Oxytrees: Model Trees for Bipartite Learning

Pedro Ilídio[1,2], Felipe Kenji Nakano[1,2], Alireza Gharahighehi[1,2], Robbe D'hondt[1,2], Ricardo Cerri[3], Celine Vens[1,2]

[1] KU Leuven, Campus KULAK, Dept. of Public Health and Primary Care Etienne Sabbelaan 53, Kortrijk, 8500, Belgium
[2] Itec, imec research group at KU Leuven, Etienne Sabbelaan 51, Kortrijk, 8500, Belgium
[3] Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação Av. Trab. São Carlense, 13566-590, São Carlos, São Paulo, Brazil
{pedro.ilidio, felipekenji.nakano, alireza.gharahighehi, robbe.dhondt, celine.vens}@kuleuven.be, cerri@icmc.usp.br

Abstract

Bipartite learning is a machine learning task that aims to predict interactions between pairs of instances. It has been applied to various domains, including drug-target interactions, RNA-disease associations, and regulatory network inference. Despite being widely investigated, current methods still present drawbacks, as they are often designed for a specific application and thus do not generalize to other problems or present scalability issues. To address these challenges, we propose Oxytrees: proxy-based biclustering model trees. Oxytrees compress the interaction matrix into row- and column-wise proxy matrices, significantly reducing training time without compromising predictive performance. We also propose a new leaf-assignment algorithm that significantly reduces the time taken for prediction. Finally, Oxytrees employ linear models using the Kronecker product kernel in their leaves, resulting in shallower trees and thus even faster training. Using 15 datasets, we compared the predictive performance of ensembles of Oxytrees with that of the current state-of-the-art. We achieved up to 30-fold improvement in training times compared to state-of-the-art biclustering forests, while demonstrating competitive or superior performance in most evaluation settings, particularly in the inductive setting. Finally, we provide an intuitive Python API to access all datasets, methods and evaluation measures used in this work, thus enabling reproducible research in this field.

Code — https://github.com/pedroilidio/oxytrees2025
Python package —
https://github.com/pedroilidio/bipartite_learn
Extended version — https://arxiv.org/

1 Introduction

Over the last two decades, technological advancements have led to the daily generation of enormous volumes of data, encompassing social media interactions, e-commerce transactions, IoT sensor outputs, and genomic sequences. Such data are frequently presented in complex structures, which challenge the application of traditional machine learning algorithms. In this work, we focus on the problem of bipartite learning, a machine learning task that involves predicting interactions within a network where two distinct types of instances are interconnected. More specifically, bipartite learning is the task of modeling a function $(x_1,\ x_2) \mapsto y$ mapping a pair of objects x_1 and x_2 of different types to an output y characterizing their interaction. Often, these datasets can be represented by two feature matrices, X_1 and X_2, which describe instances in each dimension, and an interaction matrix, Y, that holds their corresponding output values.

Bipartite learning has been applied to several contexts. For instance, in bioinformatics, deep learning methods are employed for drug-target interaction (Huang et al. 2020; Lin et al. 2023; Bagherian et al. 2021; Liu et al. 2024), miRNA-disease association (Tian et al. 2024) and compound-protein interaction prediction (Tsubaki, Tomii, and Sese 2019). Similarly, recommender systems can be seen as a user-item interaction problem (Aggarwal et al. 2016).

Regardless of the application, validating all possible interactions in Y is unfeasible, e.g., it is impractical to verify all possible user-item or drug-target interactions. Consequently, many interactions are not directly observed but instead predicted using computational models trained on available data. It is often assumed that some of the training interactions labeled as negative are, in fact, unconfirmed positives, a setting commonly referred to as positive-unlabeled (PU) learning (Bekker and Davis 2020). Many proposed methods can only predict interactions among instances present in the training set, i.e., they are not applicable to new instances (Park and Marcotte 2012; Schrynemackers, Kueffner, and Geurts 2013; Pahikkala et al. 2015), and hence, are not inductive (Chapelle, Schölkopf, and Zien 2006). This limitation presents a significant challenge in the context of recommender systems, where it is referred to as the cold-start problem (Gharahighehi, Pliakos, and Vens 2022). In the context of drug-protein interaction prediction, it is crucial to generalize to new disease proteins. For that purpose, state-of-the-art techniques rely on deep learning to model the complex structure of the inputs (e.g., proteins as sequences, molecules as graphs) (Huang et al. 2020; Dehghan et al. 2023). As a result, they can only solve a specific type of interaction, severely compromising interpretability, and are computationally expensive, requiring large amounts of training data.

In this work, we investigate decision tree-based methods, due to their applicability to any type of interaction prob-

lem, interpretability, capacity to handle problems with few training data, and state-of-the-art performance on tabular input (Grinsztajn, Oyallon, and Varoquaux 2022; Costa and Pedreira 2023; Blockeel et al. 2023). Although promising, the current tree-based approaches to bipartite learning also pose significant scalability limitations. Furthermore, these decision trees fail to accurately represent sparse regions of the training space, since each leaf is a very simple model (the average label) of a small localized region. In this paper, we propose a novel method, named Oxytrees, as a solution to these limitations.

The core idea of Oxytrees is to aggregate the impurity of one of the dimensions of Y into small matrices we call *proxies*. The proxies are reused when evaluating multiple split candidates, aggregating only the remaining dimension when calculating the impurity of each candidate. Additionally, Oxytrees introduce a faster leaf-assignment procedure, for quicker inference of all interactions between large batches of instances; as well as model trees for bipartite learning. Model trees (Quinlan et al. 1992; Landwehr, Hall, and Frank 2005; Costa and Pedreira 2023) return a (simple) function in the leaf nodes, rather than a constant. This way, instance pairs arriving in the same leaf can still get very different predictions, allowing to more easily model complex relationships, and at the same time potentially allowing smaller tree sizes (with larger leaves[1]), hence reducing the tree construction time even further. Leaf models also enable trees to model functions that extend beyond the dense regions of the training set, where traditional decision trees struggle.

Two preliminary investigations of our research groups relate to this study. Alves and Cerri (2022) use a single tree with gradient boosting models in the leaves with large label imbalance. The complex leaf models amplified the scalability issues. We later (Ilídio, Alves, and Cerri 2024) introduced a preliminary version of the node-splitting procedure used by Oxytrees. In both studies, only a limited set of experiments was evaluated, and the performance improvements remained unclear.

For the present study, we built ensembles of Oxytrees and compared them to state-of-the-art techniques in bipartite learning. We selected methods applicable to a wide range of inductive interaction prediction tasks and compared them across 15 datasets. Furthermore, we investigate the PU setting in detail by randomly masking 25%, 50%, and 75% of the positive interactions. To the best of our knowledge, this represents the most extensive comparative study to date in the context of inductive bipartite learning, which helps delineate the state-of-the-art of this emerging paradigm. To assist future studies and allow reproducible research, we make all our algorithm implementations and preprocessed datasets publicly available, using an accessible and intuitive Python API based on Scikit-Learn (Pedregosa et al. 2011).

2 Formal Problem Definition

Consider two independent feature spaces $\mathcal{X}_1$ and $\mathcal{X}_2$, and an output space $\mathcal{Y}$. We address the problem of modeling a function $f\colon \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{Y}$ given only finite samples $X_1 \in \mathcal{X}_1^{n_1}$, $X_2 \in \mathcal{X}_2^{n_2}$, and $Y \in \mathcal{Y}^n$, with Y representing known outputs of f corresponding to dyads in $X_1 \times X_2$. We refer to this paradigm as bipartite learning. We call $\mathcal{X}_1$ and $\mathcal{X}_2$ the two domains or dimensions of the problem.

In our specific learning setting, we focus on tasks with structured input with $\mathcal{X}_1 \subseteq \mathbb{R}^{m_1}$ and $\mathcal{X}_2 \subseteq \mathbb{R}^{m_2}$. Therefore, X_1 and X_2 can be represented as matrices $X_1 = (X_1^{ij})$ and $X_2 = (X_2^{ij})$. We use indices as superscripts and reserve the subscripts for descriptors of the matrix or vector as a whole. We consider a single output, so $\mathcal{Y} \subseteq \mathbb{R}$ and an interaction matrix Y can be built to hold the annotated interactions as $Y^{ij} = f(X_1^i,\ X_2^j)$. We specifically focus on binary classification and therefore $\mathcal{Y} = \{0,\ 1\}$. Furthermore, we assume the PU setting (Bekker and Davis 2020), meaning that positive annotations are deemed confirmed, but negative annotations can be either ground-truth negatives or unannotated positives.

3 Related work

A handful of terms have been used in the literature to describe similar learning problems, such as interaction prediction (Schrynemackers et al. 2015; Pliakos and Vens 2019; Chen et al. 2018; Bagherian et al. 2020), link prediction (Lü and Zhou 2011; Zhou 2021), dyadic prediction (Menon and Elkan 2010; Pahikkala et al. 2014; Jin et al. 2017), and network inference (Park and Marcotte 2012; Pahikkala et al. 2015; Schrynemackers, Kueffner, and Geurts 2013; Pliakos and Vens 2019). Link prediction, dyadic prediction, and network inference do not specify the existence of feature spaces (X_1 and X_2 can be derived from Y). Furthermore, they allow $X_1 = X_2$, not resulting in a bipartite network, as is also the case for interaction prediction.

To build inductive models for bipartite learning, two common approaches reorganize the data to enable traditional algorithms. We call them data-based adaptations, and they are classified as global and local (Schrynemackers et al. 2015). Global methods transform X_1 and X_2 into a single input space, whereas local methods build separate models for each domain (Schrynemackers et al. 2015).

Unlike data-based approaches, some methods propose or modify machine learning estimators on a fundamental level to take maximum advantage of the bipartite learning setting. We call them estimator-based adaptations. Estimator-based methods from the recent literature are mainly based on deep learning, matrix factorization or decision trees. Deep learning methods provide end-to-end solutions in which data are processed in their raw format. For example, graphs (compounds) and sequences (proteins), together with their known interactions, are directly used as input to the neural networks in drug-target bipartite learning. Matrix factorization methods focus on decomposing the interaction matrix in a lower-dimensional latent representation, which is then used to infer interactions (Guo, Li, and Liang 2024; Chen and Wang 2022). Neighborhood-Regularized Logistic Matrix Factorization (NRLMF) carries neighborhoods to the latent space to enable inductiveness (Liu et al. 2016, 2017, 2020). Decision tree-based methods (Costa and Pedreira 2023; Block-

[1] *Oxytree* is the popular name of a genus of trees (*Paulownia sp.*) that are not too tall, grow very fast, and have very large leaves.

eel et al. 2023) are relatively new in bipartite learning. Predictive Bi-Clustering Trees (PBCT) (Pliakos, Geurts, and Vens 2018) are an extension of Predictive Clustering Trees (Blockeel and De Raedt 1998) that allows splits using features from both X_1 and X_2. In this way, a biclustering (i.e. two-dimensional partitioning) of the interaction matrix is obtained. PBCT was then extended to ensembles (Pliakos and Vens 2019), and further combined with a pre-training step that imputes the unreliable negative annotations using NRLMF. The resulting method was named BICTR, and showed state-of-the-art performance among the methods that are applicable to all bipartite learning problems (Pliakos and Vens 2020). Nonetheless, the method was only applied to the drug-target setting, and has scalability limitations both in induction and inference time.

4 The Proposed Method: Oxytrees

This study introduces Oxytrees, proxy-based model trees for bipartite learning. It combines a refined training algorithm with a new batch inference procedure and leaf-specific models to generate outputs.

4.1 Requirements for the Impurity Function

Let Y_{node} be a subset of Y reaching a given tree node. Decision trees use an impurity function $I(Y_{\text{node}})$ to select a split in each tree node (Breiman et al. 1984; Blockeel and De Raedt 1998). Oxytrees require (see theorem C2) that $I(Y_{\text{node}})$ can be expressed in terms of two arbitrary functions μ and ρ:

$$I_{\text{Oxytrees}}(Y_{\text{node}}) = \rho(\textstyle\sum_{ij} \mu(Y^{ij}_{\text{node}})). \tag{1}$$

In this study, we specifically used the variance of the vectorized interaction matrices: $I_{\text{Oxytrees}}(Y_{\text{node}}) = \sigma^2(\text{vec}(Y_{\text{node}}))$. More explicitly:

$$I_{\text{Oxytrees}}(Y_{\text{node}}) = \frac{\sum_{ij}(Y^{ij}_{\text{node}})^2}{n_{\text{node}}} - \left(\frac{\sum_{ij} Y^{ij}_{\text{node}}}{n_{\text{node}}}\right)^2, \tag{2}$$

which corresponds to

$$\mu(z) = \begin{bmatrix}1, & z, & z^2\end{bmatrix} \qquad \rho(z_1, z_2, z_3) = \frac{z_3}{z_1} - \frac{z_2^2}{z_1^2}. \tag{3}$$

The current state-of-the-art BICTR (Pliakos and Vens 2020) considers columns or rows of Y_{node} as multiple outputs, using $I(Y_{\text{node}})$ as the average of the variances of each column or row. This is incompatible with eq. (1) (see corollary C2.1) and therefore prohibits the proposed training optimization.

More intuitively, eq. (1) is equivalent to requiring $I(Y)$ to be invariant to the order in which dyads are presented during training (see theorem C1). This is a standard assumption, the only exceptions being BICTR-related methods and local methods (Schrynemackers, Kueffner, and Geurts 2013).

4.2 Using Proxies to Speed Up Training

The impurity format in eq. (1) enables a significant improvement in computational performance. Oxytrees start the split search in a node by building proxy matrices $\tilde{Y}_1$ and $\tilde{Y}_2$ that aggregate row- and column-wise impurities, respectively:

$$\tilde{Y}_1^i = \textstyle\sum_j \mu(Y^{ij}_{\text{node}}) \qquad \tilde{Y}_2^j = \textstyle\sum_i \mu(Y^{ij}_{\text{node}}). \tag{4}$$

Oxytrees then use $\tilde{Y}_1$ to evaluate all the horizontal split candidates in the node (that partition the rows of Y_{node}), and $\tilde{Y}_2$ to evaluate all the vertical split candidates in the node (that partition the columns). Finally, the split $s(Y_{\text{node}}) = \{Y_{\text{A}},\ Y_{\text{B}}\}$ with the largest decrease in impurity ΔI is chosen:

$$\Delta I = \tfrac{1}{n}(n_{\text{node}}\ I(Y_{\text{node}}) - n_{\text{A}}\ I(Y_{\text{A}}) - n_{\text{B}}\ I(Y_{\text{B}})). \tag{5}$$

The reasoning behind the computational improvement is as follows. A large number of different candidate splits is usually evaluated in each node. For a given horizontal (resp. vertical) split, it is much more efficient to calculate ΔI from the precomputed $\tilde{Y}_1$ (resp. $\tilde{Y}_2$) than from the original Y, as this bypasses aggregation across columns (resp. rows). As a consequence, the overall split search is faster using $\tilde{Y}_1$ and $\tilde{Y}_2$ for split evaluation, even if considering the extra time needed to build the proxy matrices. Precisely, building an Oxytree has the following computational complexity

$$\text{BuildOxytree} \in \Theta(n_1^2(\log(n_1) + m)), \tag{6}$$

in which m represents the number of features chosen to be evaluated at each node and $n_1 \propto n_2$ is the number of instances in each dimension. For PBCT we have

$$\text{BuildPBCT} \in \Theta(mn_1^2 \log(n_1)). \tag{7}$$

As a result, Oxytrees reduce the training time by a factor of $\log(n_1)$ if $m \in \Omega(\log(n_1))$ and by a factor of m otherwise, in comparison to PBCT-based methods (including the state-of-the-art, BICTR). A detailed derivation of eq. (6) and eq. (7) is presented in appendix D1, while an empirical confirmation experiment is explored in section 6.3. Please refer to algorithms FindBestSplitOxytree and FindBestSplitBICTR in appendix B for pseudo-codes on the split search procedures.

4.3 Efficient Batch Inference Procedure

Consider a set of test row instances $X_{1,\text{ test}}$ and a set of test column instances $X_{2,\text{ test}}$. The current state-of-the-art algorithm (Pliakos, Geurts, and Vens 2018) traverses the tree once for each pair in $X_{1,\text{ test}} \times X_{2,\text{ test}}$ to make predictions (see fig. 1). Instead, our inference procedure receives the two entire feature matrices and, at each tree node, uses the split rule to partition either $X_{1,\text{ test}}$ or $X_{2,\text{ test}}$, depending on whether the rule represents a horizontal or vertical split. Each partition is passed to the corresponding child node, while the other, nonpartitioned, X matrix is passed to both children. When the partitions $X_{1,\text{ leaf}}$ and $X_{2,\text{ leaf}}$ reach a leaf, we output the predictions for all the dyads in $X_{1,\text{ leaf}} \times X_{2,\text{ leaf}}$. The algorithm is presented in further detail as AssignLeavesOxytree in appendix B. Note that this procedure only evaluates each split rule once for each instance in the corresponding dimension, in contrast to the state-of-the-art BICTR, which performs redundant evaluations. The resulting improvement in complexity is as follows.

Let n_{test} and n_{train} respectively be the numbers of test and training instances in each dimension. The computational complexity of the Oxytrees batch inference procedure is given by $\Theta(n_{\text{test}}^2)$ if $n_{\text{test}} \in \Omega(n_{\text{train}})$, and by $\Theta(n_{\text{test}}^2 \log(n_{\text{train}}))$ otherwise. For BICTR, the complexity is $\Theta(n_{\text{test}}^2 \log(n_{\text{train}}))$ in all cases. The most common scenario is $n_{\text{test}} \in \Theta(n_{\text{train}}) \subset \Omega(n_{\text{train}})$ (e.g. separating a fraction of the training data for testing, or performing k-fold cross validation). Hence, the batch inference procedure is expected to run $\log(n_{\text{train}})$ times faster in most situations, given sufficiently large n_{train}. These results are derived thoroughly in appendix D2 and empirically verified in section 6.3.

4.4 Generalized Output Function

Accurate predictions require a careful trade-off between leaf sizes that are small enough to capture local patterns and large enough to generalize well. Leaf models are a self-adaptive middle ground in this trade-off, allowing leaves to be larger and faster to train while maintaining representation power. Oxytrees use Regularized Least Squares with the Kronecker product kernel (RLS-Kron) as the default leaf model, a bipartite adaptation of kernel Ridge regression proposed by Van Laarhoven, Nabuurs, and Marchiori (2011). We describe below how to obtain predictions with the RLS-Kron leaf models. An illustration is provided in fig. A1.

Let $X_{1,\,\text{leaf}}$, $X_{2,\,\text{leaf}}$ and Y_{leaf} denote the partition of the training set reaching a given leaf. $x_{1,\,\text{test}} \in \mathcal{X}_1$ and $x_{2,\,\text{test}} \in \mathcal{X}_2$ are then test instances that were assigned to that leaf. Given a context-specific similarity function $\phi_1 : \mathcal{X}_1^2 \to \mathbb{R}$, we represent $x_{1,\,\text{test}}$ as a homonymous vector $\phi_{1,\,\text{test}}$ so that $\phi^i_{1,\,\text{test}} = \phi_1(x_{1,\,\text{test}},\ X^i_{1,\,\text{leaf}})$. Instances $x_{2,\,\text{test}}$ are represented analogously, and multiple individual representations are gathered in matrices $\Phi_{1,\,\text{test}}$ and $\Phi_{2,\,\text{test}}$. Finally, the predicted outputs $\hat{Y}^{ij}$ between $\Phi^i_{1,\,\text{test}}$ and $\Phi^j_{2,\,\text{test}}$ are collectively obtained as

$$\hat{Y} = \Phi_{2,\,\text{test}}\ W_{\text{leaf}}\ \Phi^T_{1,\,\text{test}}. \tag{8}$$

The matrix W_{leaf} is composed of learned linear coefficients and is determined by

$$W_{\text{leaf}} = U_2\ [\Lambda \odot (U_2^T\ Y_{\text{leaf}}\ U_1)]\ U_1^T, \tag{9}$$

in which U_1 and U_2 are the matrices of eigenvectors of $\Phi_{1,\,\text{leaf}}$ and $\Phi_{2,\,\text{leaf}}$, respectively. $\Phi_{1,\,\text{leaf}}$ and $\Phi_{2,\,\text{leaf}}$, as suggested by their labels, are the symmetric similarity matrices corresponding to $X_{1,\,\text{leaf}}$ and $X_{2,\,\text{leaf}}$. The matrix Λ has the same dimensionality as Y_{leaf} and is defined in terms of the regularization parameter α and the vectors of eigenvalues λ_1 and λ_2 corresponding again to $\Phi_{1,\,\text{leaf}}$ and $\Phi_{2,\,\text{leaf}}$:

$$\Lambda^{ij} = (\alpha + \lambda_1^i \lambda_2^j)^{-1}. \tag{10}$$

5 Experimental Setup

5.1 Validation Procedure and Evaluation

There are two ways of splitting bipartite datasets into training and test sets to validate machine learning models: instance-wise and dyad-wise splits. We analyze results for all split types by first performing an instance-wise split in each dimension, resulting in four partitions of the dataset, and then performing a dyad-wise split on the partition with training rows and training columns. This results in the four test sets defined below and illustrated in fig. 1. L and T stand for learning (i.e., training) and test, respectively. D stands for dyads, referring to a dyad-wise split. The naming scheme is inspired by Schrynemackers, Kueffner, and Geurts (2013); Pliakos, Geurts, and Vens (2018).

- **Learning set (LD):** $(X_{1\text{L}},\ X_{2\text{L}},\ Y_{\text{LD}})$
- **Transductive test set (TD):** $(X_{1\text{L}},\ X_{2\text{L}},\ Y_{\text{TD}})$
- **Semi-inductive test sets:**
  - **Column-inductive test set (LT):** $(X_{1\text{L}},\ X_{2\text{T}},\ Y_{\text{LT}})$
  - **Row-inductive test set (TL):** $(X_{1\text{T}},\ X_{2\text{L}},\ Y_{\text{TL}})$
- **Inductive test set (TT):** $(X_{1\text{T}},\ X_{2\text{T}},\ Y_{\text{TT}})$

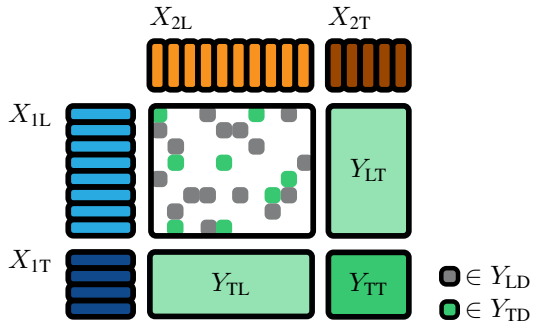

Figure 1: Diagram of the test (T) and training (i.e. learning, L) sets in the bipartite context (see section 5.1).

For cross-validation, each of the two domains was split into k folds, using label stratification to encourage consistent fold densities. Each (row fold)-(column fold) pair can represent a TT set, yielding k^2 possible TT sets in total. The corresponding semi-inductive test sets were also collected for each TT set, and the training set and transductive test set were then generated. The value of k was adjusted for each learning task, using more folds for smaller datasets (table 1).

A percentage of the positive annotations was masked (replacing ones with zeros) to generate Y_{TD}. We refer to this percentage as the positives masking percentage (PMP). All the missing annotations in the learning set (zeros that were not generated by masking) were also used as negative annotations for evaluation in the transductive setting. We explored four values for PMP: 0%, 25%, 50%, and 75%.

Regarding evaluation measures, we have employed the area under the precision and recall curve (AUPRC) and the area under the ROC curve (AUROC), following Schrynemackers, Kueffner, and Geurts (2013); Pliakos, Geurts, and Vens (2018); Pliakos and Vens (2020); Liu et al. (2022). Moreover, we have employed the Friedman test, followed by the post-hoc Nemenyi test, as suggested by Demšar (2006). We report results for the semi-inductive settings altogether, averaging ranks of corresponding TL and LT scores.

5.2 Datasets

Table 1 describes the datasets used in this work. As can be seen, we have employed a total of 15 datasets from several

biological domains. All features correspond to similarities measured between instances of the same input space. Feature matrices in the test sets represent similarities with the training instances (avoiding data leakage).

Name	Domain	Size	Folds	Density
DPI-N	Drug-NR	26×54	4×4	6.4%
TE-piRNA	TE-piRNA	60×93	4×4	3.9%
lnc-mi	LncRNA-miRNA	44×218	3×3	10%
DPI-G	Drug-GPCR	95×223	3×3	3.0%
Davis	Inhibitor-Kinase	68×442	3×3	5.0%
lnc-D	LncRNA-Disease	156×241	3×3	6.6%
lnc2Cancer	LncRNA-Cancer	367×106	3×3	7.3%
DPI-I	Drug-Ion channel	204×210	2×2	3.5%
MiRNA-D	MiRNA-Disease	462×252	2×2	11%
ERN	Gene-TF	1164×154	2×2	1.8%
SRN	Gene-TF	1821×113	2×2	1.8%
NPInter	LncRNA-Protein	586×446	2×2	18%
DPI-E	Drug-Enzyme	664×445	2×2	1.0%
KIBA	Inhibitor-Kinase	2111×229	2×2	20%
miRTarBase	MiRNA-Gene	1873×415	2×2	7.1%

Table 1: Summary of the datasets used in this study. Density refers to the percentage of positive annotations in each dataset. NR = nuclear receptor; TE = transposable element; TF = transcription factor.

DPI-N, DPI-G, DPI-I, DPI-E, SRN, ERN, Davis and KIBA are widely employed in the literature (Schrynemackers et al. 2015; Pliakos and Vens 2019; Huang et al. 2020). We have preprocessed NPInter and mirTarBase in our preliminary work (Ilídio, Alves, and Cerri 2024). We have also collected five more datasets, TE-Pirna, LncRNA-miRNA, lncRNA-disease, lncRNA-cancer, and miRNA-disease, which, to the best of our knowledge, have never been used as benchmarks for interaction prediction in general. Lnc2Cancer contains lncRNA-cancer interactions (Gao et al. 2021)[2]. The datasets LncRNA-disease, LncRNA-miRNA and miRNA-disease were made available in (Sheng et al. 2023), where the authors collected data from several repositories. TE-piRNA contains interaction between transposable elements (TEs) and piwi-RNAs (piRNAs) (Freire, dos Santos, and Cerri 2024).

5.3 Comparison Methods

We present in table 2 a list of the comparison methods used in this work. We selected prominent methods from the literature that i) were able to infer interactions for new instances, and ii) were applicable across different bipartite learning problems. We provide more details about them in appendix E. As previously mentioned, the most common deep learning approaches do not meet criterion (ii). We thus include a baseline multi-layer perceptron (MLP) (Hinton 1990) under the data-based global adaptation (Schrynemackers, Kueffner, and Geurts 2013) to still represent this model category. Following previous literature (Huang et al. 2020; Dehghan et al. 2023), we performed random undersampling of negative annotations, resulting in balanced training datasets for the MLP baseline.

[2]http://bio-bigdata.hrbmu.edu.cn/lnc2cancer/

Method	Base technique	Adapt.	TF	Ex	EB	In
MLP	Deep learning	Global				✓
RLS-avg	Linear regression (LR)	Local		✓		
RLS-Kron	LR + Kronecker product	—		✓	✓	✓
BLMNII	LR + Nearest Neighbors	Local		✓		✓
NRLMF	Matrix factorization	—			✓	✓
WkNNIR	Nearest-Neighbors	Local	✓			✓
BICTR	Decision trees	—		✓	✓	✓
Oxytrees	Decision trees	—	✓	✓	✓	✓

Table 2: Summary of methods evaluated in our experiments. References: MLP (Hinton 1990); RLS-{avg, Kron} (Van Laarhoven, Nabuurs, and Marchiori 2011); BLMNII (Mei et al. 2013); NRLMF (Liu et al. 2016); WkNNIR (Liu et al. 2022); BICTR (Pliakos and Vens 2020). See appendix E for hyperparameters of each method. Legend: Adapt. = bipartite adaptation strategy; TF = hyperparameter tuning-free; Ex = explainable; EB = Estimator-based adaptation (as opposed to data-based); In = inductive.

Our proposal does not enforce any particular ensembling technique. Following BICTR (Pliakos and Vens 2020), we used extremely randomized (Geurts, Ernst, and Wehenkel 2006) Oxytrees for this study.

Regarding hyperparameter tuning, a nested 2 by 2 cross-validation procedure was utilized, using AUPRC to select the best hyperparameter combination for each outer fold. The specific hyperparameter sets considered for each method are also presented in appendix E.

6 Results and Discussion

6.1 Oxytrees Against Previous Proposals

Inductive AUPRC and AUROC results for 0% PMP are presented in fig. 2. Similar results were found for other PMP values (fig. F1). Average values for AUROC and AUPRC in each dataset are presented by table F2 and table F1, respectively. Oxytrees had the best average ranking for all but one of the inductive settings analyzed, and no statistically significant differences were found comparing Oxytrees to the state-of-the-art. We also see that Oxytrees perform consistently better than RLS-Kron alone, indicating that the tree structure provides considerable improvement. Figure F2 presents results for the semi-inductive setting. Oxytrees, BICTR and RLS-Kron were the top performers, and the differences between their scores were not statistically significant in any case. For the transductive setting (fig. F3), NRLMF, BICTR and RLS-Kron had the best overall scores.

The scores in the inductive (tables F1 and F2) and semi-inductive (tables F5-F8) settings are considerably lower compared to the transductive setting (tables F3 and F4), which highlights the general difficulty of considering new instances.

On the other hand, NRLMF was one of the worst performers in the inductive setting compared to the other models. This shows that the original promising results of NRLMF

Figure 2: Comparisons of AUPRC and AUROC of the proposed Oxytrees against previous methods for bipartite learning. The inductive setting is analysed, for 0% positives masking percent (PMP) (see section 5.1). Further results in fig. F1.

in the transductive setting (found by Liu et al. (2016, 2017, 2020) and reproduced by us in fig. F3). do not translate into the inductive setting. This motivates the previously proposed combination of NRLMF and biclustering forests (Pliakos and Vens 2020), restricting NRLMF to the transductive step of matrix completion and delegating the inductive modeling to the forest. Furthermore, WkNNIR was not able to outperform BICTR in our experiments with 15 datasets, contrary to what was observed for drug-target interaction prediction (Liu et al. 2022).

To analyze the robustness of each model to missing positives, we compare their scores in different PMP settings relative to the setting with PMP = 0% (fig. F4). As expected, the performances of all models decrease when we increase the PMP, with very few exceptions that we attribute to the intrinsic noise of these values. BLMNII and RLS-avg were the models with the smaller performance drop in general, suggesting that the simplicity of local linear models makes them robust to missing labels. The results for MLP were inconsistent, highlighting its sensitivity to the learning setting and hyperparameter choices.

6.2 Ablation Study

We compare ensembles of Oxytrees with and without the minimum leaf size constraint and different leaf models, as indicated in table 3. We also test the interaction matrix reconstruction (YR) technique proposed by Pliakos and Vens (2020). The results are presented in fig. 3.

Removing the leaf model (and using the average label as usual) significantly degraded the performance, even when using YR. Using YR while enforcing minimum leaf dimensions of 5 by 5 was the worst overall strategy.

6.3 Empirical Complexity Analysis

In this analysis, we compared the training and inference times of Oxytrees and BICTR, for different data sizes. For each size n_1, we generated three n_1 by n_1 matrices with pseudo-random values to serve as X_1, X_2, and Y. We then built a single tree of the BICTR and Oxytrees ensembles for each n_1. We also built an Oxytree until each leaf contained a single instance (Oxytree[Deep], table 3), to isolate the effect of the larger leaves. To measure inference times, we built a completely random tree for each n_1 and used it twice to assign a leaf to each instance: through the BICTR procedure, and through the Oxytrees procedure (section 4.3). The results are shown in fig. 4. The estimated complexities follow the theoretical expectations of section 4.2 (with $m = n_1$) and section 4.3. Plotting times of Oxytrees against BICTR (fig. F5), we measured that Oxytrees trained 35.49 ± 0.75 times faster and predicted 9.98 ± 0.84 times faster ($p < 4 \cdot 10^{-5}$, Wald).

Model	Leaf model	Min. 5 by 5 leaves	Y rec.
Oxytrees	RLS-Kron	✓	
[Mean]	—	✓	
[Logistic]	Logistic reg.	✓	
[Deep]	—		
[YR]	RLS-Kron	✓	✓
[Mean, YR]	—	✓	✓
[Deep, YR]	—		✓

Table 3: Summary of method modifications compared in the ablation study. Interaction matrix reconstruction (Y rec.) was performed with NRLMF (Liu et al. 2016), following Pliakos and Vens (2020). For the "Deep" models, the trees were grown until all dyads in a node share the same label. For the "Mean" models, the mean label was used for the prediction, as usual for decision trees.

6.4 Dependency on Leaf Size

We assess the effect of setting minimum leaf dimensions on the predictive performance of biclustering forests, comparing Oxytrees with RLS-Kron leaf models, Oxytrees without leaf models and BICTR. The results for the inductive setting are shown in fig. 5. Similar results were found for the other settings (appendix F). The performance of forests without leaf models drops when the leaves are enforced to be larger, especially for leaf dimensions larger than 20 by 20. On the other hand, Oxytrees with leaf models keep their performance stable even with larger leaves, with a slight score increase up until 20 by 20 leaves. Therefore, adding RLS-Kron as leaf models enables shallower trees.

6.5 Dependency on the Number of Trees

We perform a bootstrapping procedure to evaluate how the performance of each biclustering forest varies with the number of trees. For each forest, we started by building a total of 200 trees. Then, we progressively added the outputs of each tree, calculating the resulting score in each step. We repeated the summing 50 times, with a random tree order

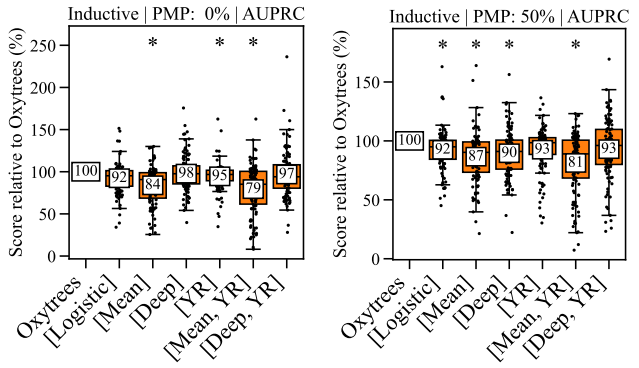

Figure 3: Performance comparison of ensembles of Oxytrees using different components (table 3). To compare across datasets, the scores are divided by the score of the main model (Oxytrees). Each point represents a cross-validation fold, and the mean value is presented in the white boxes. Asterisks indicate significance in comparison to Oxytrees ($p < 0.05$, Wilcoxon signed-rank), averaging folds for each dataset. Remaining results are presented in fig. F6.

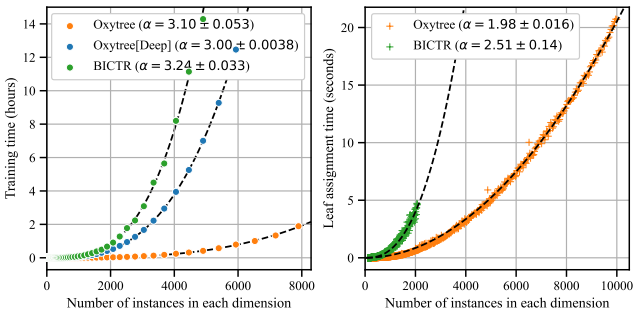

Figure 4: Empirical complexity analysis. Single trees were applied to artificial datasets of different dimensions (section 6.3). We used the last 10% of the points of each curve to approximate the asymptotic complexity as $\Theta(n^\alpha)$. α ($\pm$ standard dev.) is estimated as the slope of the linear regression in the log-log space. Further results in fig. F5.

each time. From the plot of scores vs. number of trees, we used linear interpolation to estimate the number of trees required to achieve 98% of the final score of 200 trees. The measurements for PMP 0% are reported in fig. 6. Oxytrees required significantly fewer trees ($42\% \pm 3\%$) relative to BICTR ($p < 2 \cdot 10^{-4}$, Wilcoxon).

7 Conclusion and Future Work

We proposed an ensemble of proxy-based biclustering model trees: Oxytrees. Oxytrees had competitive results among state-of-the-art techniques for inductive bipartite learning. Against the state-of-the-art biclustering forest, BICTR, Oxytrees were 35.49 ± 0.75 times faster to train ($p < 4 \cdot 10^{-8}$) and predicted 9.98 ± 0.84 times faster ($p < 4 \cdot 10^{-5}$). They also required up to 50% less trees to achieve 98% of the performance of 200 trees ($p < 2 \cdot 10^{-4}$). The model comparisons were mostly consistent across different masking percentages of positive annotations. We provide datasets and efficient implementations of the algorithms compared in this study, in an accessible API based on Scikit-Learn. As future work, we would like to explore the potential of Oxytrees for regressive bipartite learning, such as drug-target affinity prediction, and compare our results with deep learning techniques that work on the raw unstructured data (such as protein sequences and molecular structures).

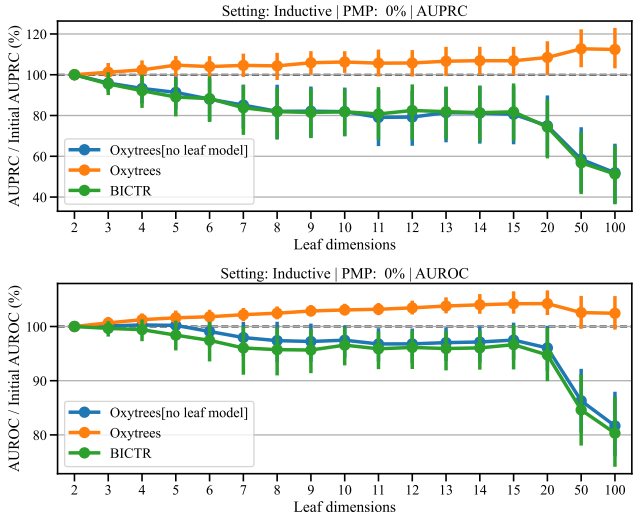

Figure 5: Performance of biclustering forests as a function of minimum leaf dimensions, relative to the initial performance with 2 by 2 leaves. Values in the x axis represent the minimum number of instances in each dimension. Markers indicate mean and standard deviation over 15 datasets.

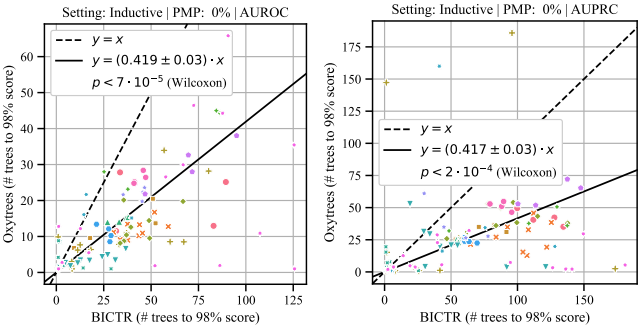

Figure 6: Comparison between Oxytrees and BICTR in terms of the expected number of trees to achieve 98% of the score of 200 trees. Each fold is plotted, with different styles for each dataset. Remaining results can be found in fig. F8.

Acknowledgements

The work received funding from the Flemish Government (AI Research Program), VOEWIAI02. The authors thank FWO, 11A7U26N, 1235924N and 1S38025N. R.C. thanks the Brazilian National Council for Scientific and Technological Development (CNPq), grant number 300934/2025-4, and the São Paulo Research Foundation (FAPESP), Brasil,

Process Number #2022/02981-8.

References

Aggarwal, C. C.; et al. 2016. Recommender systems, volume 1. Springer.

Alves, A. H. R.; and Cerri, R. 2022. A Two-step Model for Drug-Target Interaction Prediction with Predictive Bi-Clustering Trees and XGBoost. In 2022 International Joint Conference on Neural Networks (IJCNN), 1–8.

Bagherian, M.; Sabeti, E.; Wang, K.; Sartor, M. A.; Nikolovska-Coleska, Z.; and Najarian, K. 2020. Machine learning approaches and databases for prediction of drug–target interaction: a survey paper. Briefings in Bioinformatics, 22(1): 247–269.

Bagherian, M.; Sabeti, E.; Wang, K.; Sartor, M. A.; Nikolovska-Coleska, Z.; and Najarian, K. 2021. Machine learning approaches and databases for prediction of drug–target interaction: a survey paper. Briefings in bioinformatics, 22(1): 247–269.

Bekker, J.; and Davis, J. 2020. Learning from positive and unlabeled data: A survey. Machine Learning, 109(4): 719–760.

Blockeel, H.; and De Raedt, L. 1998. Top-down induction of first-order logical decision trees. Artificial intelligence, 101(1-2): 285–297.

Blockeel, H.; Devos, L.; Frénay, B.; Nanfack, G.; and Nijssen, S. 2023. Decision Trees: From Efficient Prediction to Responsible AI. Frontiers in Artificial Intelligence, 6: 1124553.

Breiman, L.; Friedman, J.; Olshen, R. A.; and Ston, C. J. 1984. Classification and regression trees. New York: Routledge. ISBN 978-1-315-13947-0.

Chapelle, O.; Schölkopf, B.; and Zien, A., eds. 2006. Semi-supervised learning. Adaptive computation and machine learning. Cambridge: MIT Press. ISBN 978-0-262-03358-9.

Chen, R.; Liu, X.; Jin, S.; Lin, J.; and Liu, J. 2018. Machine learning for drug-target interaction prediction. Molecules, 23(9): 2208.

Chen, Z.; and Wang, S. 2022. A review on matrix completion for recommender systems. Knowledge and Information Systems, 64(1): 1–34.

Costa, V. G.; and Pedreira, C. E. 2023. Recent advances in decision trees: An updated survey. Artificial Intelligence Review, 56(5): 4765–4800.

Dehghan, A.; Razzaghi, P.; Abbasi, K.; and Gharaghani, S. 2023. TripletMultiDTI: multimodal representation learning in drug-target interaction prediction with triplet loss function. Expert Systems with Applications, 120754.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7: 1–30. Publisher: JMLR. org.

Freire, H. O.; dos Santos, R. A. C.; and Cerri, R. 2024. Transposable Elements and piRNAs interaction prediction with Predictive Bi-Clustering Trees. bioRxiv.

Gao, Y.; Shang, S.; Guo, S.; Li, X.; Zhou, H.; Liu, H.; Sun, Y.; Wang, J.; Wang, P.; Zhi, H.; et al. 2021. Lnc2Cancer 3.0: an updated resource for experimentally supported lncRNA/-circRNA cancer associations and web tools based on RNA-seq and scRNA-seq data. Nucleic acids research, 49(D1): D1251–D1258.

Geurts, P.; Ernst, D.; and Wehenkel, L. 2006. Extremely randomized trees. Machine Learning, 63: 3–42. Publisher: Springer.

Gharahighehi, A.; Pliakos, K.; and Vens, C. 2022. Addressing the Cold-Start Problem in Collaborative Filtering Through Positive-Unlabeled Learning and Multi-Target Prediction. IEEE Access, 10: 117189–117198. Conference Name: IEEE Access.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems, 35: 507–520.

Guo, Y.-T.; Li, Q.-Q.; and Liang, C.-S. 2024. The rise of nonnegative matrix factorization: algorithms and applications. Information Systems, 102379.

Hinton, G. E. 1990. Connectionist learning procedures. In Machine learning, 555–610. Elsevier.

Huang, K.; Xiao, C.; Glass, L. M.; and Sun, J. 2020. MolTrans: molecular interaction transformer for drug–target interaction prediction. Bioinformatics, 37(6): 830–836.

Ilídio, P.; Alves, A.; and Cerri, R. 2024. Fast Bipartite Forests for Semi-supervised Interaction Prediction. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, SAC '24, 979–986. New York, NY, USA: Association for Computing Machinery. ISBN 9798400702433.

Jin, B.; Yang, H.; Xiao, C.; Zhang, P.; Wei, X.; and Wang, F. 2017. Multitask Dyadic Prediction and Its Application in Prediction of Adverse Drug-Drug Interaction. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).

Landwehr, N.; Hall, M.; and Frank, E. 2005. Logistic Model Trees. Machine Learning, 59(1): 161–205.

Lin, X.; Dai, L.; Zhou, Y.; Yu, Z.-G.; Zhang, W.; Shi, J.-Y.; Cao, D.-S.; Zeng, L.; Chen, H.; Song, B.; et al. 2023. Comprehensive evaluation of deep and graph learning on drug–drug interactions prediction. Briefings in Bioinformatics, 24(4): bbad235.

Liu, B.; Pliakos, K.; Vens, C.; and Tsoumakas, G. 2022. Drug-target interaction prediction via an ensemble of weighted nearest neighbors with interaction recovery. Applied Intelligence, 52(4): 3705–3727.

Liu, H.; Ren, G.; Chen, H.; Liu, Q.; Yang, Y.; and Zhao, Q. 2020. Predicting lncRNA–miRNA interactions based on logistic matrix factorization with neighborhood regularized. Knowledge-Based Systems, 191: 105261. Publisher: Elsevier.

Liu, H.; Ren, G.; Hu, H.; Zhang, L.; Ai, H.; Zhang, W.; and Zhao, Q. 2017. LPI-NRLMF: lncRNA-protein interaction prediction by neighborhood regularized logistic matrix factorization. Oncotarget, 8(61): 103975. Publisher: Impact Journals, LLC.

Liu, Y.; Wu, M.; Miao, C.; Zhao, P.; and Li, X.-L. 2016. Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. PLoS Computational Biology, 12(2): e1004760.

Liu, Z.; Chen, Q.; Lan, W.; Lu, H.; and Zhang, S. 2024. SSLDTI: A novel method for drug-target interaction prediction based on self-supervised learning. Artificial Intelligence in Medicine, 149: 102778.

Lü, L.; and Zhou, T. 2011. Link prediction in complex networks: a survey. Physica A: statistical mechanics and its applications, 390(6): 1150–1170.

Mei, J.-P.; Kwoh, C.-K.; Yang, P.; Li, X.-L.; and Zheng, J. 2013. Drug–target interaction prediction by learning from local information and neighbors. Bioinformatics, 29(2): 238–245.

Menon, A. K.; and Elkan, C. 2010. A Log-Linear Model with Latent Features for Dyadic Prediction. In 2010 IEEE International Conference on Data Mining, 364–373. Sydney, Australia: IEEE. ISBN 978-1-4244-9131-5.

Pahikkala, T.; Airola, A.; Pietilä, S.; Shakyawar, S.; Szwajda, A.; Tang, J.; and Aittokallio, T. 2015. Toward more realistic drug-target interaction predictions. Briefings in Bioinformatics, 16(2): 325–337.

Pahikkala, T.; Stock, M.; Airola, A.; Aittokallio, T.; De Baets, B.; and Waegeman, W. 2014. A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. In Calders, T.; Esposito, F.; Hüllermeier, E.; and Meo, R., eds., Machine learning and knowledge discovery in databases, 517–532. Berlin: Springer. ISBN 978-3-662-44851-9.

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

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: machine learning in Python. Journal of Machine Learning Research, 12(85): 2825–2830.

Pliakos, K.; Geurts, P.; and Vens, C. 2018. Global multi-output decision trees for interaction prediction. Machine Learning, 107: 1257–1281. Publisher: Springer.

Pliakos, K.; and Vens, C. 2019. Network inference with ensembles of bi-clustering trees. BMC Bioinformatics, 20: 1–12. Publisher: Springer.

Pliakos, K.; and Vens, C. 2020. Drug-target interaction prediction with tree-ensemble learning and output space reconstruction. BMC Bioinformatics, 21: 1–11. Publisher: Springer.

Quinlan, J. R.; et al. 1992. Learning with continuous classes. In 5th Australian joint conference on artificial intelligence, volume 92, 343–348. World Scientific.

Schrynemackers, M.; Kueffner, R.; and Geurts, P. 2013. On protocols and measures for the validation of supervised methods for the inference of biological networks. Frontiers in Genetics, 4. Publisher: Frontiers.

Schrynemackers, M.; Wehenkel, L.; Babu, M. M.; and Geurts, P. 2015. Classifying pairs with trees for supervised biological network inference. Molecular BioSystems, 11(8): 2116–2125.

Sheng, N.; Huang, L.; Lu, Y.; Wang, H.; Yang, L.; Gao, L.; Xie, X.; Fu, Y.; and Wang, Y. 2023. Data resources and computational methods for lncRNA-disease association prediction. Computers in Biology and Medicine, 153: 106527.

Tian, Z.; Han, C.; Xu, L.; Teng, Z.; and Song, W. 2024. MGCNSS: miRNA–disease association prediction with multi-layer graph convolution and distance-based negative sample selection strategy. Briefings in Bioinformatics, 25(3): bbae168.

Tsubaki, M.; Tomii, K.; and Sese, J. 2019. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. Bioinformatics, 35(2): 309–318.

Van Laarhoven, T.; Nabuurs, S. B.; and Marchiori, E. 2011. Gaussian interaction profile kernels for predicting drug–target interaction. Bioinformatics, 27(21): 3036–3043. Publisher: Oxford University Press.

Zhou, T. 2021. Progresses and challenges in link prediction. Iscience, 24(11). Publisher: Elsevier.