aggregating positive interactions. This means that there is often no experimental evidence for the absence of an interaction. Missing links in a network are either true negative interactions or false negative ones, i.e. the interaction between the vertices is simply not observed. Several researchers [7, 26, 35–37] discussed how to train supervised models in the absence of true negative interactions. To assess whether a model can detect missing interactions within a network, we propose a small modification of Settings I and E. Rather than withholding interactions, in Setting I0, each interaction of the adjacency matrix is in turn set to zero, regardless whether there was a non-zero interaction value or not. Thus, for every element in the adjacency matrix, the value is set to zero, and a prediction is made for that element using a model trained with the modified adjacency matrix. The same principle is used in Setting E0. Both variants are also depicted in Figure 1. Here, the values of edges are in turn set to zero and hence we denote this as ZOO cross-validation. Such cross-validation schemes have been used, for example, by [19] and [38]. Depending on whether one can deal with false negatives or not, Setting I, resp. Setting E, is more relevant than Setting I0, resp. E0.

Supervised network inference with two-step kernel ridge regression

Throughout this work, we use boldface small cap letters for vectors, e.g. $\mathbf{a}$, and capital letters for matrices, e.g. A. The i-th element of vector $\mathbf{a}$ is denoted by a_i. We use $A_{i.}$ to denote the i-th row vector of A, $A_{.j}$ to denote the j-th column vector of A and A_{ij} for the element at position (i,j) of matrix A. Similarly, for sets of indices $\mathcal{S} \subset \{1,\ldots,n\}$ and $\mathcal{S}' \subset \{1,\ldots,m\}$, $A_{\mathcal{S}.}, A_{.\mathcal{S}'}$ and $A_{\mathcal{S}\mathcal{S}'}$ denote the submatrices of A in which the rows, columns or both are indexed by $\mathcal{S}$ or $\mathcal{S}'$.

Predicting bipartite networks

Supervised network inference starts from a set of labeled interactions. For bipartite networks, one wants to predict an interaction between vertices of two different kinds, e.g. between proteins and ligands or between miRNAs and mRNAs. There are thus two object spaces, $\mathcal{U}$ and $\mathcal{V}$. Suppose the training set contains information on the subsets $U = \{u_i \mid i = 1,\ldots,n\} \subset \mathcal{U}$ and $V = \{v_j \mid j = 1,\ldots,m\} \subset \mathcal{V}$ of both types of objects. There is exactly one observed label Y_{ij} for every pair $(u_i, v_j) \in U \times V$, stored in the $n \times m$ adjacency matrix Y, as illustrated in Figure 1. These labels can either be binary, indicating the presence or absence of an interaction, or real-valued, indicating for example the strength of an interaction or a kinetic parameter.

The goal of supervised network inference is to learn a predictive model $f : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ that takes a pair of vertices as input and returns a value that either estimates the interaction label or that can be interpreted as a score indicating the confidence in the occurrence of an interaction. In kernel methods, such a function is based on a feature description of both vertices by means of kernel functions. Kernels are symmetric and positive-definite functions that quantify the similarity between vertices [39]. They are very popular in bioinformatics because they can be used to implicitly represent structured objects such as sequences, trees and graphs [40]. For example, the Smith–Waterman similarity scores obtained by sequence alignment are routinely used as kernel values [41, 42] for biological sequences. Let $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be the kernel function associated with vertices in $\mathcal{U}$ (for example, the Smith–Waterman similarity score between protein sequences) and, likewise, $g : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ be the kernel function associated with vertices in $\mathcal{V}$ (for example, the Jaccard similarity between

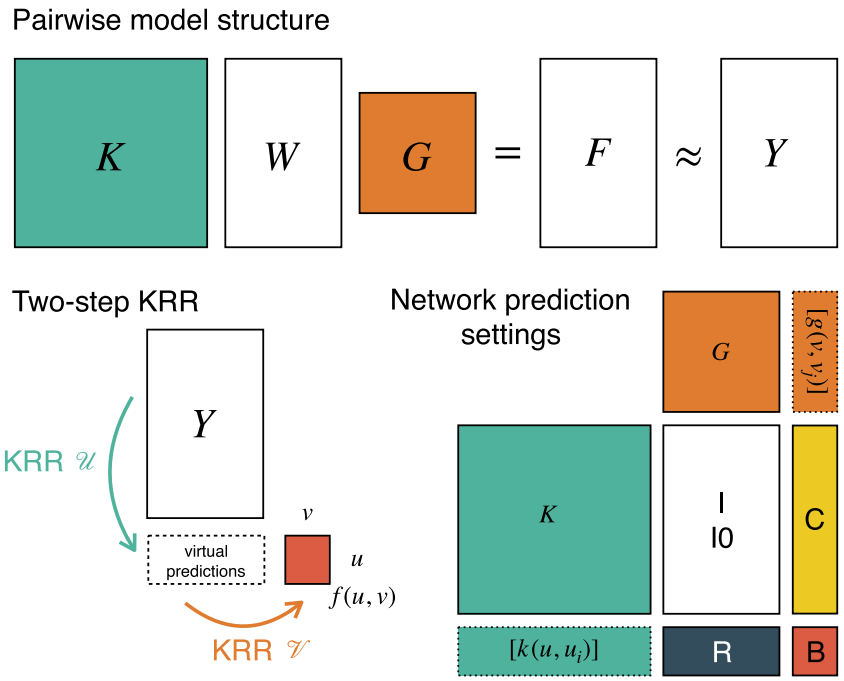

Figure 2. Overview of the two-step KRR method to network prediction. (top) The prediction matrix F is obtained by a bilinear model where the vertices in spaces $\mathcal{U}$ and $\mathcal{V}$ are described by the similarity or Gram matrices K and G, respectively. The model is parametrized by a weight matrix W, computed using Equation (2) such that the prediction matrix approximates the adjacency matrix Y. (bottom left) Conceptually, two-step KRR can be understood as performing ridge regression twice, once to generalize to new vertices of type $\mathcal{U}$ and once for type $\mathcal{V}$. The order of these predictions does not matter and in practice the predictions are computed directly using the equivalent model of Equation (1). (bottom right) Depending on whether one predicts for new vertices of type $\mathcal{U}$ or $\mathcal{V}$, the settings of Figure 1 arise.

fingerprints describing drugs). The model to be learned is of the form

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

with $W = [W_{ij}]$ an $n \times m$ matrix of weights. This model admits an equivalent primal form

$$f(u,v) = \boldsymbol{\phi}(u)^{\top} A \boldsymbol{\psi}(v)\,,$$

with $\boldsymbol{\phi}(\cdot)$ and $\boldsymbol{\psi}(\cdot)$ some feature mappings of objects of type $\mathcal{U}$ and $\mathcal{V}$, respectively, and A the matrix with the primal weights. Models of this type are commonly used for biological network inference, e.g. [4, 10, 17, 19, 23, 34, 43]. They arise naturally when using the Kronecker kernel in a kernel-based learning algorithm such as a support vector machine or KRR. All such models have an algebraic form, as depicted at the top part of Figure 2. Such pairwise kernel methods achieve state-of-the-art performance for many network inference tasks.

In this work, we will focus on the two-step KRR method for fitting the model of Equation (1). This method was independently proposed by [11] and [12], though similar methods have been proposed earlier in structural equation modeling [44, 45]. We introduce the Gram matrices

$$K = [k(u_i,u_j)] \quad \text{and} \quad G = [g(v_i,v_j)]\,.$$

The parameters for two-step KRR are obtained as

$$W = (K + \lambda_u \mathbb{I})^{-1} Y (G + \lambda_v \mathbb{I})^{-1}\,, \tag{2}$$

with $\mathbb{I}$ the identity matrix and λ_u and λ_v two regularization parameters that have to be tuned. Equation (2) can be motivated in several ways. Foremost, it is obtained when, as the name suggests, two successive KRR steps are executed: once to generalize to new objects in $\mathcal{U}$ and once to generalize to new objects in $\mathcal{V}$ (see the bottom left of Figure 1). A similar two-step approach for biological network prediction was introduced by [27], though using tree-based ensembles rather than KRR as learners. In supervised network prediction, constructing a different model for every vertex is called the *local approach*, whereas building a single model that predicts for pairs of edges (cfr. Equation (1)) is called the *global approach* [5]. For instance, in protein–ligand prediction, the local approach would involve building a separate model for every protein to predict drug affinity, whereas the global approach involves one model to predict interaction of protein–ligand pairs. Two-step KRR can be seen as a particularly efficient way of incorporating many local models in a global one. Equation (2) can also be seen as solving the inverse problem of finding the parameters W of model (1) to explain the observed labels Y. The regularization is then required to make this inverse problem well-posed, such that tiny fluctuations in the labels do not result in huge changes in model behavior. Regardless of how two-step KRR is derived, it is a theoretically well-founded method that is closely related to Kronecker KRR [13]. Both methods have the same time complexity and a nearly identical performance [46], as we also experimentally demonstrate later on.

The associated matrix of predictions, $F = [f(u_i,v_j)]$, can easily be computed as

$$F = K(K + \lambda_u \mathbb{I})^{-1} Y (G + \lambda_v \mathbb{I})^{-1} G \tag{3}$$

$$= H^k Y H^g\,, \tag{4}$$

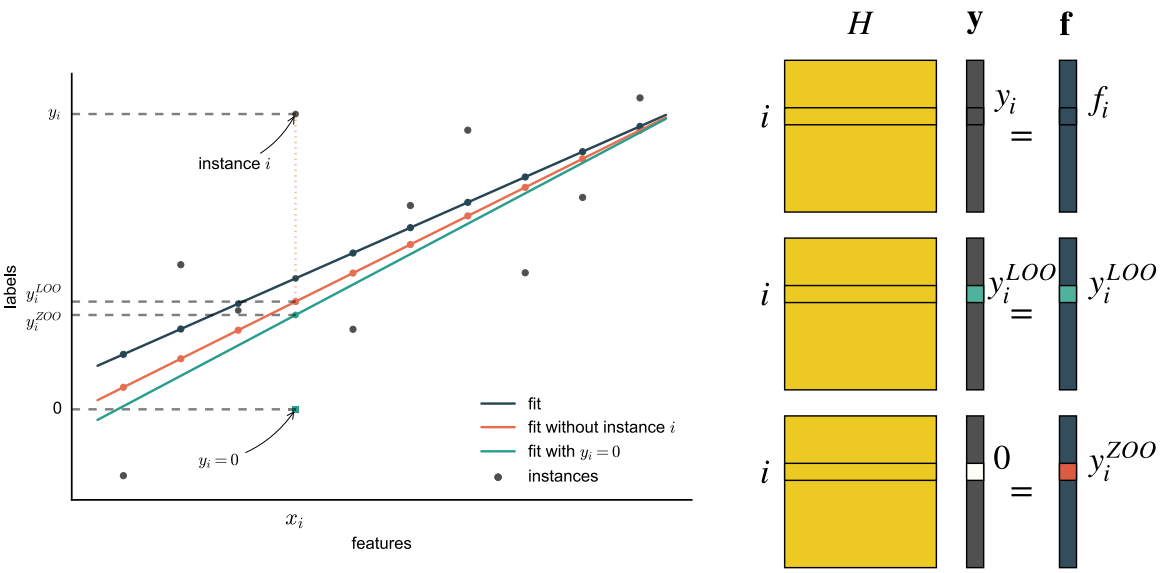

Figure 3. (left) Toy regression problem with 10 instances. The i-th instance has label y_i and a single feature x_i. The blue line is a ridge regression model fit on the data, the red line is the fit without the i-th instance and the green line is the fit in which the label of the i-th instance is set to zero. (right) Predicting and LOO validation as matrix operations.

where $H^k = K(K + \lambda_u \mathbb{I})^{-1}$ and $H^g = G(G + \lambda_v \mathbb{I})^{-1}$ are further on referred to as the hat matrices. It is the simple factorization of Equation (4) that allows for all the shortcuts discussed in this work. All shortcuts can be applied to any model which satisfies this form. Note that given the eigenvalue decompositions of the Gram matrices, i.e.

$$K = U\Lambda U^\top \quad \text{and} \quad G = V\Sigma V^\top,$$

the hat matrices can easily be obtained as

$$H^k = U\Lambda(\Lambda + \lambda_u \mathbb{I})^{-1}U^\top \quad \text{and} \quad H^g = V\Sigma(\Sigma + \lambda_v \mathbb{I})^{-1}V^\top,$$

for any values of the regularization parameters. Note that in the above computations one only has to invert diagonal matrices, which can be done by simply inverting the element on the diagonal. The eigenvalue decompositions can be computed with a time complexity of $\mathcal{O}(n^3 + m^3)$. Given this decomposition, the model weights and the predictions can be obtained by matrix multiplication for any value of the regularization parameters. This is in sharp contrast to a time complexity of $\mathcal{O}(n^3m^3)$ for training pairwise kernel methods without such algebraic tricks. The poor time complexity of pairwise kernel methods without such shortcuts is often given as a critique of these methods [6, 7, 37].

Predicting homogeneous networks

A special, yet important case occurs when interactions between objects of the *same* kind are predicted, e.g. ppi networks, metabolic networks or gene regulatory networks. Here, there is only one set of objects $U \subset \mathcal{U}$ and the adjacency matrix Y is square. The equations hence simplify to

$$f(u, u') = \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} k(u, u_i) k(u', u_j) \tag{5}$$

$$W = (K + \lambda_u \mathbb{I})^{-1} Y (K + \lambda_u \mathbb{I})^{-1} \tag{6}$$

$$F = H^k Y H^k. \tag{7}$$

In some cases, a special structure can be imposed on the adjacency matrix. When the label of every pair (u, u') is always the same as the label of (u', u), this is called a *symmetric* interaction [47]. Such relations occur for example in ppi networks. For symmetric interactions, we expect that $f(u, u') = f(u', u)$ for any $u, u' \in \mathcal{U}$. Likewise, *skew-symmetric* prediction functions satisfy that $f(u, u') = -f(u', u)$ for any $u, u' \in \mathcal{U}$ [47]. Skew-symmetric functions are of relevance when modeling gene regulatory networks and flows in a metabolic network.

Whenever Y is symmetric, the learned prediction function will also be symmetric, i.e. $W = W^\top$. This can be seen from Equation (6). The same holds for skew-symmetric adjacency matrices, where a skew-symmetric adjacency training network will lead to parameters that satisfy $W = -W^\top$.

Shortcuts for LOO cross-validation

Two-step KRR is, as implied by its name, merely performing KRR twice. As such, traditional LOO cross-validation shortcuts can be used as building blocks toward the shortcuts for the more complex network cross-validation schemes. The formal derivations of these shortcuts are presented in the supplementary materials.

Basic shortcuts for LOO cross-validation

The traditional LOO shortcuts can be applied for any model that minimizes a squared loss and where a vector of labels $Y_{i.}$ (here rows from the adjacency matrix) and the corresponding vector of predictions $F_{i.}$ are linked through a hat matrix [48]:

$$F_{i.} = HY_{i.}.$$

Crucially, the hat matrix depends *only* on the feature descriptions of the instances and *not* on the labels. Popular methods such as (kernel) ridge regression, splines, spectral regularization methods and extreme learning machines fall into this category. Figure 3 shows a toy regression problem to illustrate the different shortcuts.

The following theorem gives the shortcut to compute the corresponding predictions for an adjacency matrix with the rows indexed by $\mathcal{S}$ removed.

Theorem 1 For an $n \times m$ matrix of labels Y and an $n \times n$ hat matrix H, the elements indexed by the set $\mathcal{S} \subset \{1, \ldots, n\}$ of the leave-$\mathcal{S}$-out split are given by the $|\mathcal{S}| \times m$ matrix

$$F^{\text{HO}}(Y, H, F, \mathcal{S}) = (\mathbb{I} - H_{\mathcal{S}\mathcal{S}})^{-1}(F_{\mathcal{S}.} - H_{\mathcal{S}\mathcal{S}}Y_{\mathcal{S}.}) .$$

The proof of the theorem, adapted from [49], is given in the supplementary materials. The main idea is that if one replaces the rows indexed by $\mathcal{S}$ in the label matrix by the corresponding rows of $F^{\text{HO}}(Y, H, F, \mathcal{S})$, the corresponding predictions with a model trained with using these labels again yield $F^{\text{HO}}(Y, H, F, \mathcal{S})$. The holdout values are as such 'in equilibrium' with the other label. This is illustrated in Figure 3, in which the LOO prediction for instance i coincides with the corresponding prediction of the model fitted without the i-th instance.

Rather than removing an instance, one can also opt for setting the value to zero. Such a setting is relevant to assess whether the model can detect false negatives, i.e. whether a zero should be a one in the adjacency matrix. We use the term ZOO for the scheme in which models are fitted using data where the labels are set to zero one-by-one. The following theorem gives a computational shortcut to compute the predictions for an adjacency matrix with an arbitrary number of rows set to zero.

Theorem 2 For an $n \times m$ matrix of labels Y and an $n \times n$ hat matrix H, if the elements indexed by the set $\mathcal{S} \subset \{1, \ldots, n\}$ are set to zero, the corresponding predictions are given by the $|\mathcal{S}| \times m$ matrix

$$F^{\text{ZO}}(Y, H, F, \mathcal{S}) = F_{\mathcal{S}.} - H_{\mathcal{S}\mathcal{S}}Y_{\mathcal{S}.} .$$

LOO in bipartite networks

Table 1 shows the shortcuts for the different cross-validation settings for bipartite networks. The shortcuts for Settings I and I0 can easily be deduced by transforming the network prediction problem into a standard prediction problem using the vec operator, which stacks the columns of a matrix into a vector. As such, Equation (3) becomes

$$\text{vec}(F) = (H^g \otimes H^k)\text{vec}(Y) , \tag{8}$$

with $\otimes$ the Kronecker product. We have made use of the identity $\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X)$, which holds for any conformable matrices A, B and X. Theorems 1 and 2 then yield the shortcuts for Settings I and I0. The shortcuts for the Settings R, C and B can be obtained by considering the two steps of two-step KRR separately and applying Theorem 1 on the rows, columns or both, respectively.

The shortcuts for bipartite networks have an approximate time complexity of $\mathcal{O}(nm)$, given that the hat matrices and the prediction matrix have been precomputed. In the case of Settings R, C and B, the matrices H^kY and YH^g, intermediate results toward computing F should also be kept in cache to reach this time complexity.

Table 1. The different shortcuts for bipartite networks. As shown in Figure 1, the settings I (interactions), R (rows), C (columns) and B (both) are considered. The variation of Setting I, Setting I0, is when a single element at position (i, j) is set to zero, rather than being withheld. For bipartite networks $F = H^kYH^g$

Setting	shortcut
I	$F^{\text{I}}_{ij} = \frac{F_{ij} - H^k_{ii}H^g_{jj}Y_{ij}}{1 - H^k_{ii}H^g_{jj}}$
I0	$F^{\text{I0}}_{ij} = F_{ij} - H^k_{ii}H^g_{jj}Y_{ij}$
R	$F^{\text{R}}_{i.} = \frac{F_{i.} - H_{ii}Y_{i.}H^g}{1 - H^k_{ii}}$
C	$F^{\text{C}}_{.j} = \frac{F_{.j} - H^g_{jj}H^kY_{j.}}{1 - H^g_{jj}}$
B	$F^{\text{B}}_{ij} = (F_{ij} - H^k_{ii}Y_{i.}H^g_{.j} - H^g_{jj}H^k_{i.}Y_{.j} + H^k_{ii}H^g_{jj}Y_{ij})$ $(1 - H^k_{ii})^{-1}(1 - H^g_{jj})^{-1}$

Table 2. The different shortcuts for homogeneous networks. As shown in Figure 1, Settings E (edges) and V (vertices) are considered. The shortcuts for Setting E can be used for symmetric (S) and skew-symmetric (SS) matrices. The variation of Setting E, Setting E0, is when the label of the edge from vertex i to vertex j is set to zero, rather than being withheld. For homogeneous networks $F = H^kYH^k$

Setting	shortcut
E (S)	$F^{\text{E}}_{ij} = \frac{F_{ij} - Y_{ij}(H^k_{ii}H^k_{jj} + (H^k_{ij})^2)}{1 - (H^k_{ii}H^k_{jj} + (H^k_{ij})^2)}$
E (SS)	$F^{\text{E}}_{ij} = \frac{F_{ij} - Y_{ij}(H^k_{ii}H^k_{jj} - (H^k_{ij})^2)}{1 - (H^k_{ii}H^k_{jj} - (H^k_{ij})^2)}$
E0 (S)	$F^{\text{E0}}_{ij} = F_{ij} - Y_{ij}(H^k_{ii}H^k_{jj} + (H^k_{ij})^2)$
E0 (SS)	$F^{\text{E0}}_{ij} = F_{ij} - Y_{ij}(H^k_{ii}H^k_{jj} - (H^k_{ij})^2)$
V	$F^{\text{V}}_{i.} = F^{\text{LOO}}_{i.}H^k + H^k_{i.}\frac{F^{\text{LOO}}_{i.}H^k_{.i} - F^{\text{LOO}}_{ii}}{1 - H^k_{ii}}$
-	$F^{\text{LOO}}_{i.} = \frac{H^kY - H^k_{ii}Y_{i.}}{1 - H^k_{ii}}$

Leave-one-pair-out in homogeneous networks

Homogeneous networks impose additional dependencies in the adjacency matrices. To derive the shortcuts for leaving out edges, Equation (7) can also be stated in vector form, as Equation (1). By leaving out the values at positions (i, j) and (j, i) and using the properties of symmetry and skew-symmetry, the shortcuts for Setting E can be derived. The shortcut for Setting V can be obtained by applying the shortcut of Theorem 1 twice. The shortcuts for homogeneous networks are summarized in Table 2. Given that F and H^k are precomputed, these leave-out-out adjacency matrices can be computed with a time complexity of $\mathcal{O}(n^2)$, i.e. constant time complexity for each element.

Experiments

Data sets

In the experiments we will illustrate the speed of the LOO shortcuts provided in this work. To this end, we use four benchmark data sets of bipartite protein–ligand interaction networks collected by [50] and one homogeneous ppi data set from [31, 34].

Table 3. Overview of the different biological networks discussed in this work. We use four bipartite networks from [61], enzymes (*e*), G protein-coupled receptors (*gpcr*), ion channels (*ic*) and nuclear receptors (*nr*) and one homogeneous ppi (*ppi*) network from [62]

data set	*e*	*gpcr*	*ic*	*nr*	*ppi*
# targets	664	95	204	26	769
# drugs	445	223	210	54	-
fraction of interactions (%)	0.99	3.00	3.45	6.41	1.25
median degree targets	2	3	5	3	7
median degree drugs	2	2	3	1	-

The protein–ligand networks were obtained using bioinformatics databases KEGG [51], BRENDA [52], SuperTarget [53] and DrugBank [54]. Note that for the enzyme data set, the interactions relate to regulators, i.e. small molecules acting as inhibitor or activator. Cofactors (except when indicated as regulators) and molecules with a mass smaller than 100 dalton were excluded. The kernel matrix for the proteins is filled using normalized Smith-Waterman scores [55], the kernel matrix for the drugs is obtained by the SIMCOMP similarity score [17].

The homogeneous network is a metabolic protein–protein network from the yeast *Saccharomyces cerevisiae*, collected from the KEGG database. The proteins, *in casu* enzymes, are adjacent in the network if they catalyze subsequent steps in a pathway. Kernel matrices are available describing expression, the phylogenetic profiles of genes, protein localization and yeast two-hybrid ppi data. For the experiments, we combined these four different kernel matrices by averaging, as this approach resulted in the best model by [34] and subsequent studies.

The data sets are described in Table 3. We compare the shortcuts for the bipartite networks with Kronecker KRR (for which only as shortcut for Setting I can be derived), whereas in the subsequent section we study the scalability of the shortcuts for homogeneous networks. In the supplementary materials, we also experimentally study the general holdout shortcuts for Settings R, C and B using a large data set from [23] to predict affinity of proteins for RNA probes. Using larger fold sizes results in relatively smaller, but still substantial speedups.

Bipartite networks

We compare two-step KRR with Kronecker KRR, which have identical time complexities for training and prediction. Kronecker KRR has only one regularization parameter λ, whereas two-step KRR has two regularization parameters, λ_u and λ_v. The main advantages of two-step KRR is the ease of implementation and that the LOO shortcuts can be used. Hence, if the optimal regularization parameter is sought using grid search of d values, two-step KRR requires d^2 computations of the performance in contrast to d for Kronecker ridge regression. For this reason, we also consider a version of two-step KRR where $\lambda_u = \lambda_v = \lambda$, such that also a one-dimensional grid of hyperparameters has to be explored. This trade-off results in a faster tuning at the cost of potentially obtaining a slightly inferior model. In the experiments, the regularization parameters λ, λ_u and λ_v are selected from $\{10^{-7}, 10^{-6}, \ldots, 10^5, 10^6\}$.

For each data set, we perform the LOO cross-validation for Settings I, R, C and B for the different values of the regularization parameters. The prediction matrices obtained using cross-validation are evaluated using micro-wise AUC (i.e. AUC computed over the complete adjacency matrix). For Setting I, a computational shortcut is available for all models. For the remaining settings, only two-step KRR has the respective shortcuts. Hence, in the case of Kronecker KRR, at least one eigenvalue decomposition has to be performed when withholding a protein, ligand or both.

Table 4 shows the best performances for both methods, as well as the running times for performing the complete cross-validation and hyperparameter grid search. For the different settings and data sets, we can observe that both methods have a similar performance, with two-step KRR often slightly outperforming Kronecker KRR. For all methods, Setting B is the hardest and Setting I the easiest, as expected.

When comparing the running times for model selection, we can observe the computational advantage of using the shortcuts. For Setting I, both Kronecker and two-step KRR have a holdout shortcut, hence both are fast. Kronecker KRR has to iterate over a set of 15 regularization values, while two-step KRR has to search a grid of 15×15 regularization parameters, making the latter slower. Both methods are very fast in practice, however, since the main bottleneck is computing the eigenvalue decomposition of the two Gram matrices. For Settings R and C, there is only an efficient algorithm for two-step KRR. For data sets larger than data set *nr*, two-step KRR is much faster than Kronecker KRR. For Setting B two-step KRR is several magnitudes faster compared to Kronecker KRR. For the latter method it was not even possible to perform this cross-validation for the *e* data set within 3 days.

In the supplementary materials, we show how the performance of the different methods changes with the different regularization parameter(s). Two conclusions can be drawn from these experiments. Firstly, the performance is quite sensitive to the chosen regularization parameters. Secondly, the optimal regularization parameters are quite different, depending on the prediction setting that is evaluated. This illustrates the importance of the four cross-validation settings discussed in this work. Using the shortcuts, the best model for the given task can be obtained.

Homogeneous networks

In this section, we explore how the computing time scales for performing cross-validation in homogeneous networks. We again use the four protein–ligand networks of the previous settings, though they are turned into eight square matrices by either multiplying a label matrix with its transpose or multiplying the transpose of an adjacency matrix with the adjacency matrix itself. These represent the number of molecules two proteins or ligands have in common as binding partner. The goal here is to predict for a given molecule how many indirect interactions there are with another molecule of the same type. Though this setting is arguably somewhat artificial, it is well suited to demonstrate our shortcuts. The values of the new label matrices are variance-stabilized by means of a square root transformation.

In addition to the protein–ligand networks, we also use the *ppi* network of [34]. In this work, the proteins are described using seven different Gram matrices, encoding information on the location, expression, phylogeny, etc. Following the original paper, we summed these kernel matrices, as this resulted in the best performing model.

For the different data sets, we measured the time to train a model (i.e. compute the hat matrix H), to make a prediction (i.e. compute the prediction matrix F given H) and to compute the LOO matrices for Settings E and V using our shortcuts. Every computation was done for values of λ in $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$ and we computed the average time over the 11 experiments.

Table 4. Overview of the performance and running time using Kronecker kernel ridge regression (KK), two-step kernel ridge regression (TS) and the two-step method with a single regularization parameter (TSS) for the different drug-target data sets and cross-validation settings. One experiment could not be completed in less than 3 days of running time. See main text for details. All experiments were performed using a basic Numpy implementation of the models and cross-validation shortcuts. All experiments were run on an AMD Opteron server (2500.159 MHz)

		best performance (AUC)				running time			
data		e	gpcr	ic	nr	e	gpcr	ic	nr
Set.	met.								
I	KK	0.964	0.948	0.972	0.866	7.64s	0.22s	0.54s	0.01s
I	TS	0.964	0.942	0.971	0.886	68.68s	3.14s	5.645s	0.215s
I	TSS	0.964	0.942	0.961	0.886	5.96s	0.22s	0.44s	0.02s
R	KK	0.945	0.892	0.947	0.737	1.57h	12.88s	91.14s	0.24s
R	TS	0.948	0.910	0.948	0.783	68.79s	3.17s	5.645s	0.205s
R	TSS	0.948	0.900	0.948	0.724	5.93s	0.22s	0.43s	0.01s
C	KK	0.843	0.871	0.808	0.840	0.69h	81.56s	97.13s	1.01s
C	TS	0.851	0.872	0.808	0.852	68.53s	3.12s	5.745s	0.24s
C	TSS	0.846	0.871	0.803	0.848	5.94s	0.22s	0.43s	0.02s
B	KK	-	0.823	0.769	0.711	>3d	2.15h	5.48h	26.5s
B	TS	0.828	0.834	0.770	0.727	98.6s	3.855s	7.39s	0.25s
B	TSS	0.814	0.827	0.770	0.707	7.95s	0.27s	0.54s	0.01s

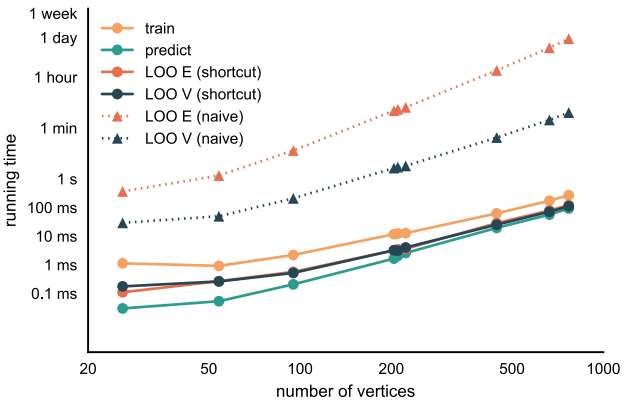

Figure 4. Time for training, predicting and LOO computation in several homogeneous networks. For the training time, the time to construct the hat matrix is taken. Constructing the LOO matrices for Settings E and V using the provided shortcuts takes less time than training, but more than for computing the prediction matrix. The time for computing LOO matrices naively is estimated from the training and prediction times. Experiments performed on a MacBook Pro 2.5 GHz Intel Core i5.

Figure 4 shows how the computing time scales with the number of vertices in the network. Computing the hat matrix is dominated by the eigenvalue decomposition of the hat matrix and takes the most time. Computing the prediction matrix given the hat matrix takes the least time, as this only involves multiplying three matrices. The time to compute the cross-validation matrices takes an intermediate amount of time, though for larger networks these times seem to converge asymptotically to the prediction time. For reference, we added the times it would take to naively compute the cross-validation matrices. For Setting E, resp. Setting V, we took $\binom{n}{2} = \frac{n(n-1)}{2}$, resp. n, times the training time plus one time the prediction time. Again, our shortcuts are many orders of magnitude faster than computing the cross-validation naively. We refer to the supplementary materials for an overview of the performance of the different models.

Discussion and conclusion

Recent surveys have highlighted the importance of correctly blocking when performing cross-validation in 'structured' data [7, 9, 10, 57]. In this work we have presented a series of algebraic shortcuts for LOO cross-validation for the biological network inference problem. These shortcuts are a valuable tool for selecting the best model and for accurately estimating the model performance on new data. The shortcuts apply to a simple, though powerful network inference model, two-step KRR. Kernel methods are generally liked by the computational biology community, both because they are strong learners and because prior knowledge can naturally be assimilated. Given the eigenvalue decomposition of the Gram matrices of the vertices, LOO cross-validation can be performed for any of the discussed settings and any values of the regularization parameters in roughly the time needed to make a prediction of the original adjacency matrix. LOO cross-validation provides nearly unbiased, though sometimes high variance estimates of the generalization error of a model [58]. This is because all the models are trained using largely the same data. Our shortcuts can easily be extended to leaving out larger blocks of the adjacency matrix, using Theorem 1. The shortcuts are described in the supplementary materials.

We believe that linear methods (in a possibly nonlinear feature space) will remain relevant for biological network inference. These will continue to benefit from larger data sets and new methods to generate more biologically relevant feature representations for the vertices. Randomized algorithms allow for approximating the decomposition of huge Gram matrices [59] or constructing a nonlinear feature description directly [60]. Recent advances in convolutional neural networks have resulted in intriguing ways to generate representations for molecules [61], proteins [62] and nucleic acids [63]. In such cases, one would prefer to work in the primal form and the hat matrix is hence computed as

$$H = \Psi(\Psi^\top\Psi + \lambda\mathbb{I})^{-1}\Psi^\top,$$

with Ψ a feature matrix.

Availability

The methods and shortcuts in this work are available in the RLScore software package [64].

Key Points

- Given a biological network as training data, supervised machine learning methods can be used to detect missing interactions or even predict interactions between previously unseen vertices. Pairwise kernel methods are particularly suited to this task, as they provide a natural way of representing an edges.
- Tuning and validating supervised network inference models is no trivial task because several prediction settings can be distinguished, depending on whether the vertices of the test set occur in the training data or not.
- Algebraic tricks can be applied to train, tune and validate pairwise kernel-based methods at lightning speed.

Funding

Research Foundation - Flanders (FWO17/PDO/067 to M.S.). Academy of Finland (grants 311273 and 313266 to T.P. and A.A.).

References

1. Wodak SJ, Pu S, Vlasblom J, Séraphin B. Challenges and rewards of interaction proteomics. *Mol Cell Proteomics* 2009;**8**(1):3–18.
2. Bonetta L. Interactome under construction. *Nature* 2010; **468**(7325):8–11.
3. Prinz F, Schlange T, Asadullah K. Believe it or not: how much can we rely on published data on potential drug targets? *Nat Rev Drug Discov* 2011;**10**(9):712–13.
4. Ben-Hur A, Noble WS. Kernel methods for predicting ppis. *Bioinformatics* 2005;**21**(Suppl 1):i38–46.
5. Vert J-P. Reconstruction of biological networks by supervised machine learning approaches. In: *Elements of Computational Systems Biology*. Hoboken: John Wiley & Sons, 2008, p. 165–88.
6. Ding H, Takigawa I, Mamitsuka H, Zhu S. Similarity-based machine learning methods for predicting drug-target interactions: a brief review. *Brief Bioinform* 2013;**14**(5):734–47.
7. Schrynemackers M, Küffner R, Geurts P. On protocols and measures for the validation of supervised methods for the inference of biological networks. *Front Genet* 2013; **4**(262):1–16.
8. Newman ME. Network structure from rich but noisy data. *Nat Phys* 2018;**14**:542–5.
9. Park Y, Marcotte EM. Flaws in evaluation schemes for pair-input computational predictions. *Nat Methods* 2012;**9**(12):1134–6.
10. Pahikkala T, Airola A, Pietila S, *et al*. Toward more realistic drug-target interaction predictions. *Brief Bioinform* 2015;**16**(2):325–37.
11. Pahikkala T, Stock M, Airola A, *et al*. A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. *Lect Notes Comp Sci* 2014;**8725**:517–32.
12. Romera-Paredes B, Torr PHS. An embarrassingly simple approach to zero-shot learning. In: Feris R, Lampert C, Parikh D (eds). Visual Attributes, Vol. 37. Advances in Computer Vision and Pattern Recognition. Cham: Springer, 2017, p. 2152–61.
13. Stock M, Pahikkala T, Airola A, *et al*. A comparative study of pairwise learning methods based on kernel ridge regression. *Neural Comput* 2018;**30**:2245–83.
14. Schölkopf B, Tsuda K, Vert J-P. *Kernel Methods in Computational Biology*. Cambridge: The MIT Press, 2004.
15. Lodhi H. Computational biology perspective: Kernel methods and deep learning. *Wiley Interdiscip Rev Comput Stat* 2012;**4**(5):455–65.
16. Wang X, Xing EP, Schaid DJ. Kernel methods for large-scale genomic data analysis. *Brief Bioinform* 2015;**16**(2):183–92.
17. Jacob L, Vert J-P. Protein–ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics* 2008;**24**(19):2149–56.
18. Bleakley K, Yamanishi Y. Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics* 2009;**25**(18):2397–403.
19. van Laarhoven T, Nabuurs SB, Marchiori E. Gaussian interaction profile kernels for predicting drug-target interaction. *Bioinformatics* 2011;**27**(21):3036–43.
20. Gönen M. Predicting drug-target interactions from chemical and genomic kernels using Bayesian matrix factorization. *Bioinformatics* 2012;**28**(18):2304–10.
21. Li ZC, Huang MH, Zhong WQ, *et al*. Identification of drug-target interaction from interactome network with 'guilt-by-association' principle and topology features. *Bioinformatics* 2016;**32**(7):1057–64.
22. Van Peer G, De Paepe A, Stock M, *et al*. miRNA target prediction through modeling quantitative and qualitative miRNA binding site information in a stacked model structure. *Nucleic Acids Res* 2016;**45**:e51.
23. Pelossof R, Singh I, Yang JL, *et al*. Affinity regression predicts the recognition code of nucleic acid-binding proteins. *Nat Biotechnol* 2015;**33**(12):1242–9.
24. Costello J, Heiser L, Georgii E, *et al*. A community effort to assess and improve drug sensitivity prediction algorithms. *Nat Biotechnol* 2014;**32**(12):1202–12.
25. Hamp T, Rost B. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics* 2015;**31**(12):1945–50.
26. Liu H, Sun J, Guan J, *et al*. Improving compound-protein interaction prediction by building up highly credible negative samples. *Bioinformatics* 2015;**31**(12):i221–9.
27. Schrynemackers M, Wehenkel L, Babu MM, Geurts P. Classifying pairs with trees for supervised biological network inference. *Mol Biosyst* 2015;**11**(8):2116–25.
28. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P. Inferring regulatory networks from expression data using tree-based methods. *PLoS One* 2010;**5**(9):1–10.
29. Marbach D, Costello J, Küffner R, Vega N. Wisdom of crowds for robust gene network inference. *Nat Methods* 2012;**9**(8):796–807.
30. Maetschke SR, Madhamshettiwar PB, Davis MJ, Ragan MA. Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Brief Bioinform* 2014;**15**(2):195–211.
31. Yamanishi Y, Vert J-P, Kanehisa M. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics* 2004;**20**(Suppl 1):i363–70.
32. Yamanishi Y, Vert J-P, Kanehisa M. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 2005;**21**(Suppl 1): i468–77.
33. Geurts P, Touleimat N, Dutreix M, D'Alché-Buc F. Inferring biological networks with output kernel trees. *BMC Bioinformatics* 2007;**8**(2):S4.

34. Vert J-P, Qiu J, Noble WS. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics* 2007;**8**(S-10):1–10.
35. Elkan C, Noto K. Learning classifiers from only positive and unlabeled data. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, 2008, p. 213–220.
36. Cerulo L, Elkan C, Ceccarelli M. Learning gene regulatory networks from only positive and unlabeled data. *BMC Bioinformatics* 2010;**11**(228):1–16.
37. Park Y, Marcotte EM. Revisiting the negative example sampling problem for predicting protein–protein interactions. *Bioinformatics* 2011;**27**(21):3024–8.
38. De Clercq M, Stock M, De Baets B, Waegeman W. Data-driven recipe completion using machine learning methods. *Trends Food Sci Technol* 2015;**49**:1–13.
39. Schölkopf B, Smola AJ. *Learning with Kernels*. Cambridge: The MIT Press, 2002.
40. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*. Mouscron: Cambridge University Press; 2004.
41. Liao L, Noble WS. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *J Comput Biol* 2003;**10**(6):857–68.
42. Zaki N, Lazarova-Molnar S, El-Hajj W, Campbell P. Protein–protein interaction based on pairwise similarity. *BMC Bioinformatics*, 2009;**10**(150):1–12.
43. Stock M, Poisot T, Waegeman W, De Baets B. Linear filtering reveals false negatives in species interaction data. *Sci Rep* 2017;**7**(45908):1–8.
44. Bollen KA. An alternative two stage least squares (2SLS) estimator for latent variable equations. *Psychometrika* 1996; **61**(1):109–21.
45. Jung S. Structural equation modeling with small sample sizes using two-stage ridge least-squares estimation. *Behav Res Methods* 2013;**45**(1):75–81.
46. Stock M. Exact and efficient algorithms for pairwise learning. PhD diss., Ghent Univerity, 2017.
47. Waegeman W, Pahikkala T, Airola A, *et al*. A kernel-based framework for learning graded relations from data. *IEEE Trans Fuzzy Syst* 2012;**20**(6):1090–101.
48. Wahba G. *Spline Models for Observational Data*. Philadelphia: SIAM, 1990.
49. Rifkin RM, Lippert RA. Notes on regularized least squares. Technical report, MIT, 2007.
50. Yamanishi Y, Araki M, Gutteridge A, *et al*. Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics* 2008; **24**(13):i232–40.
51. Kanehisa M, Goto S, Hattori M, *et al*. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res* 2006;**34**:D354–7.
52. Schomburg I, Chang A, Ebeling C, *et al*. BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Res*. 2004;**32**:431D–3.
53. Günther S, Kuhn M, Dunkel M, *et al*. SuperTarget and Matador: resources for exploring drug-target relationships. *Nucleic Acids Res* 2008;**36**:919–22.
54. Wishart DS, Knox C, Guo AC, *et al*. DrugBank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Res* 2008;**36**:901–6.
55. Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol* 1981;**147**:195–7.
56. Hattori M, Okuno Y, Goto S, Kanehisa M. Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways. *J Am Chem Soc* 2003;**125**(39):11853–65.
57. Roberts DR, Bahn V, Ciuti S, *et al*. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography* 2017;**40**(8):913–29.
58. Varma S, Simon R. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics* 2006; **7**(91):1–8.
59. Gittens A, Mahoney MW. Revisiting the Nyström method for improved large-scale machine learning. *J Mach Learn Res* 2013;**28**(3):567–75.
60. Huang G, Huang G-B, Song S, You K. Trends in extreme learning machines: a review. *Neural Netw* 2015;**61**:32–48.
61. Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, *et al*. Convolutional networks on graphs for learning molecular fingerprints. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*. Cambridge: MIT Press 2015, p. 1–9.
62. Jo T, Hou J, Eickholt J, Cheng J. Improving protein fold recognition by deep learning networks. *Sci Rep* 2015;**5**(1):17573.
63. Alipanahi B, Delong A, Weirauch MT, Frey BJ. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotechnol* 2015;**33**(8):831–8.
64. Pahikkala T, Airola A. RLScore: Regularized Least-Squares learners. *J Mach Learn Res* 2016;**17**:1–5.