Optimization Methods and Software

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/goms20

# Predicting pairwise interaction affinities with $\ell_0$-penalized least squares–a nonsmooth bi-objective optimization based approach*

**Pauliina Paasivirta, Riikka Numminen, Antti Airola, Napsu Karmitsa & Tapio Pahikkala**

**To cite this article:** Pauliina Paasivirta, Riikka Numminen, Antti Airola, Napsu Karmitsa & Tapio Pahikkala (24 Jan 2024): Predicting pairwise interaction affinities with $\ell_0$-penalized least squares–a nonsmooth bi-objective optimization based approach*, Optimization Methods and Software, DOI: 10.1080/10556788.2023.2280784

**To link to this article:** https://doi.org/10.1080/10556788.2023.2280784



Published online: 24 Jan 2024.

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2023.2280784

Taylor & Francis
Taylor & Francis Group



# Predicting pairwise interaction affinities with $\ell_0$-penalized least squares–a nonsmooth bi-objective optimization based approach*

Pauliina Paasivirta[a], Riikka Numminen[b], Antti Airola[b], Napsu Karmitsa[b] and Tapio Pahikkala[b]

[a]Department of Mathematics and Statistics, University of Turku, Turku, Finland; [b]Department of Computing, University of Turku, Turku, Finland

**ABSTRACT**

In this paper, we introduce a novel nonsmooth optimization-based method LMBM-Kron$\ell_0$LS for solving large-scale pairwise interaction affinity prediction problems. The aim of LMBM-Kron$\ell_0$LS is to produce accurate predictions using as sparse a model as possible. We apply the least squares approach with Kronecker product kernels for a loss function and a continuous formulation of $\ell_0$ pseudonorm for regularization. Thus, we end up solving a nonsmooth optimization problem. In addition, we apply a specific bi-objective criterion to strike a balance between the prediction accuracy of the learned model and the sparsity of the obtained solution. We compare LMBM-Kron$\ell_0$LS with some state-of-the-art methods using three benchmark and two simulated data sets under four distinct experimental settings, including zero-shot learning. Moreover, both binary and continuous interaction affinity labels are considered with LMBM-Kron$\ell_0$LS. The results show that LMBM-Kron$\ell_0$LS finds sparse solutions without sacrificing too much in the prediction performance.

**ARTICLE HISTORY**
Received 17 January 2023
Accepted 3 November 2023

**KEYWORDS**
Pairwise interaction affinities; nonsmooth optimization; multiobjective optimization; machine learning; Kronecker product kernel methods; generalized vec-trick; $\ell_0$-penalization; limited nemory bundle method; regression; binary classification; zero-shot learning

## 1. Introduction

Drugs can make specific changes to the pharmaceutical functions of target molecules such as enzymes, ion channels, G protein-coupled receptors (GPCRs), and nuclear receptors [47]. Learning to predict how a specific drug compound interacts with a certain target protein is essential to drug discovery. While biochemical experiments for finding drug-target interactions (DTI) require high human, material, and time investments, computational methods can find potential drug-target pairs more efficiently [11,23,24,46,98]. There are more than 90 million candidate drug compounds recorded in various databases like ChEMBL [59] and PubChem [95]. However, although e.g. PubChem contains around 35 million candidate drug compounds, only less than 7000 of them include target protein

**CONTACT** Napsu Karmitsa napsu@karmitsa.fi Department of Computing, University of Turku, Turku, FI-20014, Finland

*(*In memory of Prof. Andreas Griewank*)

information [82]. Furthermore, out of approximately 20,000 proteins in the human proteome, less than 3000 have been targeted by known drugs [51,89]. As a result, we need to work with sparse data sets where interaction labels have been experimentally validated only for a few drug-target pairs [20,32,79].

In this paper, we propose a method that allows learning sparse models for predicting DTI. Following Airola et al. [1] we tackle the problem of learning a supervised machine learning model from a labelled set of pairs. The pairs are constructed from two component sets: one for a set of drugs $X_d \in \mathcal{D}$ and another for a set of target proteins $X_t \in \mathcal{T}$, both having their own feature representations. The input data is formed out of pairs $\boldsymbol{x}_h = (\boldsymbol{d}_i, \boldsymbol{t}_j)$, $\boldsymbol{d}_i \in X_d$ and $\boldsymbol{t}_j \in X_t$, associated with the label $y_h$ representing the known interaction affinity. Typically the same components appear in multiple pairs, that is, $(\boldsymbol{d}_t, \boldsymbol{t}_v)$, $(\boldsymbol{d}_u, \boldsymbol{t}_w)$, $(\boldsymbol{d}_t, \boldsymbol{t}_w)$, and $(\boldsymbol{d}_u, \boldsymbol{t}_v)$ may all belong to the same training set. Our goal is to learn to predict labels for new drugs and/or targets. Note that the new pair $\boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{t})$ may or may not belong to drugs $X_d$ and targets $X_t$ observed during the training phase.

According to Pahikkala et al. [74] and Chen et al. [20], using quantitative bioactivity data and formulating the prediction problem as a regression problem instead of a standard binary classification problem can lead to more realistic and reliable DTI prediction models. Furthermore, we can divide pairwise prediction tasks into different experimental settings based on the assumptions regarding the overlap between the training data and the new pairs for which predictions are needed [74]. The four distinct settings can be defined as [1,74,75,84]:

- $\boldsymbol{x} \in S1$ if and only if $\boldsymbol{d} \in X_d$ and $\boldsymbol{t} \in X_t$
- $\boldsymbol{x} \in S2$ if and only if $\boldsymbol{d} \in X_d$ but $\boldsymbol{t} \notin X_t$
- $\boldsymbol{x} \in S3$ if and only if $\boldsymbol{d} \notin X_d$ but $\boldsymbol{t} \in X_t$
- $\boldsymbol{x} \in S4$ if and only if $\boldsymbol{d} \notin X_d$ and $\boldsymbol{t} \notin X_t$.

Setting S4 is the most challenging setting called zero-shot learning [80]. It corresponds to generalizing from a partially observed label matrix to predicting unknown labels for such pairs where neither the component of the first set (drugs) nor the component of the second set (targets) has a part in any input pair.

We develop a modified version of Kronecker product kernel regularized least squares (KronRLS, also known as Kronecker ridge regression, see, e.g. [1,53,74]). The prediction function learned by KronRLS, as well as other related kernel methods, is defined by a solution vector which contains a coefficient for each drug-target pair in the training set. The solution vector produced by the original KronRLS is always dense and each drug-target pair, which has non-zero coefficients in the solution vector, needs to be stored in memory when making predictions for a new pair. The main contribution of this work is to introduce a novel nonsmooth (not continuously differentiable, see, e.g. [8]) optimization based approach for producing sparse, yet still accurate, solutions for the regression formulation of the pairwise interaction affinity prediction problems under all four experimental settings S1–S4.

The most important reason to search for a sparse solution vector – that is, a vector with only a few nonzero entries – is that it allows us to recognize the most essential drug-target pairs in terms of capturing the true nature of the data while ignoring the redundant ones.

In addition, the more zeros there are in the solution vector, the faster the prediction process gets. This is due to the fact that in the prediction phase, the dominating costs are the kernel matrix multiplications with the sparse dual coefficient vector [1]. In the proposed method, the sparsity is enforced by replacing the Euclidean norm in the standard Kron-RLS with a continuous formulation of the $\ell_0$ pseudonorm [36,40]. Note that basically the $\ell_0$ pseudonorm simply counts the number of nonzero components of a vector. The resulting objective function is both nonsmooth and nonconvex. We solve this problem by applying an appropriately modified version of the limited memory bundle method (LMBM) introduced by Karmitsa (née Haarala) et al. [43,44]. We utilize this method since it is one of the few optimization algorithms capable of handling large dimensions, nonconvexity, and nonsmoothness. In addition, it has already proved itself in solving other machine learning problems such as clustering [9,49] and missing value imputation [50]. To evaluate the obtained solutions both with respect to prediction accuracy and the obtained sparsity level, a bi-objective framework is utilized. Indeed, our numerical experiments demonstrate that the proposed method, LMBM-Kron$\ell_0$LS, allows us to bring a novel bi-objective approach to large-scale pairwise interaction affinity prediction problems and to derive impressive results.

The structure of the paper is as follows. An overview of the related work is given in Section 2. Section 3 provides theoretical background on pairwise learning problems and kernel methods including Kronecker product kernels. It also defines the continuous formulation of $\ell_0$ pseudonorm and sparse solutions, and, finally, briefly recalls the basic idea of the used optimization method LMBM. In Section 4, the LMBM-Kron$\ell_0$LS algorithm is introduced together with some implementational details. In Section 5, we evaluate the performance of the new method by using both real-life DTI data sets and simulated highly nonlinear pairwise data sets. In addition to solving the related pairwise interaction affinity prediction problems as a realistic regression formulation, we consider a more standard classification formulation to make the results more comparable with the previous research. The obtained results are compared against the state-of-the-art methods Kron-RLS and Kronecker support vector machine (KronSVM) [1,73]. Python implementations of the algorithms (including F2PY interface for the LMBM-Kron$\ell_0$LS algorithm) and tests described in this paper can be found at https://github.com/pavetsu14/LMBM-KronL0LS. In Section 5, we will also show that LMBM-Kron$\ell_0$LS can find much sparser solutions than the competing methods without sacrificing too much in the prediction performance. Finally, Section 6 concludes the paper.

The acronyms and notations used throughout the paper are listed in Table 1.

## 2. Related work

The DTI prediction problem has been studied broadly in the past. Computational methods can be roughly divided into three classes: ligand-based approaches, docking simulation, and chemogenomic approaches. Ligand-based methods like quantitative structure-activity relationship (e.g. [19,26]) utilize the idea that similar molecules usually bind to similar proteins and, thus, predict interactions by comparing a new ligand with the known protein ligands. A severe disadvantage with ligand-based methods is that they perform poorly when the number of known ligands is insufficient [20]. In its turn, docking simulation (e.g. [60]) is a widely used approach in biology. It is a computational technique

**Table 1.** Acronyms and notations.

| | |
|---|---|
| DTI | drug-target interactions |
| GPCR | G protein-coupled receptor |
| GVT | generalized vec-trick |
| CI | concordance index |
| S1–S4 | experimental settings |
| RLS | regularized least squares |
| KronRLS | Kronecker product kernel regularized least squares |
| KronSVM | Kronecker support vector machine |
| LMBM | limited memory bundle method |
| LMBM-Kron$\ell_0$LS | the proposed method |
| ESP | early stopping procedure |
| MO1, MO2 | main objectives |
| SP1, SP2 | starting point procedures |
| $\mathcal{D}, \mathcal{T}$ | spaces of drugs and targets |
| $\mathcal{X}$ | space of drug-target pairs |
| $\mathcal{Y}$ | label space |
| $\mathcal{H}$ | hypothesis space (e.g. reproducing kernel Hilbert space) |
| $X_d, X_t$ | sets of observed drugs and targets, $X_d \in \mathcal{D}, X_t \in \mathcal{T}$ |
| $\boldsymbol{d}, \boldsymbol{t}$ | drugs and targets |
| $\boldsymbol{x}$ | drug-target pairs, $\boldsymbol{x}_h = (\boldsymbol{d}_i, \boldsymbol{t}_j)$ |
| $\boldsymbol{y}$ | labels, label $y_h$ is assosiated with pair $\boldsymbol{x}_h$ |
| $\boldsymbol{p}$ | predicted labels |
| $\boldsymbol{a}$ | dual coefficient (variable) |
| $N$ | number of pairs in training set |
| $M, Q$ | numbers of unique drugs and targets |
| $f$ | prediction function |
| $\mathcal{L}$ | loss function |
| $C$ | penalty function |
| $\lambda, \rho$ | regularization parameters |
| $\boldsymbol{A} \otimes \boldsymbol{U}$ | Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{U}$ |
| $k_{\mathcal{D}}, k_{\mathcal{T}}$ | kernel functions |
| $k_{\mathcal{D},\mathcal{T}}$ | pairwise kernel function (also Kronecker product kernel) |
| $\boldsymbol{K}, \boldsymbol{D}, \boldsymbol{T}$ | kernel matrices |
| $\boldsymbol{r}, \boldsymbol{s}$ | index sequences |
| $\boldsymbol{R}, \boldsymbol{S}$ | index matrices |
| $k$ | (maximum) number of non-zero elements, $k \in [1, \ldots, N]$ |
| $J$ | objective function (for optimization) |
| $\partial J(\boldsymbol{a})$ | Clarke subdifferential of $J$ at $\boldsymbol{a}$ |
| $\boldsymbol{\xi}$ | subgradient, $\boldsymbol{\xi} \in \partial J(\boldsymbol{a})$ |
| $\|\boldsymbol{a}\|_2, \|\boldsymbol{a}\|_1, \|\boldsymbol{a}\|_0, \|\boldsymbol{a}\|_{[k]}$ | Euclidean norm, $\ell_1$-norm, $\ell_0$-pseudonorm, and $\ell_k$-norm |
| $[n]$ | index set $\{1, \ldots, n\}$ ($n \in \mathbb{N}$) |

used to predict how two molecules, typically a small molecule (ligand) and a larger target molecule (protein), interact and bind to each other. The simulation attempts to predict the most favourable binding pose or orientation of the ligand within the protein's binding site. Nevertheless, docking simulation suffers from serious drawbacks related to the availability of the three-dimensional structures of targets and the applicability to large-scale computations; especially in terms of computational time [32].

Chemogenomic approaches refer to methods that combine chemical and genomic data to study the interactions between drugs and their target proteins. These approaches utilize large-scale data on chemical compounds and genetic information to uncover connections between specific drugs and target proteins and to predict potential DTIs. Indeed, chemogenomic approaches have been applied successfully on large-scale DTI prediction problems, for instance, in [26,101,103]. Integrating both chemical space of drugs and genomic space of target proteins into a unified pharmacological space allows these methods to utilize the abundant biological data. Chemogenomic approaches

partition into different categories, such as machine learning-based methods (see, e.g. [1,16,17,30,33,34,37,52,53,56–58,66,67,76,85,86,90,100,102,106,108]) and network-based methods (see, e.g. [21,22,25,96,99,105,107]). From these methods, machine learning-based methods have gained the most attention due to their reliable prediction results. In addition, the possibilities of predicting DTIs via neural network structures and deep learning have been studied with some successes recently (e.g. [6,11,54,55,71,78,79,94,109]).

A specific case of chemogenomic machine learning approaches is KronRLS, where all pairs of drugs and targets are combined into one to make a Kronecker product. This approach gives a notable decrease in elapsed computational time compared to other methods. Indeed, the three top-performing methods in the IDG-DREAM Drug-Kinase Binding Prediction Challenge[1] [27] were an ensemble of KronRLS models, a multitask graph convolutional network-based method, and the XGBoost algorithm. In addition, KronRLS performs similarly to the recently proposed deep learning-based MDeePred method [78]. In this work, we extend the generalized vec trick based KronRLS training method [1] by combining it with $\ell_0$ pseudonorm regularization in order to produce sparse solutions.

Optimization is at the core of machine learning. Although it is a well-known fact that many machine learning problems lead to solving nonsmooth optimization problems (e.g. hinge-loss, lasso, and ReLU), the nonsmooth optimization methods are scarcely used in the machine learning society. The common practice is to minimize nonsmooth functions by ignoring the nonsmoothness and applying a popular and simple smooth solver (like the Newton method) or by applying some smoothing techniques to these functions. However, algorithms based on these approaches may become inaccurate once the size of the data and, thus, the number of smoothing parameters, increases. The welcome exceptions for these approaches are given, for instance, in [41,42], where abs-linear forms of prediction tasks are solved via successive piecewise linearization method, in [9,10,48,49], where nonsmooth formulations of clustering and regression problems are solved with various nonsmooth optimization methods, and in [3–5,35], where nonsmooth optimization is applied in classification and separation tasks.

## 3. Background

In this section, we provide the theoretical background on pairwise learning problems and kernel methods and describe the used performance measure. In addition, we give a continuous formulation of the $\ell_0$ pseudonorm and define sparse solutions. Finally, we briefly recall the basic ideas of LMBM.

### 3.1. Pairwise learning problem

Let $\mathcal{D}$ and $\mathcal{T}$ be the spaces of drugs and targets, respectively. Then, all the possible drug and target pairs are given by the Cartesian product $\mathcal{X} = \mathcal{D} \times \mathcal{T}$. Further, the label space is denoted by $\mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{0, 1\}$ for classification problems. In pairwise learning, the training set consists of a set of pairs $(\boldsymbol{x}_h, y_h)_{h=1}^N \in (\mathcal{D} \times \mathcal{T} \times \mathcal{Y})^N$. Let $X_d = \{\boldsymbol{d}_i\}_{i=1}^M \subset \mathcal{D}$ and $X_t = \{\boldsymbol{t}_j\}_{j=1}^Q \subset \mathcal{T}$ denote the sets of components (drugs and targets) appearing in the training set as a part of at least one pair. Here, constants $M$ and $Q$ define the numbers of unique components connected to the training pairs.

Our goal is to learn a prediction function $f : \mathcal{D} \times \mathcal{T} \to \mathcal{Y}$ from the training set, such that $f$ can correctly predict the labels for a new pair $(\boldsymbol{d}, \boldsymbol{t}) \in \mathcal{D} \times \mathcal{T}$. As mentioned in the introduction, the new pair $(\boldsymbol{d}, \boldsymbol{t})$ may or may not belong to the observed drugs $X_d$ and targets $X_t$.

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

$$f^* = \arg\min_{f \in \mathcal{H}} \mathcal{L}(\boldsymbol{p}, \boldsymbol{y}) + \lambda C(f), \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^N$ are the predicted and $\boldsymbol{y} \in \mathbb{R}^N$ the correct outputs, $\mathcal{L}$ is a convex non-negative loss function typically chosen as a norm, $\lambda > 0$ is a regularization parameter and $C$ is a penalty function. The standard formulation of problem (1) is called regularized least squares (RLS) or ridge regression. It is obtained by choosing the reproducing kernel Hilbert space (RKHS) as the hypothesis space $\mathcal{H}$, a squared loss to loss function $\mathcal{L}$, and a Euclidean norm for penalty $C$ [70,77,91].

### 3.2. Kernel methods and Kronecker product

Kernel methods are among the most widely used approaches in machine learning due to their applicability and strong foundations in statistical learning theory [1]. The methods represent the data using positive semi-definite kernel functions [1,45]. A major advantage of the kernel-based methods is that they provide a principled framework in which many types of data sources can be integrated [97]. Therefore, we follow works like [1,12,13,80,93] and assume that the feature representation of a drug-target pair is defined as the product of the drug and target kernels, respectively, resulting in the Kronecker product kernel. The Kronecker product kernel can be motivated as the simplest kernel that models actual pairwise interactions in drug and target features [91]. In addition, the Kronecker product kernel is a universal kernel if the drug and target kernels are universal (e.g. Gaussian kernels) [87,93]. This signifies that training Kronecker product kernel-based learning algorithms (e.g. ridge regression and SVM) is universally consistent allowing learning complex non-linear relations.

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. By choosing RKHS associated with $k_{\mathcal{D},\mathcal{T}}$ as the hypothesis space $\mathcal{H}$, the generalized representer theorem [83] implies that the minimizing function $f$ can be given as [72,91]:

$$f(\boldsymbol{d}, \boldsymbol{t}) = \sum_{h=1}^{N} a_h k_{\mathcal{D},\mathcal{T}}((\boldsymbol{d}_h, \boldsymbol{t}_h), (\boldsymbol{d}, \boldsymbol{t})),$$

where $\boldsymbol{a} \in \mathbb{R}^N$ is the vector of dual coefficients.

Let $k_{\mathcal{D}} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ and $k_{\mathcal{T}} : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ denote positive semidefinite kernel functions defined for the components $\boldsymbol{d}$ and $\boldsymbol{t}$. The Kronecker product kernel is defined as the product of these two base kernels. That is,

$$k_{\mathcal{D},\mathcal{T}}(\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{d}', \boldsymbol{t}') = k_{\mathcal{D}}(\boldsymbol{d}, \boldsymbol{d}') k_{\mathcal{T}}(\boldsymbol{t}, \boldsymbol{t}').$$

Next, we will define the Kronecker product and some index matrices, which allow us to utilize a computational shortcut called the generalized vec-trick (GVT) [1]. GVT generalizes Roth's column lemma [81] (also known as vec-trick) for incomplete training data (i.e. for data where all interactions are not known) and it allows us to form the Kronecker product kernel matrix implicitly. GVT takes advantage of the sparsity of training labels and its usage leads both to computationally efficient training and test phases (see Theorem 3.4 below).

**Definition 3.1 (Kronecker product):** Let $\boldsymbol{A} \in \mathbb{R}^{a\times b}$ and $\boldsymbol{U} \in \mathbb{R}^{u\times v}$ be matrices, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{U}$ is the $ua \times vb$ block matrix:

$$\boldsymbol{A} \otimes \boldsymbol{U} = \begin{bmatrix} a_{11}\boldsymbol{U} & \cdots & a_{1b}\boldsymbol{U} \\ \vdots & \ddots & \vdots \\ a_{a1}\boldsymbol{U} & \cdots & a_{ab}\boldsymbol{U} \end{bmatrix}.$$

**Definition 3.2 (Index matrix [1]):** Let $\boldsymbol{A} \in \mathbb{R}^{a\times b}$ be a matrix and let $\boldsymbol{s} = (s_1, \ldots, s_c)^T \in [a]^c$ be a sequence of $c$ row indices of $\boldsymbol{A}$ ($[a]$ denotes the index set $\{1, \ldots, a\}$). We say that

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{e}_{s_1}^T \\ \vdots \\ \boldsymbol{e}_{s_c}^T \end{bmatrix},$$

is a row indexing matrix for $\boldsymbol{A}$ determined by $\boldsymbol{s}$. Here $\boldsymbol{e}_i$ is the $i$th standard basis vector of $\mathbb{R}^a$. The column indexing matrix is defined analogously.

**Definition 3.3 (Kronecker product index matrix [1]):** Let $\boldsymbol{A} \in \mathbb{R}^{a\times b}, \boldsymbol{U} \in \mathbb{R}^{u\times v}$ be matrices and let $\boldsymbol{S}$ be an index matrix for Kronecker product $\boldsymbol{A} \otimes \boldsymbol{U}$. Then, $\boldsymbol{S}$ can be expressed as

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{e}_{(b_1-1)u+v_1}^T \\ \vdots \\ \boldsymbol{e}_{(b_c-1)u+v_c}^T \end{bmatrix},$$

where $\boldsymbol{b} = (b_1, \ldots, b_c)^T \in [a]^c$ and $\boldsymbol{v} = (v_1, \ldots, v_c)^T \in [u]^c$ are sequences of row indices of $\boldsymbol{A}$ and $\boldsymbol{U}$, respectively. The entries of $\boldsymbol{b}$ and $\boldsymbol{v}$ are given as

$$b = \lfloor (i-1)/u \rfloor + 1, \quad v = (i-1)\mathrm{mod}u + 1, \tag{2}$$

where $i \in [au]$ denotes a row index of $\boldsymbol{A} \otimes \boldsymbol{U}$, and $b \in [a]$ and $v \in [u]$ map to the corresponding rows in $\boldsymbol{A}$ and $\boldsymbol{U}$, respectively. The column indexing matrix is defined analogously.

**Theorem 3.4 (Speed up with GVT [1]):** Let $\boldsymbol{A} \in \mathbb{R}^{a\times b}, \boldsymbol{U} \in \mathbb{R}^{u\times v}$ be matrices and let $\boldsymbol{R} \in \{0,1\}^{c\times au}$ and $\boldsymbol{C} \in \{0,1\}^{d\times bv}$ be the row and column index matrices for Kronecker product $\boldsymbol{A} \otimes \boldsymbol{U}$, such that $\boldsymbol{R}$ is determined by the sequences $\boldsymbol{b} = (b_1, \ldots, b_c)^T \in [a]^c$ and $\boldsymbol{v} = (v_1, \ldots, v_c)^T \in [u]^c$, and $\boldsymbol{C}$ by $\boldsymbol{w} = (w_1, \ldots, w_d)^T \in [b]^d$ and $\boldsymbol{t} = (t_1, \ldots, t_d)^T \in [v]^d$.

In addition, assume that sequences $\boldsymbol{b}$, $\boldsymbol{v}$, $\boldsymbol{w}$, and $\boldsymbol{t}$, when considered as mappings, are surjective. Let $\boldsymbol{z} \in \mathbb{R}^d$. Then, the product

$$\boldsymbol{R}(\boldsymbol{A} \otimes \boldsymbol{U})\boldsymbol{C}^T\boldsymbol{z} \tag{3}$$

can be computed in $\mathcal{O}(\min(ad + vc, ud + bc))$ time using a sparse Kronecker product multiplication algorithm known as GVT.

In our application, the training set consists of a sequence of labelled pairs $(\boldsymbol{x}_h, y_h)_{h=1}^N = (\boldsymbol{d}_{r_h}, \boldsymbol{t}_{s_h}, y_h)_{h=1}^N \in (\mathcal{D} \times \mathcal{T} \times \mathcal{Y})^N$, where $\boldsymbol{r} = (r_1, \ldots, r_N)^T \in [M]^N$ and $\boldsymbol{s} = (s_1, \ldots, s_N)^T \in [Q]^N$ are the index sequences mapping the training pairs to their corresponding components. According to Definitions 1 and 2, Kronecker product index matrix $\boldsymbol{R} \in \{0, 1\}^{N \times MQ}$ can be determined by using these sequences $\boldsymbol{r}$ and $\boldsymbol{s}$. In addition, let $\boldsymbol{D}_{i,j} = k_{\mathcal{D}}(\boldsymbol{d}_i, \boldsymbol{d}_j)$ and $\boldsymbol{T}_{i,j} = k_{\mathcal{T}}(\boldsymbol{t}_i, \boldsymbol{t}_j)$ denote the component kernel matrices for the training data. Then (cf. Equation (3))

$$\boldsymbol{K} = \boldsymbol{R}(\boldsymbol{T} \otimes \boldsymbol{D})\boldsymbol{R}^T$$

is the Kronecker product kernel matrix containing the kernel evaluations corresponding to the training pairs. Note that Kronecker product kernel matrix $\boldsymbol{K}$ is not formed explicitly in our computations but GVT is used instead.

Kronecker product kernel matrix $\boldsymbol{K}$ can be used to train Kronecker product kernel methods by plugging it into any existing kernel machine solver. Previously Van Laarhoven et al. [53] showed that a special case of the ordinary RLS model, namely KronRLS, can produce state-of-the-art results for DTI prediction, whereas Airola and Pahikkala et al. [1,74] proposed GVT strategy for more efficient training in KronRLS, and showed that, especially in the zero-shot learning setting S4, Kronecker product kernel methods outperform a variety of baseline methods.

Briefly, given a kernel matrix of the pairs $\boldsymbol{K}$ and their real-valued interaction affinity labels $y_h$, we formulate the problem of learning a prediction function $f$ as finding a minimizer of the following (dual) objective function with respect to dual variable $\boldsymbol{a} \in \mathbb{R}^N$:

$$J_{\text{RLS}}(\boldsymbol{a}) = \sum_{i=h}^{N} (y_h - p_h)^2 + \lambda(\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a}), \tag{4}$$

where $\boldsymbol{p} = \boldsymbol{K}\boldsymbol{a}$, $\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a}$ is the squared norm of $f$ in the matrix form, and $\lambda > 0$ is a user-provided regularization parameter determining a compromise between the prediction error and the model complexity [1,74].

In addition to the real-valued interaction affinity labels, we consider the binarized interaction labels. KronRLS is, of course, a valid model to be applied in both cases but, when predicting binarized class labels, the KronSVM method should also be considered in order to compare with state-of-the-art methodologies in both situations. In the case of KronSVM, we minimize the objective function

$$J_{\text{SVM}}(\boldsymbol{a}) = \sum_{h=1}^{N} \max\{0, 1 - p_h y_h\}^2 + \lambda(\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a}) \tag{5}$$

with respect to dual variable $\boldsymbol{a} \in \mathbb{R}^N$. Note that here $y_h \in \{-1, 1\}$.

### 3.3. Concordance index

Concordance index (CI) [38] is an evaluation metric that can be used for measuring the prediction accuracy of a learned model. CI is a rank-based performance measure that is defined as the probability that the predictions for two sample points are in the same order as their real labels. CI is an especially convenient performance metric when it is more important to predict the relative order of labels than their exact values: for instance, ranking the drugs according to their increased likelihood of interacting with a given target. In the case of binary interaction labels, CI is equal to the widely used metric area under the receiver operating characteristic curve [74].

### 3.4. $\ell_0$ pseudonorm and sparse solutions

Our goal is to create a learning algorithm that produces accurate predictions using as sparse a model as possible. Thus, instead of using Euclidean norm as a regularizer like in the standard KronRLS method, we propose taking advantage of the continuous formulation of the $\ell_0$ pseudonorm [36,40].

The $\ell_0$ pseudonorm $\|\boldsymbol{a}\|_0$ simply counts the number of nonzero components of a vector $\boldsymbol{a}$. Therefore, it allows us to control the number of nonzero components of the variable vector (either in the original feature space or upon appropriate kernel transformation) determining the regression curve. Optimization problems that seek sparsity of solutions have recently received broad attention [40]. They can be used, for example, to select a subset of informative variables in regression analysis (see, e.g. [2,14,64,65]).

The sparsity of the solution is enforced by introducing a constraint on the $\ell_0$ pseudonorm, thus, defining a cardinality-constraint optimization problem [40]:

$$\begin{cases} \underset{\boldsymbol{a}}{\text{minimize}} & \mathcal{L}(\boldsymbol{p}, \boldsymbol{y}) \\ \text{subject to} & \|\boldsymbol{a}\|_0 \leq k, \end{cases} \tag{6}$$

with a parameter $k \in [1, \ldots, N]$ defining the number of elements we allow to be nonzero. Due to the nonconvexity and discontinuity of the $\ell_0$ pseudonorm, the optimization problem (6) is known to be NP-hard [68]. Therefore, the $\ell_0$ pseudonorm in (6) is often replaced with its tight convex relaxation – the $\ell_1$ norm [70].

In this work, we tackle problem (6) in a different way. We rely on the class of polyhedral $k$-norms $\|\boldsymbol{a}\|_{[k]}$, defined as the sum of the $k$ largest components of the vector $\boldsymbol{a}$. The fact that $k$-norms are intermediate between $\ell_1$ and $\ell_\infty$, allows us to derive the following equivalence:

$$\|\boldsymbol{a}\|_0 \leq k \iff \|\boldsymbol{a}\|_1 - \|\boldsymbol{a}\|_{[k]} = 0.$$

It is noteworthy that the cardinality-constraint in (6) is defined by a discontinuous function whereas the exact reformulation $\|\boldsymbol{a}\|_1 - \|\boldsymbol{a}\|_{[k]}$ is a continuous function, although both describe the same feasible set [36,40].

In practice, we transfer the cardinality-constrained problem (6) into minimizing an objective function consisting of a loss function and a penalty term [40]:

$$\underset{\boldsymbol{a}}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{p}, \boldsymbol{y}) + \rho(\|\boldsymbol{a}\|_1 - \|\boldsymbol{a}\|_{[k]}) \tag{7}$$

with the penalty parameter $\rho > 0$. It is worth noting that $\|\boldsymbol{a}\|_1 - \|\boldsymbol{a}\|_{[k]} \geq 0$ for all $\boldsymbol{a} \in \mathbb{R}^N$ and, thus, we may view formulation (7) as a penalty function formulation of problem (6). In addition, formulation (7) can be viewed as one of the nonconvex regularization methods, which are recently attracting attention (e.g. [39]). However, to our knowledge, this is the first time the extension of the $k$-norm approach has been applied to a regression problem utilizing kernel transformations.

### 3.5. Limited memory bundle method

Since the objective function of problem (7) is both nonsmooth and nonconvex, a closed-form solution cannot be determined like in the standard KronRLS. Due to the fact that LMBM [43,44] is developed for solving general nonconvex large-scale nonsmooth optimization problems, we modify and use it to solve optimization problem (7). Before going to any further details of the method, we point out that for a nonsmooth function $J$ the gradient $\nabla J(\boldsymbol{a})$ does not need to exist for all $\boldsymbol{a}$. Thus, we need to generalize the notion of gradient [8,29]:

**Definition 3.5:** The *Clarke's subdifferential* $\partial J(\boldsymbol{a})$ of a locally Lipschitz continuous function $J : \mathbb{R}^N \to \mathbb{R}$ at a point $\boldsymbol{a} \in \mathbb{R}^N$ is given by

$$\partial J(\boldsymbol{a}) = \text{conv}\left\{ \lim_{i \to \infty} \nabla J(\boldsymbol{a}^i) \mid \boldsymbol{a}^i \to \boldsymbol{a} \text{ and } \nabla J(\boldsymbol{a}^i) \text{ exists} \right\},$$

where 'conv' denotes the convex hull of a set.

Each vector $\boldsymbol{\xi} \in \partial J(\boldsymbol{a})$ is called a subgradient and the point $\boldsymbol{a}^* \in \mathbb{R}^N$ is stationary, if $0 \in \partial J(\boldsymbol{a}^*)$. Note that stationarity is a necessary condition for local optimality [8].

A simplified flowchart of LMBM is given in Figure 1. LMBM exploits the ideas of the variable metric bundle method [92] by utilizing null steps, simple aggregation of subgradients, and subgradient locality measures. The usage of null steps gives further information about the nonsmooth objective in situations when the search direction does not satisfy the criterion for a serious step (see Figure 1). In its turn, the aggregation procedure gives a possibility to retain the global convergence without solving the rather complicated quadratic direction finding subproblem appearing in the standard bundle methods (see, e.g. [7]). Instead, LMBM uses the subgradient locality measures and the convex combination of at most three previously computed subgradients to form a new aggregate subgradient. The subgradient locality measure generalizes the classical linearization error for nonconvex functions and hence offers a way to approximate how much the subgradient calculated at a trial point ($\boldsymbol{a}_{\text{new}}$ in Figure 1) deviates from being a member of the subdifferential of the current iteration point ($\boldsymbol{a}$ in Figure 1)[43,44].

LMBM calculates the search direction with a limited memory approach [18]. In contrast to the variable metric bundle method, LMBM avoids storing and manipulating large matrices since it uses only a few correction vectors to represent the limited memory variable metric updates. As a result, the number of operations needed for the calculation of the search direction and the aggregate values is only linearly dependent on the number of variables which makes LMBM suitable for large-scale optimization. Especially, there is no time-consuming quadratic direction finding subproblem and the size of the bundle is not dependent on the dimension of the problem (cf. standard bundle methods) [43,44].

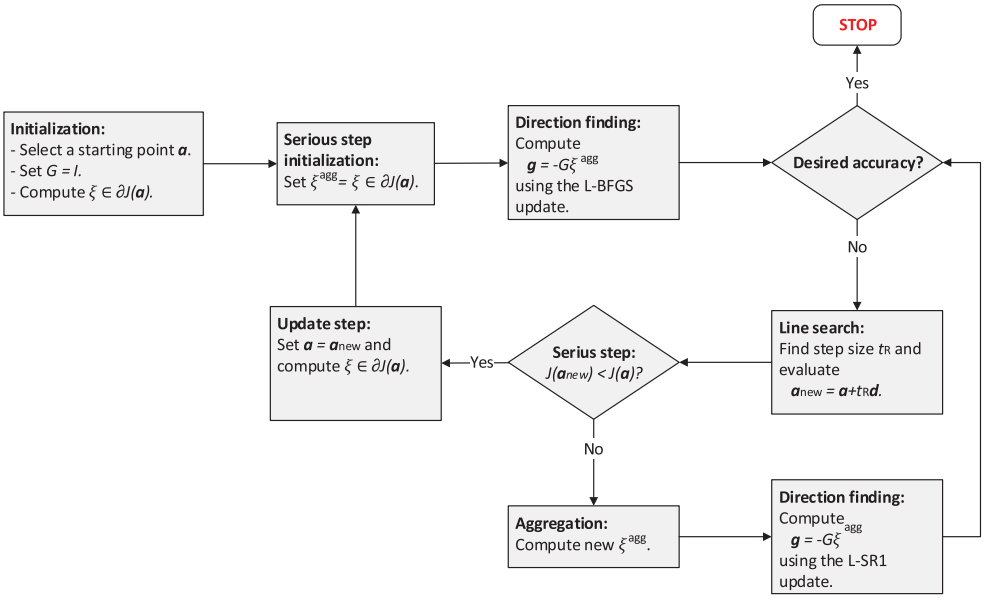


**Figure 1.** LMBM. Here, matrix $G$ denotes the inverse variable metric approximation of the Hessian and vector $\xi^{agg}$ is an aggregate subgradient.

For all locally Lipschitz continuous and upper semismooth objective functions, with the objective function value and an arbitrary subgradient available at every point, LMBM is guaranteed to converge globally [44].

## 4. LMBM-Kron$\ell_0$LS

Next, we propose a new nonsmooth optimization-based method LMBM-Kron$\ell_0$LS to solve large-scale pairwise interaction affinity prediction problems. We aim to learn a model from a database of known drugs, targets, and their observed interactions and to use it to predict how new previously unseen drug-target pairs interact. The training phase corresponds to an iterative reduction of the objective function value $J(\boldsymbol{a})$ defined as a combination of the loss function part of the Equation (4) and part conveying the penalized regularization ideology in (7). That is,

$$J(\boldsymbol{a}) = \|\boldsymbol{y} - \boldsymbol{p}\|_2^2 + \rho(\|\boldsymbol{a}\|_1 - \|\boldsymbol{a}\|_{[k]}), \tag{8}$$

where $\boldsymbol{p} = f(\boldsymbol{d}, \boldsymbol{t}) = \boldsymbol{K}\boldsymbol{a}$, $k \in \mathbb{Z}_+$, and $\rho > 0$.

### 4.1. Algorithm

The execution of the proposed algorithm varies a little bit depending on the main objective and the starting point procedure choices (MO1, MO2, SP1, and SP2, see Subsections 4.1.1 and 4.1.2, respectively). However, the fundamental idea of LMBM-Kron$\ell_0$LS is to repeatedly apply LMBM for searching a dual vector $\boldsymbol{a}$ that minimizes the objective function (8) with different values of hyperparameters $k$ and $\rho$ given in Subsection 4.1.5. In order to

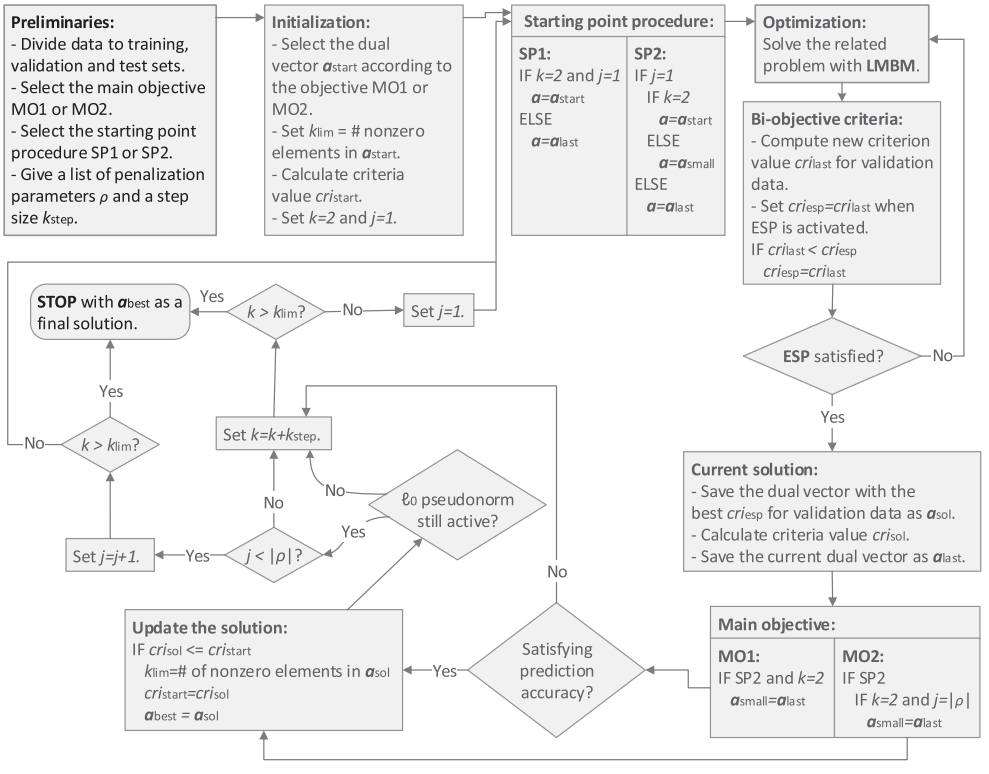


**Figure 2.** LMBM-Kron$\ell_0$LS method. Criteria values are calculated with formula (9) and the subscripts denote the dual vectors that were used, e.g. $cri_{\text{start}} = cri(\boldsymbol{a}_{\text{start}})$.

choose the best possible solution with respect to the selected main objective, a bi-objective optimization approach (see Subsection 4.1.3) and a separate validation data are used. In addition, the test for the desired accuracy in original LMBM (see Figure 1) is replaced with an early stopping procedure (ESP) to be described in Subsection 4.1.4. The flowchart of the algorithm is presented in Figure 2. We will next go trough the main parts of the algorithm.

### *4.1.1. Main objectives*

In order to solve the pairwise interaction affinity prediction problem, the user first needs to choose the main objective. The main objective determines whether the goals of good prediction performance and sparsity of the solution are equally important or whether one is considered more important than the other. The main objectives are referred to as MO1 and MO2 according to the following definitions.

MO1 Maintaining satisfactory prediction accuracy with respect to a non-sparse reference solution is a primary goal and adding sparsity is only a secondary goal.

MO2 The sparsity of a solution and the prediction performance are equally important goals.

Both MO1 and MO2 aim to find accurate predictions using as sparse solution vectors as possible. This is done by applying the bi-objective criterion (see Subsection 4.1.3) to the

solution returned by the optimization algorithm LMBM. In the case of MO2, we just minimize the criterion. In the case of MO1, the idea is to induce as sparse a solution as possible while staying at a close range from the reference solution's prediction accuracy. Thus, we add an extra constraint into the bi-objective criterion ensuring that the CI obtained with LMBM-Kron$\ell_0$LS (wrt. validation data) is not decreasing too much from the one obtained with the reference method. To increase the probability of finding such solutions, the non-sparse reference solution $\boldsymbol{a}_{\text{ref}}$ is used as an initial dual vector $\boldsymbol{a}_{\text{start}}$ in LMBM-Kron$\ell_0$LS with MO1. Even though the accuracy of the prediction and sparsity of the solution vector are conflicting objectives, it is possible to find a clearly better sparsity level with just a small deterioration in prediction accuracy. Since the reference method is not used (or needed) with MO2, the initial dual vector $\boldsymbol{a}_{\text{start}}$ is set equal to the vector of ones instead of the solution obtained with the reference method.

#### 4.1.2. Starting point procedure

In addition to main objective, the starting point procedure SP1 or SP2 needs to be selected. The starting point procedure determines the dual vectors used as starting points in the iterative optimization algorithm LMBM. Since the cardinality constraint of problem (6) is placed in the objective function as an exterior penalty function, the original problem – attached to the fixed sparsity desire $k$ – is transformed into a sequence of unconstrained problems (7) (or (8) in our specific case). This allows us to generate a sequence of infeasible solution points whose limit is an approximate solution of the original constrained problem (6) [88]. It is a common practice that when solving a sequence of problems with successively increasing penalty parameter $\rho$, the optimal point of a subproblem becomes the starting point for the following subproblem. This kind of approach is adopted in the first starting point inheritance procedure SP1, where the latest solution $\boldsymbol{a}_{\text{last}}$ is always the starting point of the next iteration. Note that this includes the cases where the sparsity parameter $k$ increases.

The second inheritance procedure for starting points SP2 is generated especially for the cases where SP1 fails. The fundamental idea behind SP2 is to push the learning process towards a sparse solution. In practice, the last dual vector obtained while solving optimization problems related to the smallest considered $k$ value $\boldsymbol{a}_{\text{small}}$ is used as a starting point each time after increasing the value of $k$. This is particularly useful under setting S4, where the training set is disjoint from both test and validation sets. In this case, it is unlikely that the model benefits from incorporating the maximum amount of training data information into the model (note that each dual variable relates to one training data sample). Instead, using the information of only a few most relevant dual variables may be profitable.

#### 4.1.3. Bi-objective criteria

In the best possible scenario, we could learn a light and sparse model without losing anything from the prediction accuracy of a dense model. In practice, there is, on the one hand, a good prediction performance of the model, but on the other hand, computational efficiency, needed storage space, and (at least to some extent) interpretability. In order to compare the 'goodness' of a dual vector $\boldsymbol{a}$ both in terms of prediction accuracy and acquired level of sparsity, a bi-objective criterion is needed. Our approach relies on a multiobjective framework (see, e.g. [63]) and searches for a dual vector whose prediction performance and sparsity level are as similar to the ideal values as possible. The prediction performance

is measured with CI and sparsity is measured as the number of nonzero elements ($nz$) in the dual vector, that is, a number of elements with an absolute value smaller than a given small threshold value, $10^{-4}$ in our case.

We calculate the bi-objective criteria as

$$cri\,(\boldsymbol{a}) = \frac{|\mathrm{CI}\,(\boldsymbol{a}) - \mathrm{CI}_{\mathrm{ideal}}|}{|\mathrm{CI}_{\mathrm{ideal}} - \mathrm{CI}_{\mathrm{worst}}|} + \frac{|nz\,(\boldsymbol{a}) - nz_{\mathrm{ideal}}|}{|nz_{\mathrm{ideal}} - nz_{\mathrm{worst}}|} \tag{9}$$

for a dual vector $\boldsymbol{a}$, where the ideal values correspond to a perfect predictor ($\mathrm{CI}_{\mathrm{ideal}} = 1$) and the sparsest possible dual vector ($nz_{\mathrm{ideal}} = 2$) whereas the worst values correspond to a random predictor ($\mathrm{CI}_{\mathrm{worst}} = 0.5$) and the least sparse dual vector ($nz_{\mathrm{worst}} = N$). The denominators are the ranges within which the values of CI and $nz$ can vary. This kind of scaling guarantees that the relative changes in both objectives have an equal impact on the obtained criteria value.

In addition to measuring the goodness of the dual vector $\boldsymbol{a}_{\mathrm{sol}}$ that is returned by the optimization algorithm LMBM (see the boxes 'Current solution' and 'Update the solution' in Figure 2), the bi-objective criterion value is used inside LMBM as a terminating condition instead of the prediction accuracy (see the diamond 'Desired accuracy?' in Figure 1) together with ESP.

#### 4.1.4. Early stopping procedure

For the optimization method LMBM, the objective function value with respect to training data is a non-increasing function of the iteration index. However, with respect to independent validation data, the measured error often decreases at first and then increases as the model starts to overfit. Therefore, it would be good to stop the training at the point with the smallest validation error. This is what ESP does. Standard ESP bases on the validation error (see, e.g. [15]). However, due to the bi-objective nature of our learning problem, we need to monitor the progress of the criteria value $cri_{\mathrm{esp}}$ (see, Figure 2 and Subsection 4.1.3) instead of the validation error in order to make ESP fit to our problem. The optimization process related to a pair of hyperparameters $(k, \rho)$ continues until the criteria value $cri_{\mathrm{esp}}$ has not improved on a sufficient number of iterations. In practise, we fix the number of iterations which indicates that a better solution is unlikely to emerge even if we continued the optimization procedure. Namely, if LMBM loops 200 iterations without any improvement on $cri_{\mathrm{esp}}$ value for the validation data, we stop. Naturally, one can modify the parameter values applied in this work.

Utilizing ESP helps us to obtain a good generalization performance. In addition, applying ESP is simple and reduces the computational demand as the training does not have to be completed. Despite 'terminating' the optimization process, it can be concluded that early stopping allows all relevant parameters to become the effective model parameters since the parameters' convergence speed is related to their importance [15,69,104].

The approach of inheriting previous solution vectors to the starting points for LMBM causes a need for another modification. If we activated ESP right from the first iteration of LMBM, we would most likely terminate the solution process way too early, especially with a larger penalization parameter $\rho$. The reason for this is that, with a high probability, a solution point of the previous learning problem (with hyperparameter pair $(k, \rho)$) is very near to a local minimum of the next learning problem (say with hyperparameter pair $(k, \rho')$). Therefore, better $cri_{\mathrm{esp}}$ values are not likely to be found at the neighbourhood of

this starting point, and that is why we activate ESP only after a fixed number of iterations in LMBM are completed. In practice, we force LMBM to loop 50 iterations before it starts to monitor the progress of ESP.

#### 4.1.5. Hyperparameter tuning

As mentioned before, there are two model parameters in the LMBM-Kron$\ell_0$LS method: $k$ refers to the number of the elements of the dual vector $\boldsymbol{a}$ permitted to have nonzero values while $\rho$ refers to the value of the penalization parameter representing the importance of satisfying the cardinality constraint related to the $\ell_0$ pseudonorm penalization term.

It is impossible to scan through the whole parameter space of all possible $k$ and $\rho$ values. Instead, representative subsets of possible parameter values are taken into consideration. Possible values of $k$ are determined by a step size $k_{\text{step}}$ given by the user. As our goal is to find as sparse a solution as possible with a satisfactory prediction accuracy, it makes sense to start from the strongest sparsity desire: $k = 2$. Hence, the outer iteration loop of the algorithm iterates through values $k = 2, 2 + k_{\text{step}}, 2 + 2k_{\text{step}}, 2 + 3k_{\text{step}}, \ldots$ until the terminating condition is satisfied. The choice of step size needs to be balanced between making sure that we do not jump over any promising value of $k$ and having a reasonable runtime. The larger the step size, the fewer distinct values of $k$ are explored. On the other hand, the smaller the step size, the longer running times can be expected, and sometimes a larger step size not only finds a solution faster but can also lead to a much better solution (in terms of criteria value).

In addition to the step size $k_{\text{step}}$, the user provides the list of alternative values of $\rho$ to the method as input. For each considered value of $k$, all possible values of $\rho$ are taken into account via the inner iteration loop. Symbol $|\rho|$ in Figure 2 represents the length of this list and $j = 1, \ldots, |\rho|$ is an index. According to [36] small $\rho$ values may lead to significant violations of the penalized cardinality constraint at the optimum of the problem and, on the other hand, large values may result in trivial solutions ($\boldsymbol{a} = \mathbf{0}$). Constructing an appropriate list of values for $\rho$ is a complex task since the targeted properties are partly contradictory. Nevertheless, adding only one new value to the $\rho$ list can have quite a significant effect on the obtained result if the insertion is done by changing either one of the $\rho$ list's extremes. There are a couple of logical guidelines to keep in mind when considering the insertion of the new smallest or largest $\rho$ list value:

(1) decreasing the largest value in the $\rho$ list makes it less likely to inherit forward a solution with a high sparsity level and poor prediction accuracy;
(2) decreasing the smallest value in the $\rho$ list makes it more likely to find a solution with satisfactory prediction accuracy.

The LMBM-Kron$\ell_0$LS algorithm either iterates through the entire list of given $\rho$ values or stops as a termination criterion is satisfied earlier. That is, either the $\ell_0$ pseudonorm is not active anymore – meaning that the desired maximum amount of nonzero elements in a dual vector is reached – or, in the case of MO1, the generalization performance is no longer good enough (see Figure 2). When we move out of the inner iteration loop, $k$ is increased with a given step size $k_{\text{step}}$. The execution of the algorithm stops if the current best solution contains fewer nonzero elements than the current or the next value of $k$ would

desire. In other words, if $k_{\text{lim}} < k$. Note that the upper limit $k_{\text{lim}}$ is not fixed but depends on the number of nonzero components in the best solution found so far.

### 4.2. Implementation

The developed LMBM-Kron$\ell_0$LS method is run as a Python program. However, the optimization routine LMBM operates in Fortran95. Therefore, we used an F2PY interface generator to incorporate the Fortran95 program inside the Python program. The most vital reason for applying Fortran95 implementation of LMBM is that LMBM consists mainly of simple arithmetic operations, which are executed most effectively in low-level libraries.

Following Airola et al. [1] we use GVT as a basic building block for developing a computationally efficient Kronecker product kernel method. The usage of GVT allows us to form the Kronecker product kernel matrix implicitly. This is a massive advantage since typically the runtime – and in many cases also the memory use – of kernel solvers grow at least quadratically with respect to the number of sample points. This makes the explicit definition of kernel matrix often not feasible in practice.

Computational costs for iterative Kronecker product kernel method training depend both on the cost of a single iteration and the number of iterations needed to reach a good solution. The cost of a single iteration is dominated by the computations of the subgradient which can be notably decreased by utilizing the combination of GVT and ESP. According to [1] (see also Theorem 3.4), the cost for each iteration of the KronRLS method is $\mathcal{O}(MN + QN)$ whilst training ridge regression in a naive way with the explicitly formed Kronecker product kernel using the standard solvers of the systems of linear equations would result in $\mathcal{O}(N^2)$. In the prediction phase, the dominating costs are the kernel matrix multiplications with the sparse dual coefficient vector.

Taking into account what is stated above, we modified LMBM (see Figure 1) by moving all the calculations involving data away from Fortran. We created external subroutines which take care of computing the objective function value $J(\boldsymbol{a})$ and determining an arbitrary element (subgradient) from the subdifferential $\boldsymbol{\xi} \in \partial J(\boldsymbol{a})$. The ESP is implemented similarly because the evaluation of the goodness of the current iteration point is performed with respect to the validation data.

## 5. Numerical experiments

We conducted numerical experiments to demonstrate that the proposed LMBM-Kron$\ell_0$LS method can produce sparse solutions to pairwise interaction affinity prediction problems under any of the four experimental settings while maintaining a competitive prediction performance compared with the state-of-the-art algorithms. We start this section by describing the data sets and continue by explaining how the hyperparameters of the method are selected and how they affect the method's behaviour.

The experiments are run on a computer equipped with Intel Core i7-11800H CPU (2.30 GHz). In order to assure compatibility with F2PY, we chose to use Python version 2.7.18. In the creation of an interface, GFortran was used as a Fortran compiler. As stated earlier, F2PY (Fortran to Python interface generator) is used to provide a connection between Python and Fortran languages. We utilized a standalone command line tool f2py

of the F2PY to build Python C/API extension modules making it possible to call Fortran95 external subroutines and to access Fortran95 module data (including allocatable arrays).

### 5.1. Data sets

Numerical experiments are run on three real data sets and two generated data sets. As we want to consider both the regression and classification formulation of the pairwise interaction affinity prediction problem, we do the experiments for two versions of each explored data set similar to Pahikkala et al. [74]: one with quantitative interaction affinities and another with binarized interaction levels. In the latter, the output gets value 1 if the drug and target interact and $-1$ if they do not.

The real data sets that are used have been introduced by Davis et al. [31], Metz et al. [62], and Merget et al. [61]. Cichonska et al. [28] have updated the Merget data to the form that we used. Each of the data sets consist of three matrices $X_D \in \mathbb{R}^{n_D \times n_D}$, $X_T \in \mathbb{R}^{n_T \times n_T}$ and $Y \in \mathbb{R}^{n_D \times n_T}$, where the first two are feature matrices for drugs and targets kinases, respectively, and the last one is the drug-target interaction affinity matrix. For Davis data, all pairwise interactions are known, whereas the other two are naturally sparse (interaction information is available only for a small subset of all possible drug-target pairs). The numbers of drugs, targets, and their known interactions are presented in Table 2 for each data

**Table 2.** Numbers of drug-target pairs whose interaction is known, unique drugs, unique targets, and numbers of positives and negatives in the binary case for the data sets used in the experimental study.

| | | | | | Binarized outputs | |
|---|---|---|---|---|---|---|
| Data set | Subset | # pairs | # drugs | # targets | pos. | neg. |
| Davis | All data | 30,056 | 68 | 442 | 1527 | 28,529 |
| | Training | 6544 | 41 | 266 | 394 | 6150 |
| | S1 | 2181/2181 | 41/41 | 266/266 | 134/131 | 2047/2050 |
| | S2 | 3608/3608 | 41/41 | 88/88 | 185/214 | 3423/3394 |
| | S3 | 3724/3458 | 14/13 | 266/266 | 134/162 | 3590/3296 |
| | S4 | 1232/1144 | 14/13 | 88/88 | 31/55 | 1201/1089 |
| Metz | All data | 93,356 | 1421 | 156 | 3200 | 90,156 |
| | Training | 20,052 | 826 | 94 | 612 | 19,440 |
| | S1 | 6665/6670 | 724/719 | 94/94 | 190/221 | 6475/6449 |
| | S2 | 10,239/12,618 | 708/741 | 31/31 | 324/438 | 9,915/12,180 |
| | S3 | 11,278/10,791 | 284/283 | 94/94 | 358/482 | 10,920/10,309 |
| | S4 | 3410/4028 | 241/252 | 31/30 | 114/177 | 3296/3851 |
| Merget | All data | 167,995 | 2967 | 226 | 9972 | 158,023 |
| | Training | 35,707 | 1719 | 136 | 1993 | 33,714 |
| | S1 | 11,842/11,852 | 1343/1334 | 136/131 | 616/631 | 11,226/11,221 |
| | S2 | 21,090/20,438 | 1293/1614 | 45/45 | 1154/1198 | 19,936/19,240 |
| | S3 | 20,587/18,727 | 588/592 | 136/134 | 1225/1175 | 19,362/17,552 |
| | S4 | 7312/6411 | 450/557 | 45/44 | 458/478 | 6854/5933 |
| Checker | Training | 2500 | 100 | 100 | 1253 | 1247 |
| | Test | 62,500 | 500 | 500 | 31,355 | 31,245 |
| | Validation | 2500 | 100 | 100 | 1245 | 1255 |
| Checker Noise | Training | 2500 | 100 | 100 | 1282 | 1218 |
| | Test | 62,500 | 500 | 500 | 31,491 | 31,009 |
| | Validation | 2500 | 100 | 100 | 1 275 | 1225 |

Note: Subsets S1–S4 refer to the experimental settings that are defined in Section 1 and their information is given individually for test/validation sets.

set. Note that the number of drug features (target features) is equal to the number of drugs (targets).

The numbers of positive and negative interactions in the binarized case are also presented in Table 2. The affinities are measured in different ways in different data sets. Thereby also the threshold values for the binary outputs vary: for Davis data, the drug and protein are considered interacting if $K_d \leq 30$ nM similarly to [74]. For Metz data, the interaction was considered as positive if $pK_i \geq 7.6$. For Merget data, a threshold value 7 was used as in [28].

In addition to the full data set characteristics, the same things are described for different subsets. All three real data sets are split into training, test, and validation sets in the same manner. The data sets are split into nine subsets as visualized in Figure 3: there is one training set, four test sets, and four validation sets. The subsets are created by splitting the drugs and targets into training, test, and validation drugs and targets, respectively. In the visual example, 'test targets' are those of indices {127, 166, 230, 232, 265, 375}, 'validation targets' are those of indices {54, 67, 155, 207, 340, 421, 441} and the rest are 'training targets' $X_t$. Same for drugs: 'test drugs' are those with indices {13, 18, 21, 28, 39}, 'validation drugs' are those with indices {15, 25, 38, 42, 47, 53} and the rest are 'training drugs' $X_d$. After that, the pairs are assigned to the subsets according to the definitions of the subsets. For example, pair $(\boldsymbol{d}_{16}, \boldsymbol{t}_{127})$ is in the test set for setting S2 since $\boldsymbol{d}_{16} \in X_d$ and $\boldsymbol{t}_{127}$ is a 'test target'. On the other hand, pair $(\boldsymbol{d}_{53}, \boldsymbol{t}_{441})$ is in the validation set for setting S4 since $\boldsymbol{d}_{53}$ is a 'validation drug' and $\boldsymbol{t}_{441}$ is a 'validation target'. The pairs in test and validation sets in setting S1 are such that both the drug $\boldsymbol{d}_i$ is a 'training drug' and the target $\boldsymbol{t}_t$ is a 'training target' but the exact pair $(\boldsymbol{d}_i, \boldsymbol{t}_j)$ is not included in the training data.

Further, we consider a sparse variant (labels are assigned only for 25% of all the possible pairs) of the Checkerboard simulation similar to [1]. In this simulation, the rows and columns of the Checkerboard table are considered as drugs and targets, respectively. Both

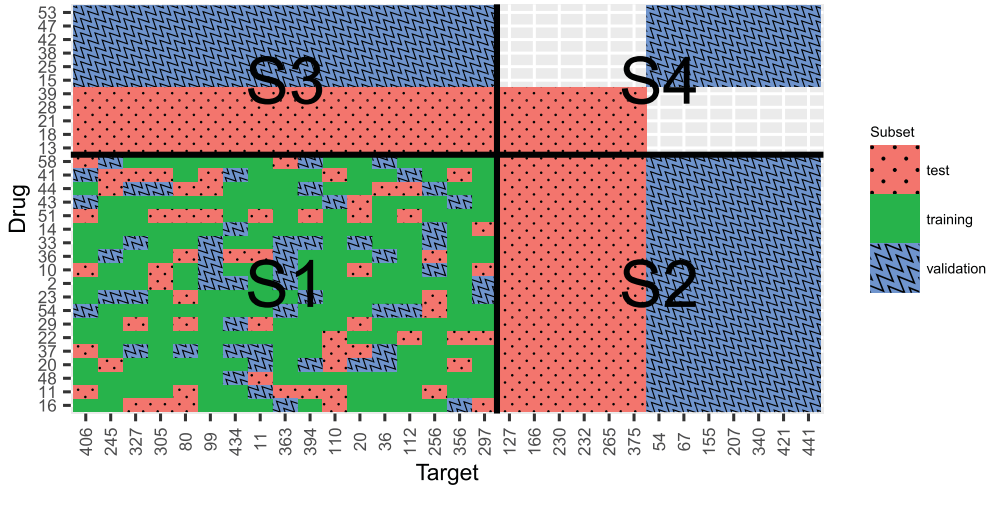


**Figure 3.** Visual demonstration of the splits with a complete data set of 30 drugs and 30 targets. Training set is mutual in every experimental setting S1–S4, but the test and validation sets are different in each setting.

the drugs and the targets have a single feature describing them, drawn from continuous uniform distribution $U(0, 5)$. In the case of binary interaction labels, the output assigned to a pair $(\boldsymbol{d}, \boldsymbol{t})$ is $+1$ whenever both $\lfloor d \rfloor$ and $\lfloor t \rfloor$ are either odd or even, and $-1$ when one of them is odd and the other even [1]. In order to create a regression problem with continuous interaction affinities, we simply multiplied the class label of a pair in the checkerboard classification problem with both feature values associated with the pair.

In the case of Checker data, there is no need for a train-validation-test splitting procedure. The simple reason is that we can generate training, validation, and test sets separately using different seed values. Since the amount of data creates no restrictions here, we decided to generate equally sized training and validation sets. The test set, in turn, was generated much larger than the training set as, in practice, we often have quite a small amount of data to train a model and a huge amount of possible pairs we need to create the predictions for.

In addition to testing how well the obtained model can handle nonlinearities, the Checker data allows us to examine the robustness of the obtained model against noise. We introduce random noise to the data by flipping the class / the output values' sign of each pair with 0.2 probability.

### 5.2. General setup

Every method used in this experiment uses Kronecker product kernel matrices. They were computed via Gaussian RBF kernel. Following Airola and Pahikkala [1] the applied values for the kernel width parameter $\gamma$ were $10^{-5}$ for the Metz data and 1.0 for the simulated Checker data. Data preprocessing was used with Davis and Merget data sets so that the resulting ranges of values of the drug and target similarity matrices and the interaction matrix were similar to the corresponding ranges in Metz data. After that, the value $\gamma = 10^{-5}$ was used also with these data.

In order to solve DTI prediction problems with MO1 of the proposed method, the reference model needs to be selected and its prediction accuracy needs to be measured. We used KronRLS [1] as a non-sparse reference model. Other options could be KronSVM [1] or the proposed method without regularization (i.e.$\rho = 0$) or early stopping. For both KronRLS and KronSVM (see, Equations (4) and (5), respectively), we selected an optimal hyperparameter $\lambda$ from a set $\{2^{-10}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{3}, 2^{4}, 2^{5}, 2^{10}\}$. This tuning of $\lambda$ was done separately for each data set and each setting. There was no need for a larger test grid since based on more exhaustive tests these hyperparameter values represent the main changes observed in the prediction performance. For every obtained solution vector, the prediction accuracy was determined against the validation data set. The solution vector with the best CI value for the validation data set was chosen to be used as a reference point in the proposed method's MO1.

Other things that can have a significant impact on the quality of the obtained results and hence need to be decided before running the method are the accepted decrease in CI in the case MO1, the step size $k_{\text{step}}$ and the possible values of $\rho$. In this work, we allowed a 5% decrease in CI, decided the step size to be 5% of the training data size for the real data sets and 1% for the generated Checker data sets, and ended up searching for an optimal penalty parameter $\rho$ from a set $\{0.1, 0.5, 1.0, 2.0, 4.0, 12.0, 50.0, 500.0\}$. In addition, the training-test-validation split of data affects the results. The drugs and targets were split into subsets

according to proportions 60%–20%–20%. The training/S1 pairs were onwards split into training, S1 test, and S1 validation subsets according to the same proportions while making sure that the definitions of the settings still hold.

For each data set and setting the learning problem is solved with both main objectives and both starting point procedures of the proposed method. We report the results of the starting point procedure that produced better criterion value.

### 5.3. Results

The results presented in Tables 3 and 4 represent the performances of the reference models KronRLS and KronSVM and the proposed method on independent test data. For each data set and setting the learning problem is solved with both main objectives and both starting point procedures. In Tables 3 and 4, we report the results with both main objectives. For different starting point procedures we report the results of the one that produced better criterion value as well as the procedure (SP) it was obtained. The sparsity percentage (Sparsity) is the fraction of zeroes in the dual solution. We point out that KronRLS always produces dense solutions (i.e. Sparsity $= 0$ for every data and setting).

It is clearly visible that S4 is the most difficult setting whereas in setting S1 the methods perform the best in terms of CI, as expected. In most cases, the performance in setting S2 is better than in S3; Metz data with continuous labels being the only exception.

With MO1, LMBM-Kron$\ell_0$LS finds much sparser solutions than KronRLS while allowing only a predetermined deviation from the CI obtained with KronRLS. The accepted decrease affects the results substantially. Sometimes better results could be obtained if the accepted decrease was smaller, but then there was also a risk of situations where the method does not find any acceptable solutions.

Even higher sparsity percentages are obtained with LMBM-Kron$\ell_0$LS with MO2 when the decrease in CI is not restricted. Often the CI is slightly lower than the corresponding CI with MO1. However, there are also exceptions in both directions: sometimes the prediction performance is much worse than with MO1 (e.g. Davis S4 with continuous labels) and

**Table 3.** Results for data sets with continuous labels.

| | | | LMBM-Kron$\ell_0$LS | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | KronRLS | MO1 | | | MO2 | | |
| Data set | Setting | CI | Sparsity | CI | SP | Sparsity | CI | SP |
| Davis | S1 | 0.847 | 73.9 | 0.818 | 1 | 89.3 | 0.803 | 1 |
| | S2 | 0.799 | 78.5 | 0.772 | 1 | 87.1 | 0.747 | 1 |
| | S3 | 0.760 | 73.5 | 0.732 | 1 | 88.6 | 0.687 | 1 |
| | S4 | 0.600 | 59.8 | 0.574 | 1 | 94.2 | 0.505 | 1 |
| Metz | S1 | 0.806 | 54.9 | 0.779 | 1 | 89.2 | 0.686 | 2 |
| | S2 | 0.690 | 83.6 | 0.659 | 1 | 93.9 | 0.636 | 2 |
| | S3 | 0.732 | 64.9 | 0.713 | 1 | 94.5 | 0.648 | 2 |
| | S4 | 0.648 | 89.6 | 0.636 | 2 | 94.1 | 0.623 | 2 |
| Merget | S1 | 0.808 | 39.0 | 0.793 | 1 | 94.0 | 0.637 | 1 |
| | S2 | 0.788 | 34.8 | 0.784 | 1 | 91.3 | 0.579 | 2 |
| | S3 | 0.687 | 64.6 | 0.660 | 1 | 92.1 | 0.614 | 2 |
| | S4 | 0.666 | 74.8 | 0.651 | 1 | 89.7 | 0.643 | 1 |
| Checker | | 0.823 | 77.4 | 0.829 | – | 81.4 | 0.826 | – |
| Checker Noise | | 0.658 | 71.3 | 0.663 | – | 80.8 | 0.665 | – |

**Table 4.** Results for binarized data sets.

| | | | KronSVM | | LMBM-Kron$\ell_0$LS | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | KronRLS | | | MO1 | | | MO2 | | |
| Data set | Setting | CI | Sparsity | CI | Sparsity | CI | SP | Sparsity | CI | SP |
| Davis | S1 | 0.927 | 0.0 | 0.925 | 76.0 | 0.896 | 1 | 93.7 | 0.880 | 1 |
| | S2 | 0.882 | 57.6 | 0.871 | 73.7 | 0.851 | 1 | 93.8 | 0.859 | 2 |
| | S3 | 0.841 | 57.6 | 0.842 | 83.0 | 0.828 | 2 | 94.4 | 0.798 | 1 |
| | S4 | 0.755 | 0.0 | 0.729 | 83.7 | 0.735 | 2 | 87.1 | 0.722 | 2 |
| Metz | S1 | 0.910 | 78.0 | 0.921 | 83.5 | 0.880 | 1 | 89.3 | 0.854 | 1 |
| | S2 | 0.858 | 0.1 | 0.859 | 89.1 | 0.829 | 2 | 89.7 | 0.833 | 1 |
| | S3 | 0.828 | 0.1 | 0.827 | 88.6 | 0.805 | 2 | 89.9 | 0.803 | 1 |
| | S4 | 0.777 | 0.1 | 0.772 | 82.7 | 0.762 | 1 | 89.9 | 0.728 | 1 |
| Merget | S1 | 0.873 | 0.0 | 0.880 | 49.9 | 0.860 | 1 | 88.5 | 0.805 | 2 |
| | S2 | 0.847 | 0.1 | 0.846 | 63.4 | 0.821 | 1 | 90.0 | 0.811 | 1 |
| | S3 | 0.792 | 72.9 | 0.806 | 93.9 | 0.775 | 1 | 87.8 | 0.762 | 1 |
| | S4 | 0.777 | 0.5 | 0.777 | 91.2 | 0.742 | 2 | 88.1 | 0.746 | 1 |
| Checker | | 0.956 | 81.9 | 0.972 | 88.8 | 0.949 | – | 87.4 | 0.951 | – |
| Checker Noise | | 0.734 | 5.2 | 0.733 | 86.8 | 0.731 | – | 89.5 | 0.736 | – |

sometimes the prediction performance is even better than with MO1 (e.g. Davis S2 with binary labels).

In addition to the choices of the main objective and starting point procedure, the data split has an impact on the results, both the state-of-the-art methods and the proposed method. A more comprehensive study would be needed for estimating the impact of the splits. Based on the results of the different data sets, it can be concluded that the results suggest that the proposed method LMBM-Kron$\ell_0$LS provides a valuable approach for enforcing sparsity on pairwise prediction problems, such as DTI affinity predictions.

## 6. Conclusions

In this paper, we have proposed a new nonsmooth optimization-based method LMBM-Kron$\ell_0$LS, which allows learning sparse models for predicting pairwise interaction affinities. The method aims for two contradictory goals: to obtain the best possible prediction accuracies while generating these predictions with a minimal number of variables. This means that we need to solve a bi-objective optimization problem.

The numerical experiments demonstrate that LMBM-Kron$\ell_0$LS can find sparse counterparts for a non-sparse reference solution without sacrificing too much from the prediction accuracy. In practice, the method allows the user to determine the limit value for acceptable prediction accuracy (main objective 1 (MO1)). If the user does not wish to set any limit, they can choose main objective 2 (MO2), which treats the sparsity of a solution and the prediction accuracy as equally important goals.

In the experiments, LMBM-Kron$\ell_0$LS found a sparse solution in every case. When the maximum decrease in CI is restricted to 5%, the obtained sparsity percentages are near 70% in the case of continuous and about 80% in the case of binary labels. In both cases, the average decreases in CI are about 2%. On the other hand, when the decrease in CI is not restricted (that is, with MO2), the obtained sparsity percentages are near 90% with both continuous and binary labels. In addition, the average decrease in CI is approximately 10% with continuous and only about 4% with binary labels.

In the end, we want to emphasize that, in contrast to most of the DTI prediction methods, LMBM-Kron$\ell_0$LS can predict pairwise interaction affinities at the state-of-the-art level even when they are modelled in a more realistic fashion as continuous values. The implementations for the LMBM-Kron$\ell_0$LS method, as well as for the F2PY interface, are made freely available under open source license.[2]

## Notes

1. The IDG-DREAM Drug-Kinase Binding Prediction Challenge provides up-to-date the most comprehensive benchmarking of machine learning-based drug-target prediction methods, comparing the results of 212 active challenge participants.
2. https://github.com/pavetsu14/LMBM-KronL0LS

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work has been supported by the Academy of Finland (A.A, grant 340140, 345805, N.K. grants 313269, 289500, T.P. grants 311273, 313266, 340182, and 345804)

## Notes on contributors

***Pauliina Paasivirta*** received her M.Sc. degree in applied mathematics at the University of Turku in 2018. At the time of writing this article, she was pursuing a Ph.D. degree in nonsmooth optimization and supervised machine learning models at the University of Turku. During her postgraduate studies she made research work also on neural network models and ways to handle missing values in data. Nowadays, she is working as a Consultant at Siili Solutions.

***Riikka Numminen*** received her M.Sc. degree in applied mathematics at the University of Turku in 2018. Currently she is a doctoral researcher at the Department of Computing at the University of Turku pursuing a Ph.D. degree in computer science. She is interested in the theoretical perspective of machine learning methods.

***Antti Airola*** received his M.Sc. in software engineering (2006) and D.Sc. in information and communication technology (2011) from University of Turku, Finland. He is an assistant professor at the Department of Computing at University of Turku. His research interests include machine learning and data analytics and their applications especially in the biomedical domain. He has co-authored over 80 peer-reviewed articles, won multiple international data science competitions, and received several research excellence awards such as the HATUTUS award for the best PhD thesis in the area of pattern recognition in Finland (2010 – 2011) and IEEE Computational Intelligence Society Outstanding Transactions on Fuzzy Systems Paper award (2015).

***Napsu Karmitsa*** (nee Haarala) is an Adjunct Professor in applied mathematics at the University of Turku (UTU), Finland, since 2011. She received her MSc degree in Organic Chemistry in 1998 and PhD degree (approved with honors) in Scientific Computing in 2004 both from the University of Jyväskylä, Finland. Currently, she is a senior research fellow at the Department of Computing at University of Turku. Dr. Karmitsa's research is focused on nonsmooth optimization (NSO) and data analysis. Special emphasis is given to nonconvex, global, and large-scale NSO, and applications of NSO in data mining and machine learning. In these research areas she has authored three books and several research papers.

***Tapio Pahikkala*** is a Professor of Machine Learning with the Department of computing, University of Turku, Finland, from which he also received his doctoral degree in 2008. His current research interests and work include theory and algorithmics of machine learning, data analysis, and artificial intelligence, as well as their applications on various different fields. He has led numerous research projects covering these areas, supervised twelve doctoral theses, held several positions of trust in academia, authored more than 150 peer-reviewed scientific articles and the methods developed by him have been used by the winning teams of several international scientific competitions/challenges.

## References

[1] A. Airola and T. Pahikkala, *Fast kronecker product kernel methods via generalized vec trick*, IEEE. Trans. Neural. Netw. Learn. Syst. 29 (2018), pp. 3374–3387.
[2] T.S. Arthanari and Y. Dodge, *Mathematical Programming in Statistics* Vol. 341, Wiley, New York, 1981.
[3] A. Astorino and A. Fuduli, *Nonsmooth optimization techniques for semisupervised classification*, IEEE. Trans. Pattern. Anal. Mach. Intell. 29 (2007), pp. 2135–2142.
[4] A. Astorino and A. Fuduli, *Support vector machine polyhedral separability in semisupervised learning*, J. Optim. Theory. Appl. 164 (2015), pp. 1039–1050.
[5] A. Astorino and M. Gaudioso, *Ellipsoidal separation for classification problems*, Optim. Methods Softw. 20 (2005), pp. 267–276.
[6] M. Bagherian, E. Sabeti, K. Wang, M.A. Sartor, Z. Nikolovska-Coleska, and K. Najarian, *Machine learning approaches and databases for prediction of drug-target interaction: a survey paper*, Brief. Bioinformatics. 22(1) (2021), pp. 247–269.
[7] A. Bagirov, M. Gaudioso, N. Karmitsa, M.M. Mäkelä, and S. Taheri, *Numerical Nonsmooth Optimization: State of the Art Algorithms*, Springer, 2020.
[8] A. Bagirov, N. Karmitsa, and M. Mäkelä, *Introduction to Nonsmooth Optimization: Theory Practice and Software*, Springer, 2014.
[9] A. Bagirov, N. Karmitsa, and S. Taheri, *Partitional Clustering Via Nonsmooth Optimization: Clustering Via Optimization*, Springer, 2020.
[10] A.M. Bagirov, S. Taheri, N. Karmitsa, N. Sultanova, and S. Asadi, *Robust piecewise linear l1-regression via nonsmooth DC optimization*, Optim. Methods Softw. 37 (2022), pp. 1289–1309.
[11] M. Bahi and M. Batouche, *Drug-Target Interaction Prediction in Drug Repositioning Based on Deep Semi-Supervised Learning*, Proceedings of the 6th international conference on computer intelligence and its application Vol. 522, 2018, pp. 302–313.
[12] J. Basilico and T. Hofmann, *Unifying Collaborative and Content-Based Filtering*, Proceedings of the 21st international conference on machine learning, 2004.
[13] A. Ben-Hur and W. Noble, *Kernel methods for predicting protein-protein interactions*, Bioinformatics 21 (2005), pp. 38–46.
[14] D. Bertsimas, A. King, and R. Mazumder, *Best subset selection via a modern optimization lens*, Ann. Stat. 44 (2016), pp. 813–852.
[15] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.
[16] K. Bleakley and Y. Yamanishi, *Supervised prediction of drug-target interactions using bipartite local models*, Bioinformatics 25 (2009), pp. 2397–2403.
[17] K. Buza and L. Peška, *Drug-target interaction prediction with bipartite local models and hubness-aware regression*, Neurocomputing 260 (2017), pp. 284–293.
[18] R. Byrd, J. Nocedal, and R. Schnabel, *Representations of quasi-newton matrices and their use in limited memory methods*, Math. Program. 63 (1994), pp. 129–156.
[19] D.S. Cao, S. Liu, Q.S. Xu, H.M. Lu , J.H. Huang, Q.N. Hu, and Y.Z. Liang, *Large-scale prediction of drug-target interactions using protein sequences and drug topological structures*, Anal. Chim. Acta. 752 (2012), pp. 1–10.
[20] R. Chen, X. Liu, S. Jin, J. Lin, and J. Liu, *Machine learning for drug-target interaction prediction*, Molecules 23 (2018), pp. 2208–2222.
[21] X. Chen, M.X. Liu, and G.Y. Yan, *Drug-target interaction prediction by random walk on the heterogeneous network*, Mol. Biosyst. 8 (2012), pp. 1970–1978.

[22] J. Chen, J. Wang, X. Wang, Y. Du, and H. Chang, *Predicting Drug Target Interactions Based on Gbdt*, Lecture Notes in Computer Science Vol. 10934, Springer, Cham, 2018.
[23] X. Chen, C. Yan, X. Zhang, X. Zhang, F. Dai, J. Yin, and Y. Zhang, *Drug-target interaction prediction: databases, web servers and computational models*, Brief Bioinform. 17 (2016), pp. 696–712.
[24] H. Chen and Z. Zhang, *A semi-supervised method for drug-target interaction prediction with consistency in networks*, PLoS ONE 8(5) (2013), pp. e62975. https://doi.org/10.1371/journal.pone.0062975.
[25] F. Cheng, C. Liu, J. Jiang, W. Lu, W. Li, G. Liu, W. Zhou, J. Huang, and Y. Tang, *Prediction of drug-target interactions and drug repositioning via network-based inference*, PLoS Comput. Biol. 8(5) (2012), pp. e1002503. https://doi.org/10.1371/journal.pcbi.1002503.
[26] F. Cheng, Y. Zhou, J. Li, W. Li, G. Liu, and Y. Tang, *Prediction of chemical-protein interactions: multitarget-qsar versus computational chemogenomic methods*, Mol. Biosyst. 8 (2012), pp. 2373–2384.
[27] A. Cichońska, B. Ravikumar, R. Allaway, F. Wan, S. Park, O. Isayev, S. Li, M. Mason, A. Lamb, Z. Tanoli, and M. Jeon, *Crowdsourced mapping of unexplored target space of kinase inhibitors*, Nat. Commun. 12 (2021). Article number 3307.
[28] A. Cichonska, T. Pahikkala, S. Szedmak, H. Julkunen, A. Airola, M. Heinonen, T. Aittokallio, and J. Rousu, *Learning with multiple pairwise kernels for drug bioactivity prediction*, Bioinformatics 34 (2018), pp. i509–i518.
[29] F.H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley-Interscience, New York, 1983.
[30] M.C. Cobanoglu, C. Liu, F. Hu, Z.N. Oltvai, and I. Bahar, *Predicting drug-target interactions using probabilistic matrix factorization*, J. Chem. Inf. Model. 53 (2013), pp. 3399–3409.
[31] M.I. Davis, J.P. Hunt, S. Herrgard, P. Ciceri, L.M. Wodicka, G. Pallares, M. Hocker, D.K. Treiber, and P.P. Zarrinkar, *Comprehensive analysis of kinase inhibitor selectivity*, Nat. Biotechnol. 29 (2011), pp. 1046–1051.
[32] H. Ding, I. Takigawa, H. Mamitsuka, and S. Zhu, *Similarity-basedmachine learning methods for predicting drug-target interactions: a brief review*, Brief. Bioinformatics. 15 (2013), pp. 734–747.
[33] A. Ezzat, P. Zhao, M. Wu, X.L. Li, and C.K. Kwoh, *Drug-target interaction prediction with graph regularized matrix factorization*, IEEE/ACM Trans. Comput. Biol. Bioinform 14 (2016), pp. 646–656.
[34] G. Fu, Y. Ding, A. Seal, B. Chen, Y. Sun, and E. Bolton, *Predicting drug target interactions using meta-path-based semantic network analysis*, BMC. Bioinformatics. 17 (2016). Article number 160.
[35] M. Gaudioso, G. Giallombardo, G. Miglionico, and E. Vocaturo, *Classification in the multiple instance learning framework via spherical separation*, Soft. Comput. 24 (2020), pp. 5071–5077.
[36] M. Gaudioso, E. Gorgone, and J. Hiriart-Urruty, *Feature selection in svm via polyhedral k-norm*, Optim. Lett. 14 (2020), pp. 19–36.
[37] M. Gönen, *Predicting drug-target interactions from chemical and genomic kernels using bayesian matrix factorization*, Bioinformatics 28 (2012), pp. 2304–2310.
[38] M. Gönen and G. Heller, *Concordance probability and discriminatory power in proportional hazards regression*, Biometrika 92 (2005), pp. 965–970.
[39] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, *A General Iterative Shrinkage and Thresholding Algorithm for Non-Convex Regularized Optimization Problems*, Proceedings of the 30th international conference on machine learning Vol. 28, 2013, pp. 37–45.
[40] J. Gotoh, A. Takeda, and K. Tono, *Dc formulations and algorithms for sparse optimization problems*, Math Program. Series B 169 (2018), pp. 141–176.
[41] A. Griewank and A. Rojas, *Treating Artificial Neural Net Training as a Nonsmooth Global Optimization Problem*, in *Machine Learning, Optimization, and Data Science. LOD 2019*. Lecture Notes in Computer Science(), Vol. 11943, G. Nicosia, P. Pardalos, R. Umeton, G. Giuffrida and V. Sciacca, eds., Springer, Cham, 2019.

[42] A. Griewank and A. Rojas, *Abs-Linear Learning by Mixed Binary Quadratic Optimization*, Proceedings of operations research, 2019, Springer Lecture Notes in Computer Science, 2020.
[43] M. Haarala, K. Miettinen, and M. Mäkelä, *New limited memory bundle method for large-scale nonsmooth optimization*, Optim. Methods Softw. 19 (2004), pp. 673–692.
[44] N. Haarala, K. Miettinen, and M. Mäkelä, *Globally convergent limited memory bundle method for large-scale nonsmooth optimization*, Math. Program. 109 (2007), pp. 181–205.
[45] T. Hofmann, B. Schölkopf, and A. Smola, *Kernel methods in machine learning*, Ann. Stat. 36 (2008), pp. 1171–1220.
[46] A. Hopkins, *Drug discovery: predicting promiscuity*, Nature 462 (2009), pp. 167–168.
[47] A. Hopkins and C. Groom, *The druggable genome*, Nat. Rev. Drug Discov 1 (2002), pp. 727–730.
[48] K. Joki, A. M. Bagirov, N. Karmitsa, M. M. Mäkelä, and S. Taheri, *Clusterwise support vector linear regression*, Eur. J. Oper. Res. 287 (2020), pp. 19–35.
[49] N. Karmitsa, A. Bagirov, and S. Taheri, *Clustering in large data sets with the limited memory bundle method*, Pattern. Recognit. 83 (2018), pp. 245–259.
[50] N. Karmitsa, S. Taheri, A. Bagirov, and P. Mäkinen, *Missing value imputation via clusterwise linear regression*, IEEE. Trans. Knowl. Data. Eng. 34 (2022), pp. 1889–1901.
[51] S. Kim, P. Thiessen, E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B.A. Shoemaker, and J. Wang, *Pubchem substance and compound databases*, Nucleic. Acids. Res. 44 (2016), pp. 1202–1213.
[52] Q. Kuang, Y. Li, Y. Wu, R. Li, Y. Dong, Y. Li, Q. Xiong, Z. Huang, and M. Li, *A kernel matrix dimension reduction method for predicting drug-target interaction*, Chemometr. Intell. Lab. Syst. 162 (2017), pp. 104–110.
[53] T. Laarhoven, S. Nabuurs, and E. Marchiori, *Gaussian interaction profile kernels for predicting drug-target interaction*, Bioinformatics 27 (2011), pp. 3036–3043.
[54] I. Lee, J. Keum, and H. Nam, *Deepconv-dti: prediction of drug-target interactions via deep learningwith convolution on protein sequences*, PLoS. Comput. Biol. 15(6) (2019), pp. e1007129. https://doi.org/10.1371/journal.pcbi.1007129.
[55] M. Lee, H. Kim, H. Joe, and H.G. Kim, *Multi-channel PINN: investigating scalable and transferable neural networks for drug discovery*, J. Cheminform. 11 (2019). Article number 46.
[56] Y. Luo, X. Zhao, J. Zhou, J. Yang, Y. Zhang, W. Kuang, J. Peng, L. Chen, and J. Zeng, *A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information*, Nat. Commun. 8 (2017). Article number 573.
[57] J.P. Mei, C.K. Kwoh, P. Yang, X.L. Li, and J. Zheng, *Drug-target interaction prediction by learning from local information and neighbors*, Bioinformatics 29 (2012), pp. 238–245.
[58] J.P. Mei, C.K. Kwoh, P. Yang, X.L. Li, and J. Zheng, *Globalized Bipartite Local Model for Drug-Target Interaction Prediction*, Proceedings of the 11th international workshop on data mining in bioinformatics 2012, pp. 8–14.
[59] D. Mendez, A. Gaulton, A. Bento, J. Chambers, M. De Veij, E. Félix, M.P. Magariños, J.F. Mosquera, P. Mutowo, M. Nowotka, and M. Gordillo-Marañón, *Chembl: Towards direct deposition of bioassay data*, Nucleic. Acids. Res. 47 (2019), pp. 930–940.
[60] X. Meng, H. Zhang, M. Mezei, and M. Cui, *Molecular docking: a powerful approach for structure-based drug discovery*, Curr. Comput. Aided. Drug. Des. 7 (2011), pp. 146–157.
[61] B. Merget, S. Turk, S. Eid, F. Rippmann, and S. Fulle, *Profiling prediction of kinase inhibitors: toward the virtual assay*, J. Med. Chem. 60 (2017), pp. 474–485.
[62] J.T Metz, E.F Johnson, N.B Soni, P.J Merta, L. Kifle, and P.J Hajduk, *Navigating the kinome*, Nat. Chem. Biol. 7 (2011), pp. 200–202.
[63] K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, 1999.
[64] R. Miyashiro and Y. Takano, *Mixed integer second-order cone programming formulations for variable selection in linear regression*, Eur. J. Oper. Res. 247 (2015), pp. 721–731.

[65] R. Miyashiro and Y. Takano, *Subset selection by mallow's cp: a mixed integer programming approach*, Expert. Syst. Appl. 42 (2015), pp. 325–331.
[66] N. Nagamine and Y. Sakakibara, *Statistical prediction of protein chemical interactions based on chemical structure and mass spectrometry data*, Bioinformatics 23 (2007), pp. 2004–2012.
[67] N. Nagamine, T. Shirakawa, Y. Minato, K. Torii, H. Kobayashi, M. Imoto, and Y. Sakakibara, *Integrating statistical predictions and experimental verifications for enhancing protein-chemical interaction predictions in virtual screening*, PLoS. Comput. Biol. 5(6) (2009), pp. e1000397. https://doi.org/10.1371/journal.pcbi.1000397.
[68] B.K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM J. Comput. 24 (1995), pp. 227–234.
[69] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer-Verlag, Berlin Heidelberg, 2001.
[70] H. Ohlsson, *Regularization for sparseness and smoothness-applications in system identification and signal processing*, Ph.D. diss., Linköping University, Department of Electrical Engineering, 2010.
[71] H. Öztürk, E. Ozkirimli, and A. Özgür, *Deepdta: Deep drug-target binding affinity prediction*, Bioinformatics 34(17) (2018), pp. i821–i829.
[72] T. Pahikkala, *Fast Gradient Computation for Learning with Tensor Product Kernels and Sparse Training Labels*, Lecture Notes in Computer Science Vol. 8621, Springer, Berlin Heidelberg, 2014, pp. 123–132.
[73] T. Pahikkala and A. Airola, *Rlscore: Regularized least-squares learners*, J. Mach. Learn. Res. 17 (2016), pp. 1–5.
[74] T. Pahikkala, A. Airola, S. Pietila, S. Shakyawar, A. Szwajda, J. Tang, and T. Aittokallio, *Toward more realistic drug-target interaction predictions*, Brief. Bioinformatics. 16 (2014), pp. 325–337.
[75] Y. Park and E. Marcotte, *Flaws in evaluation schemes for pair-input computational predictions*, Nat. Methods. 9 (2012), pp. 1134–1136.
[76] L. Perlman, A. Gottlieb, N. Atias, E. Ruppin, and R. Sharan, *Combining drug and gene similarity measures for drug-target elucidation*, J. Comput. Biol. 18 (2011), pp. 133–145.
[77] T. Poggio, V. Torre, and C. Koch, *Computational vision and regularization theory*, Nature 317 (1985), pp. 314–319.
[78] A.S. Rifaioglu, R. Cetin Atalay, D. Cansen Kahraman, T. Doğan, M. Martin, and V. Atalay, *MDeePred: novel multi-channel protein featurization for deep learning-based binding affinity prediction in drug discovery*, Bioinformatics 37 (2021), pp. 693–704.
[79] A. S. Rifaioglu, E. Nalbat, V. Atalay, M. J. Martin, R. Cetin-Atalay, and T. Doğan, *Deepscreen: high performance drug-target interaction prediction with convolutional neural networks using 2-d structural compound representations*, Chem. Sci. 11 (2020), pp. 2531–2557.
[80] B. Romera-Paredes and P. Torr, *An Embarrassingly Simple Approach to Zero-Shot Learning*, Proceedings of the 32nd international conference on machine learning Vol. 37, 2015, pp. 2152–2161.
[81] W. Roth, *On direct product matrices*, Bull. Amer. Math. Soc. 40 (1934), pp. 461–468.
[82] E. Sayers, T. Barrett, D.A. Benson, E. Bolton, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. DiCuccio, S. Federhen, and M. Feolo, *Database resources of the national center for biotechnology information*, Nucleic. Acids. Res. 40 (2012), pp. 13–25.
[83] B. Schölkopf, R. Herbrich, and A.J. Smola, *A Generalized Representer Theorem*, Proceedings of the 14th annual conference on computational learning theory and 5th european conference on computational learning theory, 2001, pp. 416–426.
[84] M. Schrynemackers, R. Küffner, and P. Geurts, *On protocols and measures for the validation of supervised methods for the inference of biological networks*, Front. Genet. 4 (2013). https://doi.org/10.3389/fgene.2013.00262.
[85] J.Y. Shi and S.M. Yiu, *Srp: A Concise Non-Parametric Similarity-Rank-Based Model for Predicting Drug-Target Interactions*, Proceedings of the 2015 IEEE international conference on bioinformatics and biomedicine, 2015, pp. 1636–1641.

[86] J.-Y. Shi, S.-M. Yiu, Y. Li, H.C.M. Leung, and F.Y.L. Chin, *Predicting drug-target interaction for new drugs using enhanced similarity measures and super-target clustering*, Methods 83 (2015), pp. 98–104.
[87] I. Steinwart, *Consistency of support vector machines and other regularized kernel classifiers*, IEEE Trans. Inf. Theory 51 (2005), pp. 128–142.
[88] W. Sun and Y. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer US, 2006.
[89] The UniProt Consortium, *Uniprot: A worldwide hub of protein knowledge*, Nucleic. Acids. Res. 47 (2019), pp. 506–515.
[90] T. Van Laarhoven and E. Marchiori, *Predicting drug-target interactions for new drug compounds using a weighted nearest neighbor profile*, PLoS. ONE. 8 (2013), pp. 238–245.
[91] M. Viljanen, A. Airola, and T. Pahikkala, *Generalized vec trick for fast learning of pairwise kernel models*, Mach. Learn. 111 (2022), pp. 543–573.
[92] J. Vlcek and L. Luksan, *Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization*, J. Optim. Theory. Appl. 111 (2001), pp. 407–430.
[93] W. Waegeman, T. Pahikkala, A. Airola, T. Salakoski, M. Stock, and B. De Baets, *A kernel-based framework for learning graded relations from data*, IEEE. Trans. Fuzzy. Syst. 20 (2012), pp. 1090–1101.
[94] F. Wan, L. Hong, A. Xiao, T. Jiang, J. Zeng, and J. Wren, *Neodti: Neural integration of neighbor information from a heterogeneous network for discovering new drug-target interactions*, Bioinformatics 35 (2019), pp. 104–111.
[95] Y. Wang, S. Bryant, T. Cheng, J. Wang, A. Gindulyte, B.A. Shoemaker, P.A. Thiessen, S. He, and J. Zhang, *Pubchem bioassay: 2017 update*, Nucleic. Acids. Res. 45 (2017), pp. D955–D963.
[96] Y. Wang and J. Zeng, *Predicting drug-target interactions using restricted boltzmann machines*, Bioinformatics 29 (2013), pp. 126–134.
[97] Y.-C. Wang, C.-H. Zhang, N.-Y. Deng, and Y. Wang, *Kernel-based data fusion improves the drug-protein interaction prediction*, Comput. Biol. Chem. 35 (2011), pp. 353–362.
[98] S. Whitebread, J. Hamon, D. Bojanic, and L. Urban, *Keynote review: in vitro safety pharmacology profiling: an essential tool for successful drug development*, Drug. Discov. Today. 10 (2005), pp. 1421–1433.
[99] Z. Xia, L.Y. Wu, X. Zhou, and S.T. Wong, *Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces*, BMC. Syst. Biol. 4 (2010). Article number S6.
[100] H. Yabuuchi, S. Niijima, H. Takematsu, T. Ida, T. Hirokawa, T. Hara, T. Ogawa, Y. Minowa, G. Tsujimoto, and Y. Okuno, *Analysis of multiple compound-protein interactions reveals novel bioactive molecules*, Mol. Syst. Biol. 1(7) (2011), pp. 472.
[101] Y. Yamanishi, *Chemogenomic approaches to infer drug-target interaction networks*, Methods Mol. Biol. 939 (2013), pp. 97–113.
[102] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa, *Prediction of drug-target interaction networks from the integration of chemical and genomic spaces*, Bioinformatics 24 (2008), pp. 232–240.
[103] Y. Yamanishi, M. Kotera, M. Kanehisa, and S. Goto, *Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework*, Bioinformatics 26 (2010), pp. 246–254.
[104] Y. Yao, L. Rosasco, and A. Caponnetto, *On early stopping in gradient descent learning*, Constr. Approx. 26 (2007), pp. 289–315.
[105] M. Yildirim, K.I. Goh, M. Cusick, A.L. Barabasi, and M. Vidal, *Drug-target network*, Nat. Biotechnol. 25 (2007), pp. 1119–1126.
[106] H. Yu, J. Chen, X. Xu, Y. Li, H. Zhao, Y. Fang, X. Li, W. Zhou, W. Wang, and Y. Wang, *A systematic prediction of multiple drug-target interactions from chemical, genomic, and pharmacological data*, PLoS. ONE. 7(5) (2012), pp. e37608. https://doi.org/10.1371/journal.pone.0037608.
[107] S. Zhao and S. Li, *Network-based relating pharmacological and genomic spaces for drug target identification*, PLoS. ONE. 5(7) (2010), pp. e11764. https://doi.org/10.1371/journal.pone.0011764.

[108] X. Zheng, H. Ding, H. Mamitsuka, and S. Zhu, *Collaborative Matrix Factorization with Multiple Similarities for Predicting Drug-Target Interactions* in Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, 2013, pp. 1025–1033.
[109] N. Zong, H. Kim, V. Ngo, O. Harismendy, and J. Wren, *Deep mining heterogeneous networks of biomedical linked data to predict novel drug-target associations*, Bioinformatics 33 (2017), pp. 2337–2344.