Internal Report

Deliverable 2.3: Final implementation of ClowdFlows or TextFlows, including the final HINMINE workflow, demonstrated on nontrivial knowledge discovery benchmark problems

Jozef Stefan Institute Version 1 FINAL

Abstract: We present the implementation of the HinMIne methodology in the online workflow development and sharing platform ClowdFlows. We present the workflow, implementing the methodology, and all the widgets that comprise the workflow. We also explain the format of the workflow's inputs and outputs. The Hinmine methodology was also used in a further paper on targeted end-to-end knowledge graph decomposition where it was applied to a biological data set.
Abstract of the paper: Knowledge graphs are networks with annotated nodes and edges, representing different relations between the network nodes. Learning from such graphs is becoming increasingly important as numerous real-life systems can be represented as knowledge graphs, where properties of selected types of nodes or edges are learned. This paper presents a fully autonomous approach to targeted knowledge graph decomposition, advancing the state-of-the-art HINMINE network decomposition methodology. In this methodology, weighted edges between the nodes of a selected node type are constructed via different typed triplets, each connecting two nodes of the same type through an intermediary node of a different type. The final product of such a decomposition is a weighted homogeneous network of the selected node type. HINMINE is advanced by reformulating the supervised network decomposition problem as a combinatorial optimization problem, and by solving it by a differential evolution approach. The proposed approach is tested on node classification tasks on two real-life knowledge graphs. The experimental results demonstrate that the proposed end-to-end learning approach is much faster and as accurate as the exhaustive search approach.

Document administrative information	
Project acronym:	HinLife
Project number:	J7-7303
Deliverable number:	D2.3
Deliverable full title:	Final implementation of ClowdFlows or TextFlows, including the final HINMINE workflow, demonstrated on nontrivial knowledge discovery benchmark problems
Document identifier:	HinLife -del-D2.3-clowdflows-benchmark
Lead partner short name:	JSI
Report version:	final
Report preparation date:	31/12/2018
Lead author:	Jan Kralj
Co-authors:	Blaž Škrlj
Status:	Final

Introduction

The deliverable 2.3 of WP2 includes the final implementation of ClowdFlows, including the final version of the HinMine workflow. We implemented all the functions, used in our experiments with the HinMine methodology, in the ClowdFlows platform. The methodology is fully available as an online workflow, and the code for the workflow is also publicly available on a GitHub repository.

ClowdFlows workflow overview

The resulting workflow is shown in Figure 1. The workflow begins by loading a data set encoded as a `.gml` file. The GML (Graph Modeling Language) [1] is a text format that allows for easy representation of network data. For the HinMine methodology, the input requires that each node in the network is of a given type. In the online example, available on the address http://clowdflows.org/workflow/11019/, the methodology is run on a subset of the IMDB data set containing nodes of type `person` and nodes of type `movie`. One node type (the base node type for the HinMine methodology) must be labeled, and if more than one label is applicable for a node, the labels must be separated by a `---` separator. An example of a node looks as follows:

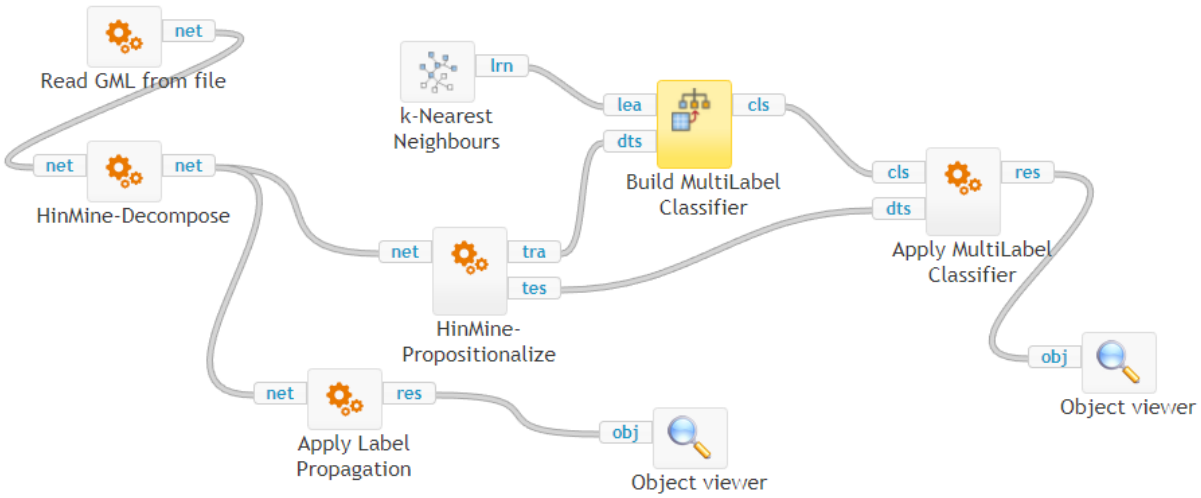
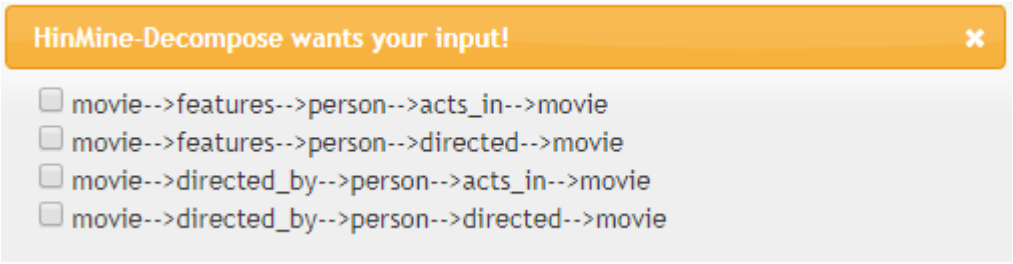


Figure 1: Overview of the HinMine methodology as a workflow in the Clowdflows platform.

```
node [
  id 6336
  label "movie_303"
  labels "Action---Adventure---Western"
  type "movie"
  name "movie_the-quick-and-the-dead"
]
```

Actions, performed by the workflow widgets

The methodology loads the GML file in the widget *Read GML from file*, where it identifies the base node type (the node type that is labeled) and training instances (all instances that are labeled). The loaded network is passed as the variable `net` to the widget *HinMine-Decompose* where network decomposition is applied. This is an interactive widget that, when first run, provides all possible decompositions of the input network. The widget discovers all possible decomposition paths and allows the user to choose which decompositions to perform. After performing the decompositions, the widget returns the decomposed network in the variable `net`. Figure 2 shows the possible decompositions of the online example. After decomposition, the methodology has two options:



1. If we classify the data set with label propagation, we can use the *Apply Label Propagation* widget. This widget performs label propagation on the network and returns the results as a numpy [2] array (variable `res`).
2. If we classify the data using propositionalization, the *HinMine-Propositionalize* widget performs the network propositionalization described in [3]. This widget constructs the feature vectors for the labeled (training, variable `tra`) and unlabeled (test, variable `tes`) nodes separately. In this way, the classifier can be trained using the *Build MultiLabel Classifier* widget on the training set. In classifier construction using the *Build MultiLabel Classifier*, any learner capable of predicting labels on data sets containing numeric values can be used as the inuput variable `lea`. In the online example, we use the *k*-nearest neighbours classifier. Finally, the induced classifier is applied to the test set and a numpy array is returned as a result in the variable `res`.

Both options above result in the workflow returning a numpy array. The array's columns represent the labels of the data set and the rows represent the unlabeled nodes. Each row contains results of label propagation applied to the given unlabeled node. The result is a vector of values between 0 and 1, and the higher the value, the more likely it is that the label is applicable to the given node.

References

[1] Himsolt, M. (1997). GML: A Portable Graph File Format. Universität Passau.
[2] Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The Numpy Array: A Structure for Effi-cient Numerical Computation. *Computing in Science & Engineering*, *13* (2), 22 ⊏ 30.
[3] Kralj, J., Robnik-Sikonja, M., Lavrač, N.: HINMINE: Heterogeneous information network mining with information retrieval heuristics. *Journal of Intelligent Information Systems (2017)* 1–33
[4] Skrlj, B., Kralj, J., Vavpetič, A., Lavrač, N.: Community-based semantic subgroup discovery. In: *Proceedings of New Frontiers in Mining Complex Patterns Workshop* (2018) 182– 196
[5] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Eppig, J. T., et al. (2000). Gene Ontology: Tool for the Unication of Biology. *Nature genetics*, *25* (1), 25.
[6] Orchard, S., Ammari, M., Aranda, B., Breuza, L., Briganti, L., BroackesCarter, F., Campbell, N.H., Chavali, G., Chen, C., del Toro, N., Duesbury, M., Dumousseau, M., Galeota, E., Hinz, U., Iannuccelli, M., Jagannathan, S., Jimenez, R., Khadake, J., Lagreid, A., Licata, L., Lovering, R.C., Meldal, B., Melidoni, A.N., Milagros, M., Peluso, D., Perfetto, L., Porras, P., Raghunath, A., Ricard-Blum, S., Roechert, B., Stutz, A., Tognolli, M., van Roey, K., Cesareni, G., Hermjakob, H.: The MIntAct project–IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Research 42(Database issue)* (January 2014) D358–63

Use of HinMine on a real-world data set

The HinMine algorithm was already used on an two from the biological domain.

In the first use case, the HinMine algorithm was used as part of an algorithm that performs targeted end-to-end graph decomposition. The algorithm is described in detail below. The data set used used is the recently introduced epigenetics knowledge graph [4], where proteins, genes and other biological entities are connected with different relations. Further, different protein-protein interactions are annotated with ground-truth edge weights corresponding to reliability of interactions. We annotate each protein in the network with the corresponding Gene Ontology [5] terms, associated with their functions, which is achieved as follows. Functional annotations are obtained from the Intact database [6]. We sort annotations by frequency and select 100 of the most common terms, which correspond to 100 classes being predicted. The classification goal for this dataset is thus protein function prediction. The network consists of 2,204 nodes and 2,772 edges. Of the nodes, 456 belong to the target type. Details of this use case are available in the paper, attached to this document.

In the second use case, the HinMine algorithm was used in DDeMON (Dynamic Deep learning from temporal Multiplex Ontology-annotated Networks), an approach for scalable, systems-level inference of function annotation using time-dependent multiscale biological information. We demonstrate the use of the proposed method on recently introduced experimental expression data, discovering multiple novel biomarkers, valuable for understanding pathogen response, as well as preliminary disease detection. The implementation and testing of DdeMON are described in the technical report attached to this document.

Targeted End-to-end Knowledge Graph Decomposition

Blaž Škrlj[1,2], Jan Kralj[2], and Nada Lavrač[2,3]

[1] Jožef Stefan Int. Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia
[2] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[3] University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia
{blaz.skrlj,jan.kralj,nada.lavrac}@ijs.si

Abstract. Knowledge graphs are networks with annotated nodes and edges, representing different relations between the network nodes. Learning from such graphs is becoming increasingly important as numerous real-life systems can be represented as knowledge graphs, where properties of selected types of nodes or edges are learned. This paper presents a fully autonomous approach to targeted knowledge graph decomposition, advancing the state-of-the-art HINMINE network decomposition methodology. In this methodology, weighted edges between the nodes of a selected node type are constructed via different typed triplets, each connecting two nodes of the same type through an intermediary node of a different type. The final product of such a decomposition is a weighted homogeneous network of the selected node type. HINMINE is advanced by reformulating the supervised network decomposition problem as a combinatorial optimization problem, and by solving it by a differential evolution approach. The proposed approach is tested on node classification tasks on two real-life knowledge graphs. The experimental results demonstrate that the proposed end-to-end learning approach is much faster and as accurate as the exhaustive search approach.

Keywords: Knowledge graphs, network analysis, supervised machine learning

1 Introduction

Network analysis was established as an independent research discipline in the early eighties [1]. While it initially addressed the analysis of homogeneous information networks, analysis of *heterogeneous information networks* [2] has recently gained a lot of attention. In contrast with standard (homogeneous) information networks, heterogeneous information networks describe heterogeneous types of entities and different types of relations. Examples of heterogeneous networks are e.g., *biological networks* that contain different entity types such as species, genes, Gene Ontology annotations [3], proteins, metabolites, etc. There are diverse types of links between such mixed biological entities; for example, genes can belong to species, encode proteins, be annotated by an ontology annotation, and so on.

In this work we focus on *knowledge graphs* [4], i.e. heterogeneous information networks with annotated nodes and annotated (relation-labeled) edges, like in the following examples from a biological knowledge graph:

$$protein_1 \xrightarrow{\text{interactsWith}} protein_2$$

$$protein_1 \xrightarrow{\text{annotatedWith}} function_A$$

$$protein_2 \xrightarrow{\text{annotatedWith}} function_B$$

Some of the common tasks on knowledge graphs [5] include relation extraction, triplet classification, entity recognition as well as entity classification, where the task is to assign correct labels to a specific set of nodes in a knowledge graph. For example, if the nodes are different proteins, one of the possible goals is to classify the proteins based on their function. Take another example, where the nodes are different movies, and the goal is to classify the movies to different genres. The problem arises when different relations between individual movies and other movies, actors or directors are considered (e.g., $actor_1 \xrightarrow{\text{actsIn}} movie_5 \xrightarrow{\text{directedBy}} director_4$).

The problem of node classification can be formally stated as follows. Let $G = (N, E, R, C)$ represent a knowledge graph, where N is a set of nodes (of possibly different types), E a set of edges, R a set of relations on edges and C the set of classes assigned to nodes. The goal is to construct a mapping $\theta : N \rightarrow C$, which assigns the most probable class to a node to be classified.

One of the possible approaches to node classification is through *network decomposition* [6, 7], where a knowledge graph is aggregated into a homogeneous network consisting of a single target node type, and the information derived from the graph is encoded in the form of weighted edges of the aggregated homogeneous network. Different down-stream machine learning algorithms can then be applied to the aggregated network to perform node classification [8, 7]. One of the remaining open questions is whether homogeneous network construction can be performed in an automated, end-to-end manner. The main contributions of this work include a novel, interpretable end-to-end knowledge graph decomposition approach based on state-of-the-art network decomposition approaches, an improved parallel decomposition algorithm and the inclusion of ground-truth edge information between the target nodes into the network decomposition process.

This paper is structured as follows. The next section provides a brief overview of state-of-the-art approaches to learning from different types of networks. We continue the discussion by describing the optimization procedure used in this work. Finally, the proposed approach is presented along with the results of its experimental evaluation.

2 Background and related work

There are different methodologies for the analysis and construction of knowledge graphs, which can be broadly divided into methods that focus on entities (nodes)

and those focused to relations (links). This section only addresses methods for node classification that significantly influenced our approach to knowledge graph decomposition. Given that our approach leverages combinatorial optimization, a brief outline of relevant optimization methods is provided as well.

2.1 Background: Network decomposition

In this section, we first explain the concept of network decomposition and then introduce the HINMINE approach, a recently developed algorithm for network decomposition.

On network decomposition. One of the main problems with node embedding approaches is that they do not take into account different types of relations. Further, in most cases only a single node type is supported. To address this task, a graph first needs to be transformed into a suitable input for down-stream learning methods, which are typically capable of handling nodes of a single type, such as logistic regression, SVM, random forests etc. In this work we refer to the addressed transformation task as *network decomposition*.

Network decomposition is formally defined as follows. Given a knowledge graph $G = (N, E, R, C)$, the main objective is to construct a mapping $G \rightarrow G_H$; where G_H corresponds to a homogeneous network $G_H = (N_H, E_H, W_H)$, where N_H are nodes of a single type (also referred to as the *target* type) and W_H represents a set of weights induced from the relations present in the knowledge graph. Network decomposition is illustrated in Figure 1.

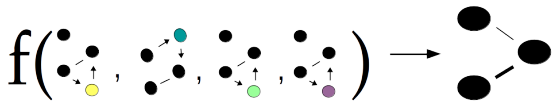


Fig. 1. Schematic representation of network decomposition. Intermediary nodes (yellow, green and purple) are used to weight the edges in the final network, consisting exclusively of target nodes (black).

The particular class of network decompositions we are interested in is based on node triplet counts. Here, a set of nodes $\{u, m, v\}$ is used to construct a weighted edge between u and v. The node m corresponds to an intermediary node, to which both u and v are connected. A triplet thus corresponds to a directed path of length two ($u \rightarrow m \rightarrow v$). In this work we build on HINMINE methodology [6], a recently proposed approach for knowledge graph decomposition, which introduces eight different heuristics for triplet enumeration and their transformation to weighted edges in the final homogeneous network.

The HINMINE approach The HINMINE approach [6] is a recent development that uses text mining inspired heuristics in network decomposition. A similar approach was taken in DeepWalk [9] showing that text mining inspired approaches can be successfully applied to network analysis. The in-house developed HINMINE methodology proposes eight text mining heuristics, which consider different triplets as different words. The simplest heuristic corresponds to simple term counts, which translates directly to $\{u, v, m\}$ triplet enumeration. Each edge is thus weighted based on the number of manually chosen triplets, the two target nodes under consideration are part of.

More formally, given a heuristic function f, a weight of an edge between the two nodes u and v is computed as

$$w(u,v) = \sum_{\substack{m \in M \\ (u,m) \in E \\ (m,v) \in E}} f(m); \tag{1}$$

where the $f(m)$ represents the weight function and m an intermediary node. Here, M represents the set of intermediary nodes and E the set of a knowledge graph's edges. All weight functions used in this study are summarized in Table 1. The node set B denotes all nodes of the base type. We use the following notations: $f(t, d)$ denotes the number of times a term t appears in the document d and D denotes the corpus (a set of documents). We assume that the documents in the set are labeled, each document belonging to a class c from a set of all classes C. We use the notation $t \in d$ to describe that a term t appears in document d. Where used, the term $P(t)$ is the probability that a randomly selected document contains the term t, and $P(c)$ is the probability that a randomly selected document belongs to class c. We use $|d|$ to denote the length (in words) of a document, and avgdl denotes the average document length in the corpus. Heuristics considered in this work are summarized in Table 1.

One of the main caveats of the current HINMINE implementation is non-automatic triplet selection—the choice of representative triplets is left to a domain expert. Further, the original HINMINE methodology does not address the issue of heuristic selection, even though it was empirically proven that heuristic selection is clearly data-dependent, i.e. different heuristics are optimal for different knowledge graphs [6]. The original contributions of this work are aimed at addressing these issues.

2.2 Related work on node classification

The task of node classification has been previously addressed in the field of complex network analysis. The first group are knowledge graph aggregation approaches, where relational graphs are used as main inputs. For example, the *Trans* family of algorithms [10, 11] projects subject-predicate-object triplets to hyperplanes, where entity resolution and similar tasks can be conducted. With the recent success of deep learning-based methods, neural network architectures for triplet embedding construction have also gained considerable attention [12].

Table 1. Term weighing schemes, taken from [6], tested for decomposition of knowledge graphs and their corresponding formulas in text mining.

Scheme	Formula
`tf`	$f(t,d)$
`if-idf`	$f(t,d) \cdot \log\left(\frac{\lvert D\rvert}{\lvert\{d' \in D : t \in d'\}\rvert}\right)$
`chi^2`	$f(t,d) \cdot \sum_{c\in C} \frac{(P(t\wedge c)P(\neg t\wedge \neg c) - P(t\wedge\neg c)P(\neg t\wedge c))^2}{P(t)P(\neg t)P(c)P(\neg c)}$
`ig`	$f(t,d) \cdot \sum_{c\in C, c'\in\{c,\neg c\} t'\in\{t,\neg t\}} \left(P(t',c') \cdot \log\frac{P(t'\wedge c')}{P(t')P(c')}\right)$
`gr`	$f(t,d) \cdot \sum_{c\in C} \frac{\sum_{c'\in\{c,\neg c\}}\sum_{t'\in\{t,\neg t\}}\left(P(t',c')\cdot\log\frac{P(t'\wedge c')}{P(t')P(c')}\right)}{-\sum_{c'\in\{c,\neg c\}} P(c)\cdot\log P(c)}$
`delta-idf`	$f(t,d) \cdot \sum_{c\in C}\left(\log\frac{\lvert c\rvert}{\lvert\{d'\in D : d'\in c\wedge t\in d'\}\rvert} - \log\frac{\lvert\neg c\rvert}{\lvert\{d'\in D : d'\notin c\wedge t\notin d'\}\rvert}\right)$
`rf`	$f(t,d) \cdot \sum_{c\in C}\log\left(2+\frac{\lvert\{d'\in D : d'\in c\wedge t\in d'\}\rvert}{\lvert\{d'\in D : d'\notin c\wedge t\notin d'\}\rvert}\right)$
`bm25`	$f(t,d)\cdot\log\left(\frac{\lvert D\rvert}{\lvert\{d'\in D : t\in d'\}\rvert}\right)\cdot\frac{k+1}{f(t,d)+k\cdot\left(1-b+b\cdot\frac{\lvert d\rvert}{\text{avgdl}}\right)}$

The second family of classification methods are based on node embeddings, including algorithms SDNE [13], LINE [14], Personalized PageRank-based methods [15, 6, 16] and similar. They produce a vectorized representation of a graph corresponding to individual nodes, from which θ can be constructed. Embeddings obtained by such methods are commonly used for classification of nodes in homogeneous networks—networks with only a single type of node, or need to be specifically adapted for e.g., hierarchical network structure [17]. In this work we leverage the latter, where node labels are learned from a homogeneous network, obtained from a knowledge graph via network decomposition. Here, a homogeneous network consists of a subset of knowledge graph's nodes, and its edges are derived from the knowledge graph's edges.

2.3 Related work on parameter optimization

Optimization is one of the key components of majority of machine learning, data mining, as well as network analysis algorithms. Given a set of constraints $\mathfrak{C}$, a parameter space ψ and a scoring function $g(x)$, where x corresponds to a n-dimensional solution vector, the optimization objective can be formulated as a maximization (or minimization) problem of the form:

$$R_{opt} = \arg\max_{x\in\psi} \big[g(x)\big];$$
$$\text{subject to } \mathfrak{C}$$

Combinatorial optimization deals with discrete parameter spaces. In a common setting, multiple sets of possible inputs are considered. In this work we leverage the differential evolution algorithm for the proposed optimization task.

Differential evolution. Differential evolution (DE) [18, 19, 20] is an iterative approach, where possible optimal solutions are represented as parts of a larger set of solutions evolving over several generations. Given a nonempty set $X \subseteq \mathbb{R}^n$ and an objective function $f : X \to \mathbb{R}$, the objective of DE is to find such $x^* \in X$, such that $f(x^*) \neq -\infty$ and $f(x^*) \leq f(x)$ holds for all $x \in X$. Parameter vectors are formulated as $x_{i,\mathcal{G}} = \{x_{1,i,\mathcal{G}}, x_{2,i,\mathcal{G}}, \ldots, x_{n,i,\mathcal{G}}\}; i = 1, 2, \ldots, A$. The number of parameter vectors A is called the population size, while $\mathcal{G}$ denotes the current generation of vectors. The set of solutions is evolved via different evolution operators;

1. **Mutation**, defined as a combination of three randomly chosen vectors x_a, x_b, x_c, combined into a new vector $x_{\mathcal{G}+1}$, where $F \in [0, 2]$ is a constant. Formally the mutation is defined as:

$$v_{\mathcal{G}+1} = x_{a,\mathcal{G}} + F(x_{b;\mathcal{G}} + x_{c;\mathcal{G}})$$

2. **Recombination**, where successful solutions from the previous generation are incorporated into current generation. Elements from the solution vector v are incorporated into vector x to form the final vector u, defined as:

$$u_{j,i,\mathcal{G}+1} = \begin{cases} v_{j,i,\mathcal{G}+1}, & \text{for } \text{rand}_{j,i} \leq CR \vee j = I_{rand} \\ x_{j,i,\mathcal{G}} & \text{for } \text{rand}_{j,i} > CR \wedge j \neq I_{rand}; \end{cases}$$

   where $i = 1, 2, \ldots, A$ and $j = 1, 2, \ldots, n$. For all i, j, $\text{rand}_{j,i}$ is a sample of a random variable distributed as $U[0, 1]$, and I_{rand} is a random element of $\{1, 2, \ldots, Q\}$ which ensures that $u_{i,\mathcal{G}+1} \neq x_{i,\mathcal{G}}$. The CR denotes crossover-rate, the probability of a recombination event.

3. **Selection**, where best individuals from current population are used for a new population, meaning that

$$x_{i,\mathcal{G}+1} = \begin{cases} u_{i,\mathcal{G}+1}, & \text{if } f(u_{i,\mathcal{G}+1}) \leq f(x_i, \mathcal{G}) \\ x_{i,\mathcal{G}} & \text{if } f(u_{i,\mathcal{G}+1}) > f(x_i, \mathcal{G}) \end{cases}.$$

Each possible solution in the population must have predetermined lower and upper bounds of possible values. Initial solutions x_0 are initialized randomly, i.e. $\forall s \in x_0; s \sim U[0, 1]$. Differential evolution can be used for optimization of both discrete, as well as continuous objective functions [21]. One of the key parts of each stochastic optimization procedures is solution representation. We discuss solution representation along with the proposed approach in the next section.

3 The proposed approach

The proposed approach consists of the following two steps. First, the learning problem and solution are presented. This step is followed by differential evolution of different decomposition heuristics.

We begin by describing some of the improvements to the original HINMINE methodology, which form the basis for the proposed knowledge graph decomposition. As stated in Equation 1, edges between target nodes are artificially constructed based on different triplet count heuristics, summarized in Table 1. In this work we first address two issues, that arise when real networks are considered: ground-truth edge information and parallelism needed for analysis of larger networks. In this section we first describe the proposed improvements. We then present the problem of finding the optimal network decomposition as an optimization problem. The section concludes with the description of the graph decomposition algorithm.

Current version of the HINMINE methodology enumerates individual triplets iteratively, i.e. one at a time. This process is spatially non-demanding, as triplets are dynamically generated. Building on this idea, we propose the following modification. First, a set of triplets is generated upfront and temporarily stored. Next, decompositions, corresponding to the triplets are computed in parallel. Once computed, the set of decompositions is returned. This modification is controlled by a single parameter—the batch size. The memory consumption increases linearly with respect to the number of triplets used in a single batch.

Ground-truth edge information. Let u and v represent two target nodes, between which an artificial edge is to be constructed. Currently, any prior information on edge information between the two nodes is not taken into account. This means that the current version of the HINMINE methodology works best for target nodes with no ground truth edges. In this work we extend the methodology to include also the set of ground truth edge weights w_g.

$$w(u,v) = \alpha\Big(w_g(u,v)\Big) + \beta\left(\sum_{\substack{m\in M\\(u,m)\in E\\(m,v)\in E}} f(u,v,m)\right); \tag{2}$$

where m represents an intermediary node between u and v. The w_g represents ground-truth network weights and f a network decomposition heuristic. We further introduce two parameters, α, β, which are used to weight the contributions of the two different edge types. When α is set to 0, only artificial edges are used—this is the current implementation of the HINMINE methodology. As the focus of this work is learning from aggregated graphs, the final decomposition D_f is normalized to a right-stochastic matrix, where the weight between two nodes is redefined as: $w_{norm}(u,v) = \frac{1}{\sum_{v=1}^{K} w(u,v)} w(u,v)$; where K represents the number of columns in the graph-adjacency matrix. The final graph $G = (N_{norm}, w_{norm})$ can be used for different down-stream learning tasks, such as node and edge

classification, as well as clustering. This proposition is one of the first network decomposition schemes, where prior information on edge weights can be taken into account.

Network decomposition as an optimization problem. The main objective of this study can be formulated as follows. Given a permissive set of decomposition heuristics $P(\mathfrak{D})$, a set of operators for combining different heuristics $\mathfrak{S}$ and a set of permissible triplet sets which are used for decomposition $P(\mathfrak{T})$, the objective is to calculate best possible network decomposition $X \in \mathbb{R}^{n \times n}$, computed with decomposition function $\tau : P(\mathfrak{D}) \times \mathfrak{S} \times P(\mathfrak{T}) \rightarrow \mathbb{R}^{n \times n}$. The n represents the number of target nodes. Individual decomposition X is evaluated via a scoring function $\rho : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$. The optimization objective function can thus be stated as:

$$X_{opt} = \underset{(d,o,t) \in P(\mathfrak{D}) \times \mathfrak{S} \times P(\mathfrak{T})}{\arg\min} \Big[\rho(\tau(d, o, t)) \Big].$$

The proposed formulation does not take into account any specific decomposition scoring function ρ as the definition of this function is context-dependent. We continue with descriptions of the parameter space, defined as $P(\mathfrak{D}) \times \mathfrak{S} \times P(\mathfrak{T})$.

The set of all possible triplet sets $\mathfrak{T}$. To construct decompositions of a knowledge graph, we use one or more possible triplets from the set $\mathfrak{T}$ of all possible triplets. We consider all forward relations between a node triplet (u, m, v), e.g., $u \xrightarrow{\text{Directed}} m \xrightarrow{\text{LikedBy}} v$; where u for example corresponds to a person, m to a movie and v to another person. Further, we also consider reverse relations, such as for example $u \xrightarrow{\text{associatedTo}} m \xrightarrow{\text{associatedTo}} v$; which emerge as relevant for biological problems, where for example different functional domains or similar are common to two proteins. The proposed approach supports decomposition based on multiple triplet sets simultaneously. Here, enumerations obtained from individual triplets are summed into a single scalar, used as input for heuristic evaluation. The proposed approach explores the triplet parameter space, defined as $P(\mathfrak{T})$, meaning that the upper bound for the number of triplets considered (exhaustive search) is $2^{|\mathfrak{T}|}$.

Set of decomposition heuristics $\mathfrak{D}$. The proposed approach can leverage all heuristics defined in Table 1. In the approach, we explore the space of all possible heuristics as well as their combination, meaning that the heuristic parameter space is $P(\mathfrak{D})$.

Set of heuristic combination operators. Let $\{h_1, h_2, \ldots, h_k\}$ be a set of matrices, obtained using different decomposition heuristics. We propose four different heuristic combination operators.

1. **Element-wise sum**. Let $\oplus$ denote elementwise matrix summation. Combined aggregated matrix is thus defined as $M = h_1 \oplus \cdots \oplus h_k$, a well defined expression as $\oplus$ represents a commutative and associative operation.
2. **Element-wise product**. Let $\otimes$ denote elementwise product. Combined aggregated matrix is thus defined as $M = h_1 \otimes \cdots \otimes h_k$.

3. **Normalized element-wise sum**. Let $\oplus$ denote elementwise summation, and $\max(A)$ denote the largest element of the matrix A. Combined aggregated matrix is thus defined as $M = \frac{1}{\max(h_1 \oplus \cdots \oplus h_k)}(h_1 \oplus \cdots \oplus h_k)$. As $\oplus$ represents a commutative operation, this operator can be generalized to arbitrary sets of heuristics without loss of generality.
4. **Normalized element-wise product**. Let $\otimes$ denote elementwise product, and $\max(A)$ denote the largest element of the matrix A. Combined aggregated matrix is thus defined as $M = \frac{1}{\max(h_1 \otimes \cdots \otimes h_k)}(h_1 \otimes \cdots \otimes h_k)$. This operator can also be generalized to arbitrary sets of heuristics.

In this work we consider whole decomposition space, i.e. $P(\mathfrak{D}) \times \mathfrak{S} \times P(\mathfrak{T})$.

Solution representation and graph decomposition algorithm Having defined the parameter search space, we discuss in this section the representation of individual solutions, evaluated with DE. Let $v \in \mathbb{R}^c; c = |\mathfrak{D}| + |\mathfrak{T}| + |\mathfrak{S}| + 2$ be a random vector with components $v_i \sim U([0, 1])$ represent a one-dimensional input vector corresponding to the set of heuristics, operators and triplets used to obtain a decomposition. Each heuristic, operator as well as triplet correspond to an element in $\mathbb{R}^c$. A *solution* vector s is defined via an indicator function $\mathbb{I}$, where each element is defined as $\mathbb{I}(v_i \geq 0.5)$. The first $|P(\mathfrak{D})|$ components of the resulting vector determine the decomposition heuristics, used for the decomposition, the second $|P(\mathfrak{T})|$ determine the triplets and the third $|\mathfrak{S}|$ components determines the heuristic combination operator used. The final 2 components of the vector correspond to parameters α and β from Equation 2.

As each field in the vector corresponds to either a triplet or a heuristic, once the indicator function is applied, a solution can be uniquely defined. The first non-zero value in the operator vector denotes the operator used for combining possible solutions. Hence, we consider only a single type of operator for each individual solution. The proposed approach can be compactly summarized as follows.

1. First, a set of decomposition triplets and heuristics is selected.
2. Next, a set of solution combination operators, described in the previous section is selected.
3. Iterative evolution consists of the following three steps. Mutation, followed by recombination and selection. Selected solutions represent potential optima, and are used in the next generation (iteration). Evolution is run for a predefined number of generations.

As this work is focused on the general approach for finding the near-optimal network decomposition, we discuss an example using a task-specific evaluation function in the next section.

4 Experimental setting

We test the proposed approach on the following datasets.

The IMDB dataset. The main classification task related to this dataset corresponds to classification of individual movie's genres, based on actors, directors and movies [6]. Here, 300 nodes are labeled, whereas the whole network consists of $6,387$ nodes and $14,714$ edges. An example triplet yielding a valid decomposition for this dataset is: $Actor \xrightarrow{\text{actsIn}} Movie \xrightarrow{\text{directedBy}} Director$. This network does not contain any ground-truth edge information, and thus the α parameter is not relevant for this problem. The DE was parameterized with 15 generations of size 10. The CR rate was set to 0.4, mutation rate to 0.05 and the selection strategy *best1bin*, the default option in [22]. The same parameterization is used for the epigenetics dataset.

The epigenetics dataset. An example from the biological domain used is the recently introduced epigenetics knowledge graph [23], where proteins, genes and other biological entities are connected with different relations. Further, different protein-protein interactions are annotated with ground-truth edge weights corresponding to reliability of interactions. We annotate each protein in the network with the corresponding *GO* terms, associated with their functions, which is achieved as follows. Functional annotations are obtained from the Intact database [24]. We sort annotations by frequency and select 100 of the most common terms, which correspond to 100 classes being predicted. The α parameter was selected directly from the interval between 0 and 1. The classification goal for this dataset is thus protein function prediction. The network consists of $2,204$ nodes and $2,772$ edges, 456 nodes are target nodes for which the classification is $Protein \xrightarrow{\text{contains}} Domain \xrightarrow{\text{contains}} Protein$.

Performance evaluation. We evaluate the performance as follows. If possible, we compute scores for all possible combinations of triplets, heuristics and operators. Once computed, a global parameter landscape is obtained, which can be used to directly assess the algorithm's performance. Further, we use randomized grid search as the baseline approach.[4] In this work we evaluate an aggregated network's quality by computing a classification performance metric in a process of 10-fold cross validation. The node label classification approach is the same as used in the original HINMINE methodology. The aggregated network is used to construct a set of personalized page-rank vectors (PPR), which represent the feature matrix F. Label matrix T corresponds to individual labels, assigned to distinct nodes and consists of $N \cdot |C|$ cells, where N is the number of all nodes and C the set of all classes.[5] The tuple (F, T) is used as input for one-vs-many logistic regression classifier. In this work we use the macro F_1 measure for evaluation of individual solutions. The macro F_1 score averages individual, pairwise F_1 scores, defined as : precision $= \frac{tp}{tp+fp}$;, recall $= \frac{tp}{tp+fn}$, $\mathrm{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$;

[4] The machine used for evaluation was an of-the-shelf Lenovo $y510p$ laptop with an i7 Intel processor (8 cores) and 4GB of RAM.

[5] The feature matrix is not memory efficient, as it uses $\mathcal{O}(N^2)$ space, yet optimization of this part of the procedure is out of the scope of this study.

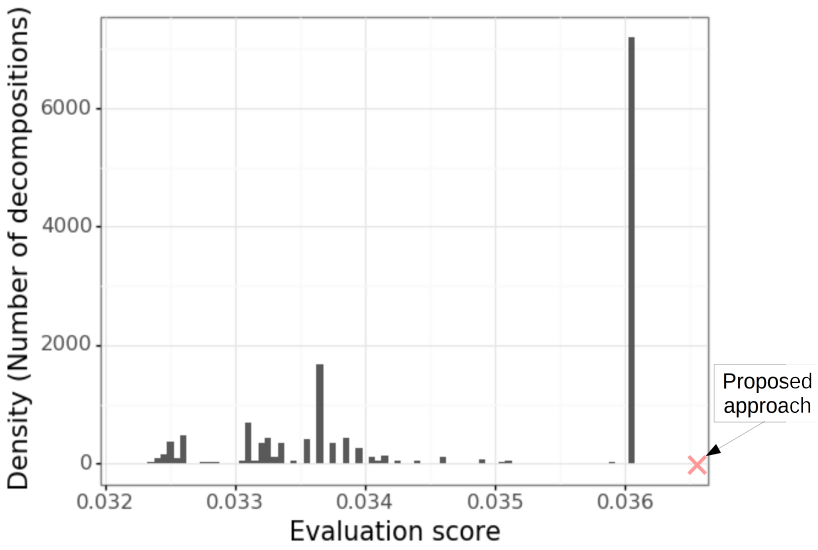


Fig. 2. Score distribution over all parameter space for the IMDB network decomposition problem. The red line denotes the optimal solution found by the proposed approach.

over all classes, where fp denotes false positives, tp true positives and fn false negatives. The C represents the set of target classes. The total score thus equals

$$\text{macro}F_1 = \frac{1}{|C|} \sum_{i \in C} F_{1(i)}.$$

We leave extensive computational evaluation of different DE parameterizations for further work.

5 Results and discussion

For the IMDB network we compute all possible decompositions and visualize the solution obtained by the proposed as the red line in Figure 2.

We observe, that the proposed approach found the global maximum—here corresponding to the rightmost combination of decompositions. The final decomposition was obtained by combining the following heuristics: `ig` and `gr`. The following triplets were used: $Movie \xrightarrow{\text{features}} Person \xrightarrow{\text{actsIn}} Movie$, $Movie \xrightarrow{\text{directedBy}} Person \xrightarrow{\text{directed}} Movie$, $Movie \xrightarrow{\text{features}} Person \xrightarrow{\text{directed}} Movie$. The combination operator used was element-wise product.

As computing all decompositions for the epigenetics problem is not computationally feasible, we compare the result obtained with the average decomposition performance obtained by sampling the decomposition distribution. The mean performance for this approach estimates to 0.0284, whereas the mean performance using 10000 different decomposition samples estimates to 0.0253. The samples were selected by randomly permuting the solution vectors with values between 0 and 1. Further, in this case the obtained decomposition is not optimal, as the best sampled decomposition scored with 0.0293. The decomposition returned by the proposed approach consisted of the following heuristics: `ig`, `gr` and `bm25`. The triplets:

$$Protein \xrightarrow{\text{contains}} Domain \xrightarrow{\text{contains}} Protein$$
$$Protein \xrightarrow{\text{interactsWith}} Protein \xrightarrow{\text{subsumes}} Protein$$
$$Protein \xrightarrow{\text{belongsTo}} Family \xrightarrow{\text{belongsTo}} Protein$$
$$Protein \xrightarrow{\text{isRelatedTo}} Phenotype \xrightarrow{\text{isRelatedTo}} Protein$$
$$Protein \xrightarrow{\text{interactsWith}} Protein \xrightarrow{\text{interactsWith}} Protein$$

were found as the combination used to obtain the best decomposition. The combination operator used for decomposition aggregation was elementwise product. The results from both experiments averaged over five runs with different stochastic seeds are summarized in Table 2.

The two experiments indicate that the proposed approach can provide the currently not known solution to proper knowledge graph decomposition and aggregation for node label classification. The approach in both cases outperforms the mean baseline, where for the IMDB dataset it performs the same as the best possible decomposition, where for the epigenetics dataset it performs worse compared to the best decompositions out of 10000 random samples. To understand the limitations of the proposed approach, different stochastic optimization procedures could be tested, yet such extensive experimental evaluation is left for further work.

The decomposition triplets, found as crucial for the epigenetics dataset are biologically relevant, supported by the following observations. Protein domains have been previously associated and recognized as key features for function prediction [25]. Latent interaction partners are often used for constructing protein-

Table 2. Empirical result summary. The F1 corresponds to macro-F1 score. The bold numbers denote the global optimum, identified for the IMDB dataset. Min and Max F1 scores denote minimum and maximum network decomposition performance.

Dataset	Min F1	Max F1	Mean F1	Proposed approach	DE	Exhaustive search
IMDB	0.0315	**0.0372**	0.0346	**0.0372**	50min	$\approx 22h$
Epigenetics	0.0211	0.0296	0.0243	0.0284	6h	$>$ 1day

protein interaction prediction tools, i.e. two proteins often interact if they both interact with a third partner [26]. Phenotypes are highly correlated with protein function [27]. Protein families are also relevant for function prediction, as they can correspond directly to protein binding sites and similar functional domains [28, 29, 30]

The proposed method performs significantly faster compared to exhaustive search. For the IMDB dataset, the global optimum was found more than twenty times faster. A similar pattern was observed for the Epigenetics dataset, yet the number of possible combinations was too exhaustive and was evaluated via sampling.

6 Conclusions

In this work we present a novel end-to-end stochastic-optimization-based approach for network decomposition and subsequent aggregation. This work builds on current state-of-the-art HINMINE methodology, which does not provide a fully automated procedure for obtaining dataset-specific decompositions. We demonstrate the use of the proposed method on two real life knowledge graphs, where in one there is also ground-truth edge classification information available. The proposed method performs better than average decomposition, which indicates that the proposed stochastic optimization in the form of differential evolution could provide a feasible approach for automated knowledge graph-based learning.

As the proposed approach automatically identifies the relations that are relevant for node classification, we can interpret the final result in terms of novel, meaningful relations, previously not considered as important for the given knowledge graph for the given classification task. The biological interpretations indicate one of the possible uses of final decomposition triplets. Apart from obtaining a better predictive model, the final result is interpretable and can be linked to existing knowledge. Further, the final set of triplets uncovers the novel candidate relations. As the set of relations differs from task-to-task, the triplets could potentially offer qualitative explanation for key relations relevant to black-box models, such as deep neural networks.

Further work includes extensive experimental evaluation on more types of knowledge graphs, as well as the investigation of different optimization routines, for example Bayesian optimization, which has proven invaluable for automated machine learning. Further, we will investigate how the spatially intensive homogeneous networks could be reduced during the learning process.

Bibliography

[1] Burt, R., Minor, M.: Applied Network Analysis: A Methodological Introduction. Sage Publications (1983)
[2] Sun, Y., Han, J.: Mining Heterogeneous Information Networks: Principles and Methodologies. Morgan & Claypool Publishers (2012)

[3] Consortium: Gene Ontology: Tool for the unification of biology. the gene ontology consortium. Nature genetics **25**(1) (May 2000) 25–29

[4] Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: SEMANTiCS (Posters, Demos, SuCCESS). (2016)

[5] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1) (2016) 11–33

[6] Kralj, J., Robnik-Šikonja, M., Lavrač, N.: HINMINE: Heterogeneous information network mining with information retrieval heuristics. Journal of Intelligent Information Systems (2017) 1–33

[7] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3) (2008) 93

[8] de Sousa, C.A.R., Rezende, S.O., Batista, G.E.: Influence of graph construction on semi-supervised learning. In: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer (2013) 160–175

[9] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2014) 701–710

[10] Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12) (2017) 2724–2743

[11] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI. Volume 14. (2014) 1112–1119

[12] Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Transactions on Knowledge and Data Engineering (2018)

[13] Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2016) 1225–1234

[14] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee (2015) 1067–1077

[15] Grčar, M., Trdin, N., Lavrač, N.: A methodology for mining document-enriched heterogeneous information networks. The Computer Journal **56**(3) (2013) 321–335

[16] Kralj, J., Valmarska, A., Robnik-Šikonja, M., Lavrač, N.: Mining text enriched heterogeneous citation networks. In: Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining. (May 2015) 672–683

[17] Žitnik, M., Leskovec, J.: Predicting multicellular function through multi-layer tissue networks. Bioinformatics **33**(14) (2017) i190–i198

[18] Fleetwood, K.: An introduction to differential evolution. In: Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November, Brisbane, Australia. (2004) 785–791
[19] Price, K., Storn, R.M., Lampinen, J.A.: Differential evolution: A practical approach to global optimization. Springer Science & Business Media (2006)
[20] Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution–an updated survey. Swarm and Evolutionary Computation **27** (2016) 1–30
[21] Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation **15**(1) (2011) 4–31
[22] Jones, E., Oliphant, T., Peterson, P.: SciPy: Open Source Scientific Tools for Python. (2014)
[23] Škrlj, B., Kralj, J., Vavpetič, A., Lavrač, N.: Community-based semantic subgroup discovery. In: Proceedings of New Frontiers in Mining Complex Patterns Workshop, Cham, Springer International Publishing (2018) 182–196
[24] Orchard, S., Ammari, M., Aranda, B., Breuza, L., Briganti, L., Broackes-Carter, F., Campbell, N.H., Chavali, G., Chen, C., del Toro, N., Duesbury, M., Dumousseau, M., Galeota, E., Hinz, U., Iannuccelli, M., Jagannathan, S., Jimenez, R., Khadake, J., Lagreid, A., Licata, L., Lovering, R.C., Meldal, B., Melidoni, A.N., Milagros, M., Peluso, D., Perfetto, L., Porras, P., Raghunath, A., Ricard-Blum, S., Roechert, B., Stutz, A., Tognolli, M., van Roey, K., Cesareni, G., Hermjakob, H.: The MIntAct project–IntAct as a common curation platform for 11 molecular interaction databases. Nucleic Acids Research **42**(Database issue) (January 2014) D358–63
[25] Marchler-Bauer, A., Derbyshire, M.K., Gonzales, N.R., Lu, S., Chitsaz, F., Geer, L.Y., Geer, R.C., He, J., Gwadz, M., Hurwitz, D.I., et al.: CDD: NCBI's conserved domain database. Nucleic Acids Research **43**(D1) (2014) D222–D226
[26] Szklarczyk, D., Franceschini, A., Wyder, S., Forslund, K., Heller, D., Huerta-Cepas, J., Simonovic, M., Roth, A., Santos, A., Tsafou, K.P., et al.: String v10: Protein–protein interaction networks, integrated over the tree of life. Nucleic Acids Research **43**(D1) (2014) D447–D452
[27] Kelley, L.A., Mezulis, S., Yates, C.M., Wass, M.N., Sternberg, M.J.: The Phyre2 web portal for protein modeling, prediction and analysis. Nature Protocols **10**(6) (2015) 845
[28] Finn, R.D., Attwood, T.K., Babbitt, P.C., Bateman, A., Bork, P., Bridge, A.J., Chang, H.Y., Dosztányi, Z., El-Gebali, S., Fraser, M., et al.: Interpro in 2017beyond protein family and domain annotations. Nucleic acids research **45**(D1) (2016) D190–D199
[29] Lee, J., Konc, J., Janežič, D., Brooks, B.R.: Global organization of a binding site network gives insight into evolution and structure-function relationships of proteins. Scientific Reports **7**(1) (2017) 11652
[30] Škrlj, B., Kunej, T., Konc, J.: Insights from ion binding site network analysis into evolution and functions of proteins. Molecular Informatics (2018)

DDeMON: Ontology-scale function prediction by Deep Learning from Dynamic Multiplex Networks

Jan Kralj JAN.KRALJ@IJS.SI
Jožef Stefan Institute, Department of Knowledge Technologies, Jamova 39, 1000 Ljubljana, Slovenia

Blaž Škrlj BLAZ.SKRLJ@IJS.SI
Jožef Stefan Institute, Department of Knowledge Technologies, Jamova 39, 1000 Ljubljana, Slovenia

Editor:

Abstract

Motivation: Biological systems can be due to abundance of data studied on multiple different levels of information, including gene, protein, RNA and other interaction networks. We explore how fusion of systems' level information with temporal dynamics of gene expression can be used in combination with non-linear approximation power of deep neural networks to predict novel genes, related to stress response and other molecular functions in *Solanum sp.*
Results: We propose DDeMON (Dynamic Deep learning from temporal Multiplex Ontology-annotated Networks), an approach for scalable, systems-level inference of function annotation using time-dependent multiscale biological information. We demonstrate the use of the proposed method on recently introduced experimental expression data, discovering multiple novel biomarkers, valuable for understanding pathogen response, as well as preliminary disease detection.

1. Introduction

In this report, we present an approach to predicting gene function from experimental data. The approach, based on our previous work on the HinMine algorithm Kralj et al. (2017) fuses data from a diverse collection of sources into a single network and then performs data analysis on the resulting network. This work represents one of the first attempts to modeling a plant's stress response with respect to both temporal, as well as static information incorporating multiple levels of biological information, including protein-protein, protein-gene and gene-gene interaction networks.

2. Data collection

The data set we used was the result of a biological experiment, performed at NIB, examining gene expression after a viral infection with the PVY (Potato Virus Y) virus Baebler et al. (2014). Potato plants were grown in a controlled environment for 4 weeks. After this period, the leaves of the plants were dusted in carborundum powder and rubbed with cheesecloth dipped in a sap prepared from the leaves of PVY-infected tobacco plants. Control plants were treated with mock inoculations where water was used instead of the sap.

On day 1, 3 and 6 after infection, RNA samples were collected from plants and analyzed using microarrays. The result was a collection of 42,034 continuous gene expression values for each treatment (mock or virus) and each day post infection (1, 3 and 6). For microarray probes mapping to the same potato plant gene, we obtained the gene expression by averaging expression values over all the probes - the 42,034 values were thus further transformed into 33,937 values, each showing expression of one particular gene.

3. The proposed DDeMON approach

The proposed approach is an extension of the existing methodology HinMine for data mining on large heterogeneous information networks.

HinMine is a two step methodology to mine heterogeneous information networks (defined by Sun and Han (2012)), in which a certain type of nodes (called the target type) is labeled. In the first step of the methodology, the heterogeneous network is decomposed into a set of homogeneous networks, containing only the target nodes of the original network. In each homogeneous network two nodes are connected if they share a particular direct or indirect link in the original heterogeneous network. Take as an example a network containing two types of nodes, *P*apers and *A*uthors, and two edge types, *Cites* (linking papers to papers) and *Written_by* (linking papers to authors). From it, we can construct two homogeneous networks of papers: the first, in which two papers are connected if one paper cites another, and the second, in which they are connected if they share a common author. The choice of links used in the network decomposition step requires expert who takes the meaning of links into account and chooses only the decompositions relevant for a given task. In the second step of the methodology, the homogeneous networks are *propositionalized* – for each node in the network, a feature vector representing the node is calculated. This step is performed using the Personalized PageRank method Page et al. (1999) and is described in more detail in Section 3.4

In the following sections, we discuss the proposed DDeMON approach in detail. We begin by describing the data used, followed by overview of the computational methodology used for feature construction and learning.

3.1 Data used

We use data from several main sources, described below.

1. *Gene expression data*. The primary data used in our approach is the gene expression data. As described in Section 2, we processed the results of the original biological experiment to obtain a time series of 12 expression values for each gene. In total, we analyzed 33,937 genes.
2. *Large knowledge graph* The first part of additional data used in the approach is the Large Knowledge Graph. This is a graph composed of publicly available data on gene-gene interactions and contains edges of various types. The construction and use of this network is further described in Section 3.2.2
3. *PubMed database* Finally, we connected the genes in our database to PubMed articles, related to them. To obtain a representative set of articles for most genes, we looked not at papers, referring the genes themselves, but rather papers, referring to homolog genes in the species Arabidopsis Thaliana.

The genes were annotated with the GoMapMan taxonomy of gene functions. GoMapMan Rotter et al. (2013) is an open web-accessible resource for gene functional annotations in the plant sciences. It was developed to facilitate improvement, consolidation and visualization of gene annotations across several plant species. Each gene in the data we analyzed

Node type	Number of nodes
Potato Gene	TODO
Arabidopsis Thaliana Gene	TODO
PubMed Article	TODO

Table 1: Nodes of the heterogeneous network, constructed by the DDeMON approach

Edge type (node types)	Number of edges	Directed
Homolog-to (Potato gene - Arabidopsis Thaliana gene)	TODO	No
Cited-by (Arabidopsis Thaliana gene - PubMed Article)	TODO	Yes
Transcription Factor-of (Potato gene - Potato gene)	TODO	Yes
Binding-to (Potato gene - Potato gene)	TODO	Yes
Experimental - healthy (Potato gene - Potato gene)	TODO	Yes
Experimental - infected (Potato gene - Potato gene)	TODO	Yes

Table 2: Nodes of the heterogeneous network, constructed by the DDeMON approach

was annotated by one or more GoMapMan bins, however, out of the 33,937 genes we analyzed, 10,231 of them belonged to the GoMapMan bin 35, meaning the function of those genes is at this point unknown. The approach, outlined in this work, presents an attempt at discovering the gene function of these genes, thus assigning the genes from bin 35 into one of the other bins.

3.2 Network construction

In order to apply the Hinmine propositionalization methodology, we used the three data sources, described in Section 3.1, to construct a single heterogeneous network. The types of nodes and edges, present in the network, are summarized in Tables 1 and 2. We describe the construction of the various aspects of the network in the following sections.

3.2.1 Constructing *Homolog* and *Cited-by* edges

In order to construct the (undirected) edges of type *Homolog* connecting potato genes with Arabidopsis Thaliana genes, we connected each potato gene to all Arabidopsis Thaliana genes in the same homolog group. To obtain homolog groups for the genes, we used data available on the GoMapMan website [1]. To then connect the Arabidopsis Thaliana genes to PubMed articles by directed edges, we used the online tool TAIR Berardini et al. (2015)[2] and converted the exports from TAIR into the network. Note that, even if these edges are not constructed explicitly constructed, the construction of an edge of type *Cited-by* between gene g and paper p also implicitly constructs an edge of type *Cites* between paper p and gene g.

3.2.2 Constructing *Transcription Factor* and *Binding* edges

The edges of the type *Transcription Factor* (either *inhibition*, *activation* or *unknown*) and *Binding* were extracted from a previously constructed comprehensive knowledge network.

1. http://protein.gomapman.org/export/current/biomine/ath_homolog and http://protein.gomapman.org/export/current/biomine/stu_homolog
2. https://www.arabidopsis.org/

To construct that network, we combined the graph of binary PIS-v2 interactions with three layers of publicly available information: protein-protein interactions (PPIs), transcriptional regulation (TR), and regulation through microRNA (miRNA). This resulted in an Arabidopsis thaliana comprehensive knowledge network with 20,012 nodes (19,812 genes, 186 miRNA families, three metabolites, and 11 viral proteins) and 70,091 connections. Each data layer covers unique gene or miRNA subsets in the entire network, with only six nodes present in all four layers, which indicates that our layer selection was well suited for inclusion.

For the purpose of our approach, we extracted all the edges from the comprehensive knowledge network into our heterogeneous network.

3.2.3 Consructing *experimental* edges

The final set of edges we constructed were edges, induced directly from the raw gene expression data, described in Section 3.1. For the purpose of DDeMON, we view the raw data as a collection of time series, each time series charting the strength of the expression of one particular gene. To transform the data into a network, we used the time series data to induce weights on edges between each pair of genes. The goal was to construct a network where a two genes are connected by a strong weight if the experimental data shows they share a similar expression profile (i.e. if the time series, describing their expression over time, are similar). In order to determine the similarity between two time series, we used Dynamic Time Warping (DTW) dtw (2007), an algorithm designed for measuring similarity between two temporal sequences. The algorithm takes as input two temporal sequences and returns the distance from one to the other.

In DDeMON, for each pair of genes g_1 and g_2, we use the inverse value of the distance between their respective expression data series as the weight of the edge between the genes. The inverse value ensures that genes with more similar expression profiles will be connected by a stronger weight. As the raw data contains both expression profiles for genes in infected plants and in healthy plants, we repeat the same procedure on both parts of the raw data, thus constructing both *Experimental_healthy* and *Experimental_infected* edges in the network.

3.3 Homogeneous networks construction

After the heterogeneous network is constructed, DDeMON uses the HinMine algorithm to analyze it. The first step of the HinMine algorithm is construction of homogeneous networks from the input heterogeneous network. In each homogeneous network two nodes are connected if they share a particular direct or indirect link in the original heterogeneous network. In particular, we used the HinMine decomposition methodology to construct a network of genes in which two potato genes are connected if they are homologs to two Arabidopsis Thaliana genes, mentioned in the same PubMed publication. This means that potato genes g_1, g_2 are connected if there exists some path from g_1 - *Homolog-to* - at_1 - *Cited-by* - p - *Cites* - at_2 -*Homolog-to* - g_2.

In addition to the common-PubMed-Article network of genes, described above, we used 6 other homogeneous networks on the same set of nodes (genes), described above: *inhibition_TF*, *activation_TF*, *TODO_TF*, *binding*, *experimental_healthy* and *experimental_infected*. All 7 homogeneous networks

3.4 Feature vector construction

To construct feature vectors from the 7 homogeneous networks, described above, we use the network propositionalization step of the HinMine algorithm. The step calculates feature vectors for each node of the homogeneous network using the personalized PageRank (P-PR) algorithm (Page et al., 1999). The personalized PageRank of node v (P-PR$_v$) in a network is defined as the stationary distribution of the position of a random walker who starts the walk in node v and then at each node either selects one of the outgoing connections or jumps back to node v. The probability (denoted p) of continuing the walk is a parameter of the personalized PageRank algorithm and is usually set to 0.85. Once calculated, the resulting PageRank vectors are normalized according to the Euclidean norm. The resulting vector contains information about the proximity of node v to each of the remaining nodes of the network. We consider the P-PR vectors of a node as a propositionalized feature vector of the node. Because two nodes with similar P-PR vectors will be in proximity of similar nodes a classifier should consider them as similar instances. We use the vectors to classify the nodes from which they were calculated. For a single homogeneous network, the propositionalization results in one feature vector per node. For classifying a heterogeneous network decomposed into k homogeneous networks Grčar et al. (2013) propose to concatenate and assign weights to the k vectors, obtained from the k homogeneous networks.

3.5 Dimensionality reduction and learning

The HinMine algorithm works by calculating Personalized PageRank-based feature vectors for each of the nodes in a network. As the dimension of these feature vectors is by definition equal to the number of nodes in the network, this means that the feature vector construction step, described above, produces and concatenates seven high-dimensional vectors for each gene in the network. We took into account the seven different aspects related to a single node, yielding $7 \cdot |V| \cdot 33{,}937$ features per node. Our initial attempts to learn from such vectors directly did not yield any promising results, as we believe the raw feature vectors contained too much noise. Our solution to this problem is dimensionality reduction, a technique commonly employed when the dimensionality of the data set is too high. We reduced the obtained feature vectors for each aspect as follows:

1. Dimensionality of each aspect was reducted to dimension d using PCA
2. All aspects were finally merged into a single data set, this time of dimensionality $|V|/cdotd/cdot8$.

This allowed us to reduce the size of the network to a more manageable size and ensure that the produced feature vectors can practically be used by a number of supervised machine learning algorithms in the final step, described in Section 3.6.

3.6 Gene function prediction

The steps of the proposed DDeMON approach, described so far, yield a set of feature vectors for each gene in the original data set. The feature vectors are derived from both expression, as well as network data, hence the obtained data set is suitable for all propositional learners. In the final step of the approach, the feature vectors can be used by any of a number of supervised machine learning algorithms. In our experiments, we tested five different classification algorithms, listed below. Note that, due to prohibitively expensive training times, the possibilities of fine-tuning parameters of the tested methods was severely limited.

- *rSVM* - a regular version of Support Vector Machines with equal weights assigned to all training examples. We used the scikit-learn implementation of the support vector machines. The optimal value of the constant C in the algorithm was determined by a grid search over

the range $\{1, 0.1, 0.01, 0.001, 0.0001\}$. Our experiments show that the optimal value for C is 0.001.

- *bSVM* - a balanced version of Support Vector Machines, a variation on the first classifier where the weight of each training example is proportional to the size of the class it belongs to. Like with the *rSVM* classifier, the optimal value for the constant C was 0.001.
- *GBM* - Gradient boosting machines are a tree ensemble learning method, which operates under the assumption, that combining multiple weak classifiers yields a strong one. In each iteration, gradient boosting algorithms exploit the gradient of the error function to weight the instances for the next tree learner. Hence, each iteration, the algorithm emphasizes different parts of the data set. In this work, we used the of-the-shelf GBM algorithm offered in SciKit-Learn library Pedregosa et al. (2011).
- *DNN* and *dDNN* - Deep neural networks - two distinct architectures (discussed next) - We implemented two feedforward deep neural network architectures, which differ by the number of hidden layers. The first architecture (DNN) consists of two hidden layers, and the second, deeper architecture consisted of four hidden layers (dDNN). The optimal number of neurons was determined by using grid search over the value range $\{20, 64, 128, 256, 386, 512, 1024\}$. The activation function which yielded the best performance was ELU, applied after all intermediary layers. We also experimented with ReLU, sigmoid activations. Finally, as the two discussed architectures learn a single target per run, we further generalized the architectures so it can learn an arbitrary number of targets in the same training phase. We achieved so by changing the last layer of the architecture. Here, we replace the single neuron (dDNN and DNN) with sigmoid activation with a set of neurons, where the number of neurons equals the number of targets. We refer to this architecture as (mlDNN - multilabel deep neural network). The loss function used was binary cross entropy. The trainable weights were optimized using Adam (Kingma and Ba (2014)), where the learning rate was set to 0,001. The final number of neurons in hidden layers for the dDNN were

$$64, 128, 386, 512$$

and for DNN

$$20, 500$$

, respectively. The neural networks were implemented using the Tensorflow library (Abadi et al. (2016)).

The above described methods were used in an identical setting: given a GoMapMan bin and a set of genes, some belonging to the bin and others not, we wanted to predict whether a new instance also belongs to the same GoMapMan bin. All methods we used were used to not simply classify new instances, but to provide a score corresponding to the likelihood of the new instance belonging to a given GoMapMan bin. Note that, except for the final mlDNN method, all other methods require separate training on each of the bins we wish to classify genes to.

3.7 Evaluation metrics

The purpose of the DDeMON approach is to correctly classify genes with unknown function into one of the existing GoMapMan bins. To evaluate the reliability of the predictions, produced by our approach, we used 10-fold cross validation and evaluated the performance of all methods, described in Section 3.6. We evaluated the performance of the method on simple binary problems, i.e. answering only the question "Does gene g belong to bin b?",

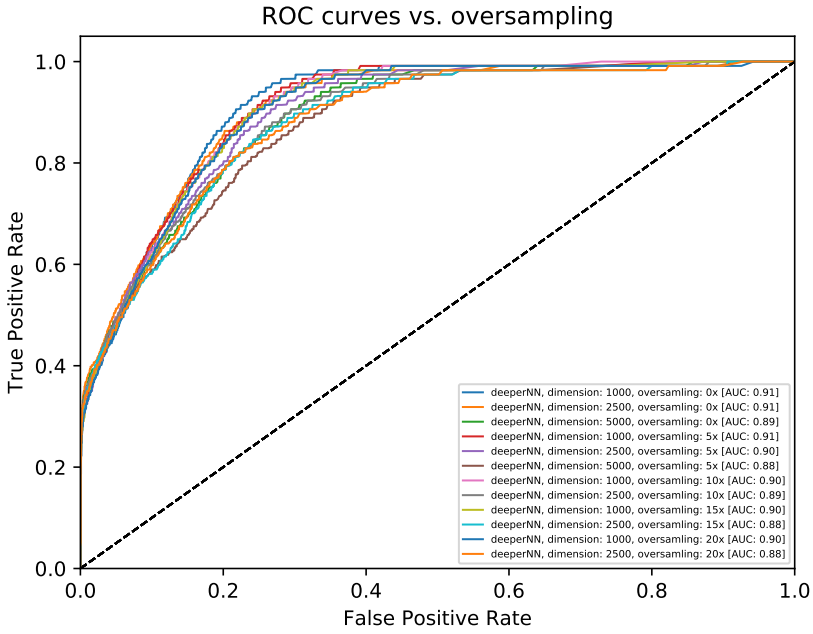


Figure 1: The performance of the *dDNN* classifier on various feature vector dimensions and oversampling values.

allowing us to compare all methods without the need for fitting all classifiers to all possible bins (this would prove prohibitively expensive).

We evaluated the results using the receiver operating characteristic curves (ROC), with corresponding AUC scalar values assessing individual binary classifications.

4. Results

Results show that all of the classifiers we tested perform well, however there are important differences in the performance of each classifier. In this section, we analyze the obtained results. We first analyze the effect of dimensionality reduction and oversampling on classifier performance in Section 4.1 and continue with a comparison of the classifiers in Section 4.2. We conclude with Section 4.3 describing the final results of the DDeMON methodology.

4.1 Effect of dimensionality reduction and oversampling

In this section, we analyze the results of experiments which show the effect of dimensionality reduction on the performance of various classifiers. In order to compare the performance of the classifiers, we tested their ability to predict whether a gene belongs to the GoMapMan bin 20.1 (biotic stress). We selected this bin because it is most closely linked to the phenomenon (virus infection) we analyzed to obtain the data, and computing predictions for all bins is not possible as it would take several months.

Figure 1 shows the performance (measured via a ROC curve) of the *dDNN* architecture with varying sizes of feature vector dimension and different oversampling factors. While

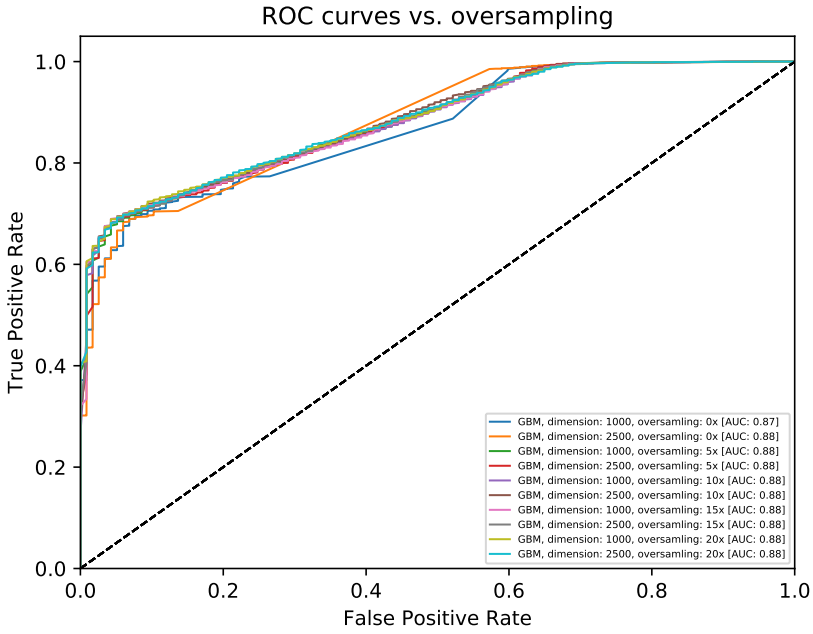


Figure 2: The performance of the *GBM* classifier on various feature vector dimensions and oversampling values.

the differences in performance are small as all classifiers run reasonably well, the best performance was acchieved using no oversampling and feature vector dimension of 1000. Figure 2, showing the performance of the *GBM* classifier, shows an even smaller difference in classifier performance as we vary the dimension of the feature vectors and the amount of oversampling we perform. While running the classifier on a data set of dimension 5000 was computationally infeasible, we still believe the classifier using a vector dimension of 1000 and no oversampling is the best choice for the next step. The classifier appears at first to be outperformed by other classifiers, however on closer inspection, we believe that the marginally smaller AUC can be explained by the fact that the classifier performed worst on exactly the two points between which the largest part of the ROC curve was interpolated, thus appearing to show a worse performance by pure chance.

The results of the experiments on neural networks and *GBM* classifiers shows that using a vector dimension of 1000 id a good choice for the DDeMON algorithm as the algorithm performs well on a data set of this dimension. We therefore performed the experiments on the *SVM* classifiers only on dimension 1000. Figure 3 shows the performance of both the *rSVM* and *bSVM* classifiers. We only show the performance of the *bSVM* classifier on a data set with no oversampling because our experiments confirm that oversampling has no effect on the classification of the *bSVM* classifier. This is expected as the balanced weights, used by *bSVM*, are designed precisely to counteract any imbalance of the set on which the classifier is trained. From Figure 3, we conclude that the best performing classifier is the *bSVM* classifier, again run on a dimension of 1000 and no oversampling.

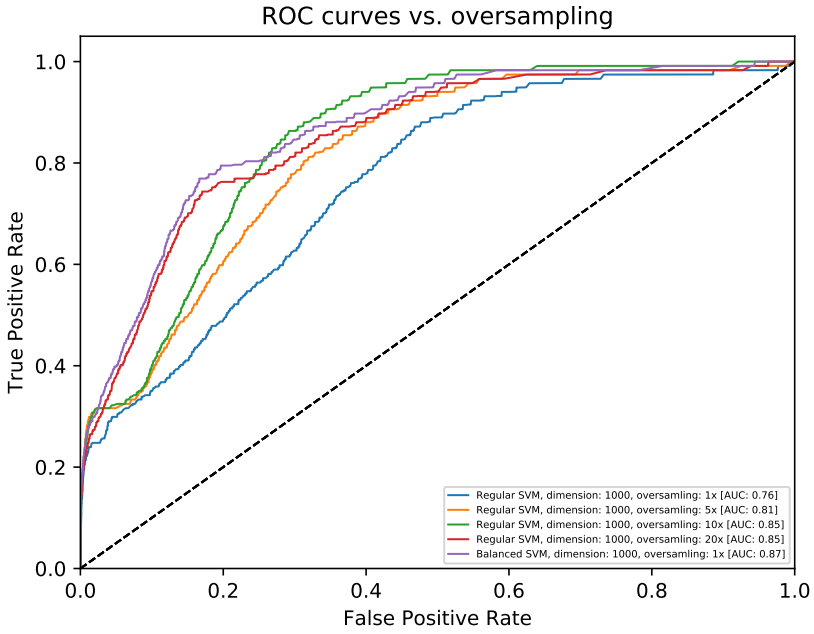


Figure 3: The performance of the *rSVM* classifier on various oversampling values and the *bSVM* classifier.

4.2 Analysis of classifier performance

In this section, we analyze the performance of different classifiers on several of the most important GoMapMan bins. Unlike the previous section, we added the multilabel *mlDNN* classifier to result of this section as the advantage of this classifier is only apparent when classifying into more than one bin, as we discuss below. We omitted the *GBM* classifier from this set of experiments due to the long training times it requires.

Limiting our range of classifiers to the *dDNN*, *SVM* and, additionally, the multilabel *mlDNN* classifiers, all run on a data set of dimension 1000 and without oversampling), we were able to test the performance of the classifiers over several GoMapMan bins. We performed 10-fold cross validation for all second-level bins (i.e., for the bin 20.1, but not for its sub-bins such as 20.1.1) which contained at least 100 of the original 33,937 genes. The results for 20 most populated bins are shown in Table 3.

While the results of Table 3 show that the *dDNN* classifier consistently outperforms the other two classifiers, this should not be the only conclusion we draw from this set of experiments. As shown in Figure 4, it is clear that the *GBM* classifier is not dominated by the *dNN* classifier. Furthermore, the fact that the *mlNN* classifier performs consistently worse than the *dDNN* classifier is an expected result, as the same neural network architecture will perform better if solving a simpler task (binary versus multilabel). The advantage of the *mlNN* classifier lies in its speed, as the classifier takes approximatelly the same time to perform predictions on *all* of the second-level GoMapMan bins as the *bSVM* and *dDNN* classifiers take to perform predictions for a single bin – meaning it performs approximatelly 100 times faster.

Table 3: Results of the best performing classifiers on the top 20 most populated GoMapMan bins.

GMM bin	number of positives	NNMultilabel_1000_0	SVM_1000_0	deeperNN_1000_0
27.3	2952	0.8952	0.8461	0.9089
29.5	2241	0.8705	0.9024	0.8836
20.1	1173	0.8613	0.8660	0.8959
30.2	1061	0.8899	0.9032	0.9030
29.4	1046	0.8741	0.7868	0.8892
29.2	973	0.8268	0.8395	0.8540
28.1	600	0.8501	0.8893	0.8703
33.99	590	0.8660	0.8779	0.8898
26.1	589	0.8794	0.8834	0.8958
31.1	507	0.9000	0.8903	0.9233
20.2	419	0.8969	0.9053	0.9022
26.2	405	0.8604	0.9090	0.8841
27.1	378	0.9098	0.8989	0.9351
29.3	344	0.8592	0.8817	0.8893
30.3	326	0.9337	0.9436	0.9406
17.2	316	0.8926	0.8788	0.9049
28.99	313	0.8881	0.8941	0.9009
30.5	298	0.8595	0.8624	0.8855
1.1	281	0.9271	0.9607	0.9449
27.4	247	0.8724	0.8654	0.8712

4.3 Final results

Concluding this report, Tables 4 and 5 shows the predictions of the DDeMON approach on two GoMapMan bins, 1.1 and 17.5. The tables show a ranked list of genes from GoMapMan bin 35 (unknown function), where a higher ranking means that the gene is more likely to belong to bin 1.1 or 17.5. Observing the tables, we see that the three classifiers we used show a high level of agreement – if a gene is ranked highly by one classifier, its ranks given by the other two classifiers are also high. This, along with high AUC scores demonstrated in previous sections, gives confidence to the predictions made by the DDeMON approach.

References

Dynamic Time Warping, pages 69–84. Springer Berlin Heidelberg, 2007.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

Š Baebler, K Witek, M Petek, K Stare, M Tušek-Žnidarič, M Pompe-Novak, J Renaut, K Szajko, D Strzelczyk-Żyta, W Marczewski, et al. Salicylic acid is an indispensable component of the

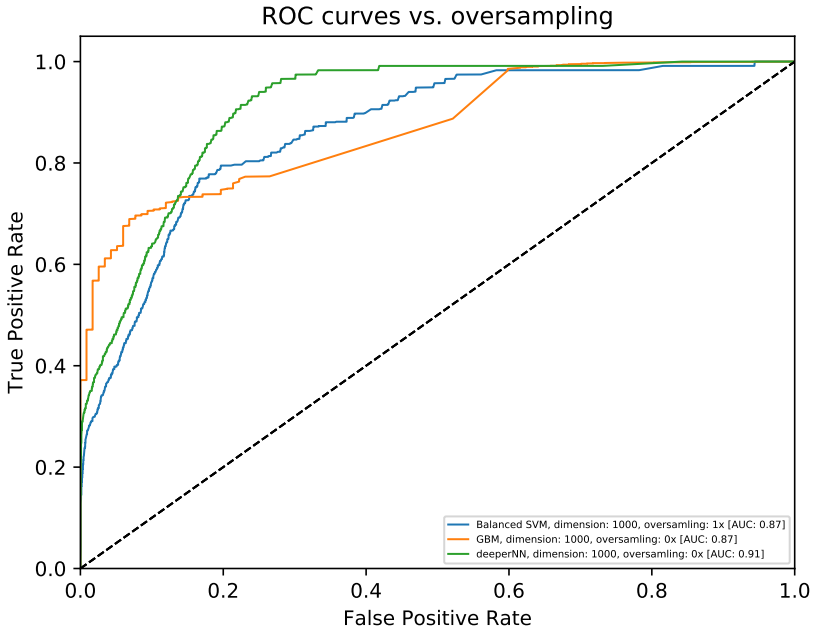


Figure 4: The performance of the three selected classifiers on various GoMapMan bin 20.1.

Ny-1 resistance-gene-mediated response against Potato virus Y infection in potato. *Journal of Experimental Botany*, 65(4):1095–1109, 2014.

Tanya Z. Berardini, Leonore Reiser, Donghui Li, Yarik Mezheritsky, Robert Muller, Emily Strait, and Eva Huala. The arabidopsis information resource: Making and mining the "gold standard" annotated reference plant genome. *genesis*, 53(8):474–485, 2015.

Miha Grčar, Nejc Trdin, and Nada Lavrač. A methodology for mining document-enriched heterogeneous information networks. *The Computer Journal*, 56(3):321–335, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jan Kralj, Marko Robnik-Šikonja, and Nada Lavrač. HINMINE: Heterogeneous information network mining with information retrieval heuristics. *Journal of Intelligent Information Systems*, pages 1–33, 2017.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, November 1999.

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

Table 4: The top 20 genes (sorted by the maximum ranked given to the genes by three classifiers) for the GoMapMan bin 1.1

ID	SVM rank	GBM rank	DNN rank
Sotub01g015190.1.1	27	47	34
Sotub12g022150.1.1	20	52	1
Sotub11g017690.1.1	33	56	21
Sotub12g009780.1.1	32	56	19
Sotub11g020220.1.1	1	58	4
Sotub10g012760.1.1	15	59	26
Sotub12g024790.1.1	13	59	16
Sotub11g006830.1.1	14	59	14
Sotub09g014120.1.1	17	59	18
Sotub10g010920.1.1	10	63	5
Sotub10g010940.1.1	11	63	7
Sotub10g010960.1.1	12	63	8
Sotub03g026560.1.1	35	66	50
Sotub05g012520.1.1	42	67	44
Sotub07g015350.1.1	48	75	45
Sotub08g012010.1.1	40	75	37
Sotub03g033550.1.1	41	75	39
Sotub02g010750.1.1	38	75	35
Sotub12g016200.1.1	25	79	36

Ana Rotter, Björn Usadel, Igor Mozetič, Kristina Gruden, Matej Korbar, Špela Baebler, and Živa Ramšak. GoMapMan: integration, consolidation and visualization of plant gene annotations within the MapMan ontology. *Nucleic Acids Research*, 42(D1):D1167–D1175, 11 2013.

Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

Table 5: The top 20 genes (sorted by the maximum ranked given to the genes by three classifiers) for the GoMapMan bin 1.1

ID	SVM rank	GBM rank	DNN rank
Sotub09g026260.1.1	1	54	1
Sotub07g027010.1.1	2	54	2
Sotub05g022390.1.1	30	62	3
Sotub09g016860.1.1	16	64	27
PGSC0003DMG400011623	25	64	25
PGSC0003DMG400035415	14	64	10
Sotub08g019470.1.1	10	64	14
PGSC0003DMG400045539	11	64	9
PGSC0003DMG400023564	8	64	5
PGSC0003DMG400022155	26	70	28
PGSC0003DMG400046184	9	75	7
PGSC0003DMG400043412	13	76	11
PGSC0003DMG400043564	23	76	23
PGSC0003DMG400044263	19	76	8
PGSC0003DMG400039590	21	76	20
Sotub06g009760.1.1	22	80	22
PGSC0003DMG400013586	29	80	4
PGSC0003DMG400021088	28	82	32
PGSC0003DMG400004046	31	82	12