Exploring transductive and inductive methods for vertex embedding in biological networks

Luca G. Cellamare*, Michele A. Bertoldi*, Alberto Parravicini†, Marco D. Santambrogio†
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milano, Italy
*{lucagiuseppe.cellamare, micheleamedeo.bertoldi}@mail.polimi.it
†{alberto.parravicini, marco.santambrogio}@polimi.it

Abstract— **Proteins perform an innumerable number of functions within living organisms, and the understanding of their interactions is a primary challenge for medicine and biology. The most natural way to represent their structure and their complex interactions are graphs, a powerful data structure to model objects (vertex) and their relations (edges). At the same time, Machine Learning is a powerful tool to extract knowledge and perform complex tasks on proteins. But, to perform machine learning on graphs its first essential to transform the information expressed as a graph into a structure that can be easily exploited by a Machine Learning model. Traditional approaches for learning representations relies on hand-crafted specialized heuristics to extract meaningful information about the entities of a graph, but hand-engineering features can be expensive and time-consuming. In recent years, vertex embedding methods have proven their potential in automatically representing graphs information as feature input for machine learning models. Existing methods can be divided into different groups, such as random-walk based algorithms or graph neural networks. These methods also differ as some approaches take into account just topological information and others can leverage additional features contained in the graph. Within this work, we show how graph embeddings can excel in the real-world task of protein role classification. We also prove how it is possible to combine embeddings from unsupervised models to match or exceed state-of-the-art results obtained by single supervised models, a promising direction of research to obtain protein embeddings able to generalize to a much wider array of tasks.**

Index Terms—**Graph Embeddings, Protein classification, Neural Embeddings**

I. Introduction

Proteins represent the most important class of biomolecules in living organisms. They carry out the majority of the cellular processes and act as structural constituents, catalysis agents and signaling molecules. Proteins are molecular machines of every biological system. For this reason, a comprehensive understanding of their interactions is essential to inspect human diseases, it could elucidate for example the molecular basis of diseases aiding in prevention, diagnosis and treatment [1]. The most natural way to represent their structure and their complex interactions are graphs, a powerful data structure to model objects (vertex) and their relations (edges).

Graphs used to represent protein interactions are particularly complex, due to the interactions between proteins and all the sub-graph structures, the tissues [2].

The roles of proteins range from catalysis of biochemical reactions to transport to signal transduction, and a single protein may play a role in multiple processes or cellular pathways, it is possible to think to a protein function as "anything that happens to or through a protein" [3]. Protein role prediction has important applications in medicine and in biotechnology, such as drug and enzymes design.

The difficulty of protein roles classification makes this task a natural target for Machine Learning (ML) models which seek to be able to use graph-structured data as feature input to make predictions.

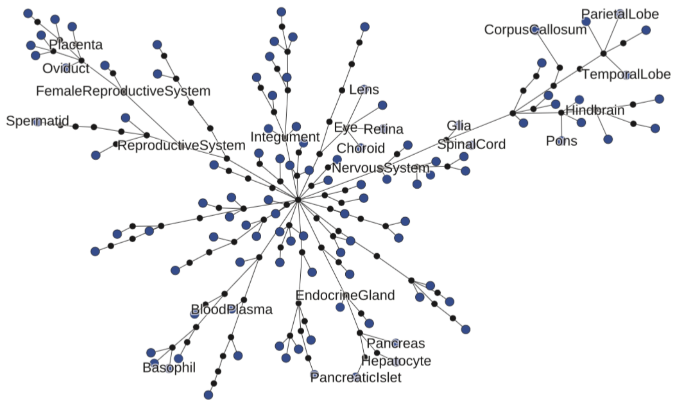

Fig. 1. Graph representation of a tissue hierarchy, from the BRENDA Tissue Ontology [4]. Courtesy of Zitnik et al. [5]

However, ML models have been most successful on data with an underlying Euclidean or grid-like structure (e.g. text or images). This is not the case of graphs: their nature implies that there are no familiar properties such as global parameterization, a common system of coordinates, vector space structure, or shift-invariance. Fundamental operations (e.g. convolutions) that are widely used in the Euclidean domain might not be well defined on non-Euclidean geometries. In fact, there is no straightforward way to encode this non-Euclidean information into a feature vector that can be fed to a model.

To leverage traditional ML models in the field of graph analytics, it is possible to devise techniques that project the non-Euclidean graph structure to a more traditional Euclidean domain. Graph embeddings algorithms compress the graph topology, vertex-to-vertex relationships, and other relevant information (i.e. text attributes, vertex degrees) as low-dimensional dense numerical vectors, which can be used as input of any ML model. The underlying idea is that each vertex is represented as a vector, which encodes the information contained in the vertex. Vertices which somehow carry the same information (i.e. are close to each other, or have similar features), should have a similar embedding, according to an appropriate distance function (e.g. Euclidean distance). There exist countless embedding algorithms that have been proposed in the literature, ranging from techniques based on matrix factorization to random walks and Graph Neural Network (GNN). In our study case, all these families of models should be able to demonstrate inductive capabilities, i.e. be able to predict labels of unseen groups of vertices or entire subgraphs. It is difficult to understand what is the most suitable class of methods and how to configure them optimally. The achievement of a generalization by harnessing heterogeneous data over several dimensions of biological variation, would enable great progress in biology and medicine [2].

In this work, we present the following contributions:

- A review of state-of-the-art vertex embedding algorithms, and show how to leverage the unique characteristics of each algorithm to obtain excellent results in the real-world task of protein role classification, with accuracies exceeding 95%.
- We show how an ensemble model created by combining transductive embedding representations in an unsupervised fashion from different embedding algorithm can easily outperform the single algorithms, and even rival traditional supervised algorithms.

II. State of the Art

The idea of representing data which isn't inherently Euclidean as dense vector has been for a long time a hot topic in the field of ML and Information Retrieval. For example, a milestone in the field of Natural Language Processing is represented by *word2vec* [6], which encodes words as numerical vectors using a simple neural network. The application of embedding algorithms to graphs is more recent, but great progress has been made in the past few years.

Embeddings algorithms can be grouped into two broad categories, depending on how they generate embeddings: *transductives* and *inductives*.

Transductive methods directly learn vertex embeddings for individual vertices and cannot be used to compute embeddings of vertices not seen during the training phase. Generating embeddings of unseen vertices would at the very least require expensive additional training (e.g. via stochastic gradient descent). Transductive methods are principally based on matrix factorization techniques and random walks.

Inductive algorithms, instead, are trained by learning parameters that represent the model that generates the embeddings. In a sense, these algorithms don't learn the embeddings, but they learn how to generate them. Hence, they are able to generate embeddings for vertices not seen during the training phase, and provide much higher flexibility. Most inductive algorithms are based on neural networks, whose parameters are learnt during the training. Historically, graph embedding algorithms have been developed using matrix factorization techniques, random walks techniques, and, more recently, neural networks modified to handle the topology of graphs.

Matrix factorization-based approaches are directly inspired by classic techniques for dimensionality reduction. They usually measure the distance between two matrices as a similarity measure. Factorization between matrices is largely theorized in mathematics but these methods are characterized by high computational costs [7].

Random walks approaches are based on the creation of random sequences of connected vertices. These algorithms were inspired by similar techniques used in Natural Language Processing, such as the aforementioned *word2vec* [6]. The key innovation of these approaches is the optimization of vertex embeddings so that vertices will have similar embeddings if they tend to co-occur on short random walks over the graph. Perozzi et al. (2014) introduced DeepWalk [8], a method that uses local information obtained from truncated random walks to learn latent representations of vertices in a network. DeepWalk has been for years a strong baseline and a source of inspiration for many works in literature [9], [10].

Another field of research has focused on extending neural networks to handle arbitrary structured graphs. Graph Neural Networks (GNNs) were introduced in Gori et al. [11] and Scarselli et al. [12] as a generalization of recursive neural networks (RNNs) that can directly operate on complex graphs. GNNs consist of an iterative process, which propagates the vertices states until equilibrium, followed by a neural network, which produces an output for each vertex based on its state.

Furthermore, most recent approaches effectively generalized convolutional neural networks to the graph domain. These methods are often categorized as *spectral* and *non-spectral* approaches. Spectral approaches work leverage the *eigenvectors* and *eigenvalues* of a graph adjacency matrix and have been successfully applied in the context of vertex classification (2016) [13] to solve the problem of non-spatially localized filters by means of a Chebyshev expansion of the graph Laplacian. This approach removed the need to compute the eigenvectors of the Laplacian and resulted in spatially localized filters. Finally, the previous method was streamlined by Kipf & Welling (2017) [14] in GCN, by restricting the filters to operate in a 1-step neighborhood around each vertex. In all of the aforementioned spectral approaches, the learned weights depend on the graph structure. Thus, a model trained on a specific structure cannot be directly applied to a graph with a different structure, and lacks inductive capabilities.

Non-spectral approaches define convolutions directly on the graph operating and are able to recognize structural properties of a vertex neighborhood that reveal both the vertex local role in the graph, as well as its global position. Hamilton et al. (2017) [15] introduced GraphSAGE, a method for computing vertex representations in an inductive manner. The key idea behind this model is to learn how to aggregate feature information from a vertex local neighborhood (e.g. the degrees or text attributes of nearby vertices). Further improvements have been reached through attention mechanisms during the propagation step. The main intuition behind attention strategy is that it allows for identifying the most relevant parts of the input. GAT, introduced by Velickovic et al. (2018) [16], does not require to sample neighborhoods and it is able to assign arbitrary weights to different neighbors, achieving state-of-the-art performances on a variety of tasks.

III. Methods

In this work, we focus our attention on the task of predicting protein roles, which can be modeled as a multi-label vertex classification problem.

This problem can be modeled in the following way: the input is composed by the graph topology, denoted with $\mathcal{V}$ and E, the set of N vertices and the set of edges, and when possible also the feature matrix X, and we obtain as output the predicted labels. To accomplish this task we used two types of embedding algorithms, *transductive* and *inductive*. In the *transductive* setting we obtain the embedding vectors in an unsupervised manner for every protein/vertex and we then feed these encoded representations to a classifier which is able to produce as output the predictions, after being trained in a supervised way leveraging the known labels. In the *inductive* case instead, we apply a supervised learning end-to-end process making use of classification labels in order to optimize the embeddings and to directly obtain predicted labels. The embedding algorithm doesn't simply learn the embeddings, but learns the parameter required to compress the input vertices and features and create embeddings as output: as a result, inductive algorithms can be used to learn embeddings of vertices not seen during the training.

Following the encoder-decoder framework introduced by Hamilton et al. [15], we describe vertex embedding methods around two key components: an encoder that maps each vertex and its additional features to a low dimensional vector or embedding vector, and a decoder, which decodes global positions of vertices and local neighborhoods in the graph from the embeddings. Within the class of *transductive* methods we considered DeepWalk [8] and Node2Vec [10]. Both methods leverage the ideas behind *word2vec* [6] to obtain numerical representations of vertices starting from random walks, i.e. sequences of vertices (*word2vec*, instead, learns representations of words starting from sentences, i.e. sequences of words). The key idea behind these methods is to learn embeddings so that

$$DEC(\mathbf{z}_i, \mathbf{z}_j) \triangleq \frac{e^{\mathbf{z}_i \mathbf{z}_j}}{\sum_{v_k \in \mathcal{V}} e^{\mathbf{z}_i \mathbf{z}_k}} \approx p_{\mathcal{G},T}(v_j | v_i) \tag{1}$$

where $p_{\mathcal{G},T}(v_j|v_i)$ is the probability of visiting v_j on a length-T random walk starting at v_i, with T usually defined to be in the range $T \in \{2, \ldots, 10\}$. We also considered a method which defines random walks in more flexible ways. Walklets [9], for example, is able to *skip* steps in each random walk. It is capable of generating a corpus of vertex pairs which are reachable via paths of a fixed length, which can then be used to learn a series of latent representations, each capturing higher order relationships from matrix adjacency A.

Recent node embedding approaches make use of encoders that rely on a vertex local neighborhood, but not necessarily on the entire graph. Intuitively, these algorithms create embeddings for a node by aggregating information from its local neighborhood. Compared to already discussed methods, these algorithms also take into consideration the feature matrix X to create the embeddings vector.

Among the inductive embedding algorithms, we experimented with a semi-supervised neural network model $f(X, A)$ GCN of Kipf et al. [14] which defined the propagation rule

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{2}$$

where $\tilde{A} = A + I$ and $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$, $W(l)$ is a weight matrix for the l-th neural network layer $H(0) = X$ and $H(L) = Z$ is the output embedding vector.

We also used in both supervised and unsupervised fashion the GraphSAGE algorithm proposed by Hamilton et al. [15]. The intuition behind GraphSAGE is that at each iteration, vertices aggregate information from their local neighbors as

$$H^l_{N(v)} \leftarrow AGGREGATE^l(\{H^{l-1}_u,\ \forall u \in N(v)\}) \tag{3}$$

We considered both the **GraphSAGE-LSTM** and **GraphSAGE-pool** variants. The first apply an LSTM aggregator to a random permutation of the neighbors of a vertex. In **GraphSAGE-pool**, instead, the neighbor vector is independently fed through a fully-connected neural network. Following this transformation, an element-wise max-pooling operation is applied to aggregate information across the neighbor set. After the aggregation of neighboring feature vectors, GraphSAGE concatenates the vertex current representation. Then, the representation is fed to a fully connected layer with nonlinear activation function which computes the representations to be used at the next step of the algorithm. The last embedding algorithm that we tested is GAT (Graph Attention Network) [16]. This algorithm implements a shared attention mechanism on vertices of form

$$e_{ij} = a(W\vec{h}_i,\ \ldots,\ W\vec{h}_j) \tag{4}$$

where e_{ij} represents the importance of node js features to node i. We refer the reader to the original papers and to [17] for a comprehensive understanding of the aforementioned methods.

IV. Experimental Results

In this section, we present our testing methodology, provide details about the hyper-parameters used in each graph embedding algorithm, and measure the performance of each algorithm. We compare the performance of both transductive and inductive models, and show how to build ensembles of unsupervised embedding algorithms.

In our case study, we consider the dataset of multi-cellular function through 107 layers networks originally presented in the work of Zitnik et al. [5]. This dataset was obtained by the mapping of tissues in the Human Protein Reference Database (HPRD) [18] to tissues in the BRENDA Tissue Ontology [4] from Greene et al. [19]. The dataset is composed of 24 graphs, 56944 vertices and 818716 edges and contains positional gene sets, motif gene sets and immunological signatures as features (50 features in total) and 121 gene ontology sets as labels[1]. Note that each vertex could have in principle any of the 121 labels, and the number of labels of a given vertex is not known in advance. Multi-label classification problems such as this are usually considered very challenging, and special care must be taken to obtain satisfactory results.

The performance of the different algorithms is measured using Micro-averaged and Macro-averaged F1 score. The F1 score is an accuracy metric computed as harmonic average of precision P and recall R. Precision is the ratio of true positives (TP) to all predicted positives (TP + FP). Recall is the ratio of true positives to all actual positives (TP + FN). The F1 score weights recall and precision equally, and moderately good performance on both will be favored over extremely good performance on one and poor performance on the other. In the context of multi-label classification, we compute the average F1 score for each protein. This metric is referred to as Mean-F1 score or Micro-averaged F1 score. Macro-F1, instead, averages results across all classes, so classes that are very rare will have a higher impact than they should. Micro-F1 aggregates all results, so rare classes have a lower impact on the final score. As a result, Macro-F1 tends to be lower than Micro-F1 when computed on the dataset we considered.

In our analysis, we evaluate both transductive and inductive embedding algorithms. Transductive algorithms, i.e. DeepWalk, Walklets, node2vec and GraphSAGE-pool, are trained in a fully unsupervised way, using both the graph topology and the vertex features (when supported by the algorithm, such as in the case of GraphSAGE-pool). DeepWalk was set with a layer size of 128, learning rate of 0.05, walk length of 10 and 15 walks per vertex. Walklets was set with a reduced walk length of 10, walk-number per vertex 80, **P** and **Q** (hyperparameters for second order walks) equal respectively to 4 and 0.25 and the output embedding dimension was increased to from 160 to 200 to obtain better performance. Node2vec was set with a walk length of 12, walk-number per vertex 10, **P** equal to 1 and **Q** equal to 5. GraphSAGE-pool was trained in the unsupervised setting with 2 layers of equal output size of 256 learning rate of 0.00005 and a batch size of 512 for the training set and 256 for the validation one. We also set a number of samples equal to 30 in layer 1 and 10 in layer 2, the model was trained with a dropout value equal to 0.1 and maximum node degree of 128.

These algorithms create vertex embeddings without knowledge of the labels that should be predicted. These embeddings could in principle be used for other tasks, and are more general, in a sense. On the other hand, transductive algorithms are unable to easily create embeddings of unseen vertices, and their training is in general more time-consuming. Embeddings created by transductive algorithms can be stored in a file, and used as input of a supervised classifier whose output is the set of labels to be predicted. Given the multi-label nature of our task, we opted for a simple multi-label SGD classifier, a linear classifier with stochastic gradient descent training. We experimented both with a logistic regression and with a linear support vector machine. In practice, we train a different classifier for each of the 121 labels. One might assume that different labels are correlated, but even with independent classifiers it is possible to obtain reasonable accuracy values, as shown in table I. In both cases we set 1000 as the number of maximum iterations and 10^{-5} as stopping criterion.

[1]https://s3.us-east-2.amazonaws.com/dgl.ai/dataset/ppi.zip

TABLE I
PERFORMANCE OF DIFFERENT EMBEDDING ALGORITHMS ON THE PPI DATASET

Category	Method	Micro-F1	Macro-F1
NA	**Features only**	0.335	0.116
Unsup.	**DeepWalk**	0.435	0.127
Unsup.	**Walklets**	**0.502**	0.391
Unsup.	**node2vec**	0.432	**0.397**
Unsup.	**GraphSAGE-pool**	0.444	0.266
Sup.	**GraphSAGE-LSTM**	0.615	0.453
Sup.	**GCN**	0.498	0.372
Sup	**GAT**	**0.978**	**0.966**
Ensemble	**(1) DeepWalk+SAGE-pool+feats**	0.495	0.214
Ensemble	**(2) Walklets+SAGE-pool+feats**	**0.557**	**0.417**

Highlighted, the best Micro and Macro F1 results obtained by unsupervised and supervised methods

Inductive algorithms, on the other hand, are able to create embeddings of unseen vertices, and can in general be trained in a supervised fashion. In this work, we train inductive algorithms using just a subset of the available protein interaction graphs, and use the remaining subgraphs as validation set and training set. Of the 24 graphs that compose our dataset, we use 20 graphs for training, 2 for the test set and 2 for the final validation set used to measure the values reported in table I.

Among the inductive algorithms that we tested, GraphSAGE-LSTM was set identically to GraphSAGE-pool except for the batch size which was 256 in this case. GCN was trained with 48 hidden parameters and final layer output of 120. The final output is then fed to a fully connected layer with size 121 and logistic activation to obtain the label predictions. The algorithm that performed the best on the PPI dataset is GAT, trained with the original configuration presented by Velickovic et al. [16]. GAT is trained with 2 layers each with 256 hidden units, 4 attention multi-heads for the first 2 layers and 6 for the third, and a learning rate of 0.005. We trained GAT over 50 epochs of training using different numbers of hidden units, to better understand how the size of a model influences performance in accuracy. As is possible to see in fig. 2 and fig. 3, the level of Micro-F1 score is very different based on the size of the hidden units. With an extreme reduction of parameter dimension, the model, despite a faster execution speed, not very significant for our task, gets significantly lower Micro-F1 score. Still, using only 72 hidden units results in almost 90% Micro-F1 score, not far from the 97% obtained by the best performing model.

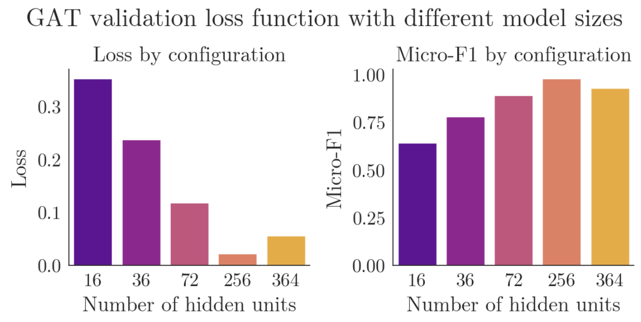

Fig. 2. GAT Micro-F1 and loss at 50 epochs of train with different number of hidden units. It can be seen how the best results are achieved at 256 units: larger models are likely to overfit, or be unable to reach the same accuracy in a limited number of iterations

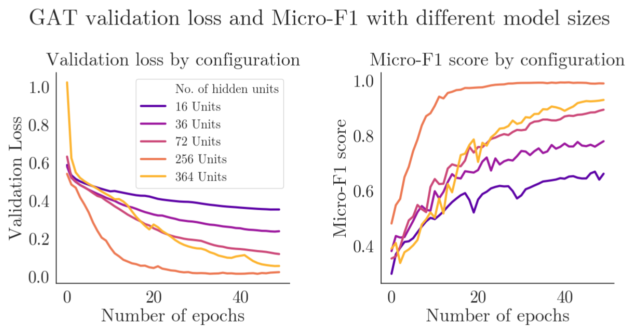

Fig. 3. GAT Micro-F1 and loss during the training, using different number of hidden units. The best results are achieved with 256 units, even though the large model might be able to reach lower loss if trained for more iterations

As is it possible to infer from table I, supervised methods clearly outperform unsupervised ones, taking advantage of known labels to optimize the embedding representations. However, the necessity of handle heterogeneous data across several dimensions of biological variation [2] means that there is a great need of the ability to generalize given by unsupervised methods. Hence, we have combined the embeddings given in output by unsupervised models, combining Walklets and DeepWalk, methods that take into consideration strictly topological information, with unsupervised GraphSAGE, a very different method that leverages features in the computation of embeddings. The idea is to create a larger ensemble model by leveraging representations of the same vertex generated by different methods, to form a richer and more comprehensive embedding vector for each vertex.

Combining the 128 sized embedding representations of DeepWalk and GraphSAGE-pool concatenated with the features, we reach a Micro-F1 score of 0.495 with an SGD classifier using as loss function a logistic regression, 500 epochs of training and 10^{-4} as stopping criterion.

Concatenating embeddings vector with a size of 200 from Walklets and GraphSAGE-pool, plus the vertex features we reach a Micro-F1 score of 0.557 during test inference. Those values achieve or match state-of-the-art results from unsupervised models and even exceed the result of GCN, a supervised model. Indeed, the ensemble model obtained an improvement of 6% over basic unsupervised methods. Moreover, as an indicator of robustness, ensemble methods never performed worse than their original components. Results are presented in fig. 4 We also noticed that normalizing the embeddings in the $[0, 1]$ or $[-1, 1]$ range can improve Micro-F1 scores by around 2%. Reducing the concatenated embedding dimensionality with techniques such as Principal Component Analysis (PCA) gave no significant gain in our experiments. The code used for our experiments is available online[2].

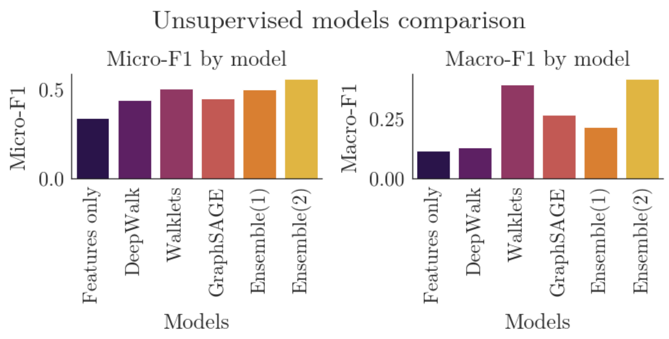

Fig. 4. Comparison between the results obtained by ensembles and single unsupervised methods. Ensemble obtains improvements of around 6% of Micro-F1 score compared to their components. **Ensemble(1)** is **DeepWalk + GraphSAGE-pool + features**, while **Ensemble(2)** is **Walklets + GraphSAGE-pool + features**

V. Conclusion and future work

An ideal Machine Learning method needs to answer biological or medical questions, identify important features and predict outcomes of heterogeneous data across several dimensions of biological variation. In the quest to reach a method with such power and flexibility, we have shown how graph embedding algorithms are able to obtain excellent results in the complex task of multi-label protein role classification and they will become increasingly important in modern biology. In the case study we have considered, supervised methods were able to achieve more than 95% Micro-averaged F1. Unsupervised methods gave less accurate predictions, but their capability is often limited by the amount of data at our disposal. We have demonstrated how it is possible to improve the performance of unsupervised embedding algorithms by leveraging ensemble techniques. Being able to improve the quality of our model by considering representations based on different characteristics is an interesting line of research to be investigated even further.

[2]https://github.com/LucaCellamare/PPI-classification-python

References

[1] A. Griffa, P. S. Baumann, C. Ferrari, K. Q. Do, P. Conus, J.-P. Thiran, and P. Hagmann, "Characterizing the connectome in schizophrenia with diffusion spectrum imaging," *Human brain mapping*, vol. 36, no. 1, pp. 354–366, 2015.

[2] M. Zitnik, F. Nguyen, B. Wang, J. Leskovec, A. Goldenberg, and M. M. Hoffman, "Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities," *Information Fusion*, vol. 50, pp. 71–91, 2019.

[3] B. Rost, J. Liu, R. Nair, K. O. Wrzeszczynski, and Y. Ofran, "Automatic prediction of protein function," *Cellular and Molecular Life Sciences CMLS*, vol. 60, no. 12, pp. 2637–2650, 2003.

[4] M. Gremse, A. Chang, I. Schomburg, A. Grote, M. Scheer, C. Ebeling, and D. Schomburg, "The brenda tissue ontology (bto): the first all-integrating ontology of all organisms for enzyme sources," *Nucleic acids research*, vol. 39, no. suppl_1, pp. D507–D513, 2010.

[5] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[7] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web - WWW '13*. ACM Press, 2013. [Online]. Available: https://doi.org/10.1145%2F2488388.2488393

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[9] B. Perozzi, V. Kulkarni, and S. Skiena, "Walklets: Multiscale graph embeddings for interpretable network classification," *arXiv preprint arXiv:1605.02115*, 2016.

[10] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[11] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] W. L. Hamilton, Z. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, pp. 52–74, 2017.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *CoRR*, vol. abs/1901.00596, 2019.

[18] T. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal *et al.*, "Human protein reference database2009 update," *Nucleic acids research*, vol. 37, no. suppl_1, pp. D767–D772, 2008.

[19] C. S. Greene, A. Krishnan, A. K. Wong, E. Ricciotti, R. A. Zelaya, D. S. Himmelstein, R. Zhang, B. M. Hartmann, E. Zaslavsky, S. C. Sealfon *et al.*, "Understanding multicellular function and disease with human tissue-specific networks," *Nature genetics*, vol. 47, no. 6, p. 569, 2015.