Predicting protein-protein interactions with protein embeddings

Bioinformatics - Group 10

Alexandre Coelho
MSc Bioinformatics and Computational Biology
fc50989@alunos.fc.ul.pt

Carolina Vidal
MSc Bioinformatics and Computational Biology
fc55783@alunos.fc.ul.pt

Lucas Silva
MSc Computer Science and Engineering
fc55922@alunos.fc.ul.pt

Rodrigo Cassanheira
MSc Bioinformatics and Computational Biology
fc55781@alunos.fc.ul.pt

Romana Esteves
MSc Bioinformatics and Computational Biology
fc55965@alunos.fc.ul.pt

1 Introduction

High-developed techniques in genomics and proteomics generate large amounts of data about the function, regulation and interaction of genes and proteins[1]. Detecting protein-protein interactions (PPI) is very important to generate knowledge and a better understanding of the protein function in biological processes in normal and disease states, including cell cycle progression, DNA replication and signal transmission. This helps, for instance, in therapeutic target identification[2,3].

In recent years, to detect PPIs, some high-throughput technologies were developed such as yeast two-hybrid screens (Y2H), tandem affinity purification (TAP), and mass spectrometric protein complex identification (MS-PCI), Tandem Affinity Purification and Mass Spectrometry (TAP-MS), affinity chromatography and Co-Immunoprecipitation (Co-IP). These methods allowed the growth of the number of PPIs but these methods have some disadvantages as the update from data, and high false positives and false negative rate[4]. Compared with these methods, the computational method has advantages such as fast verification, speed and strong repeatability, high quality and accuracy[5]. In this project we used machine learning methods.

As inputs to the machine learning models, we used embeddings. An embedding is a mapping of a discrete — categorical — variable to a vector of continuous numbers[6], these vectors in a low dimensional space capture the relationships in knowledge graphs (KG)[7]. A knowledge graph is made of nodes and edges. Each edge connects a pair of nodes indicating that there is a relation between them[8]. Resuming, the proteins are represented as vectors in the embedding space ready to be used for machine learning models such as prediction and node classification[9].

In this project, we started by testing different machine learning models: Decision Trees, Random Forests, Gaussian Naive Bayes, Multi-Layer Perceptron and Support Vector Machines. All of these machine learning models are supervised, and this feature requires labels. All the necessary information was extracted from a dataset provided to us[1]. After verifying which one got the best parameters using the most embracing dataset, we chose the best one to go on.

Our goals include seeing which is the best and most efficient supervised machine learning algorithm, which is the best embedding combination operation, and the main goal is the prediction protein-protein interaction using a learning machine model.

CCS CONCEPTS

• Machine Learning • Network Embeddings • Supervised Classification

KEYWORDS

Protein-Protein Interactions, *D. melanogaster*, *E. coli*, *H. sapiens*, *S. cerevisiae*, Embeddings, Supervised Classification

ACM Reference format:

Alexandre Coelho, Carolina Vidal, Lucas Silva, Rodrigo Cassanheira and Romana Esteves. 2020. Predicting protein-protein interactions with protein

embeddings. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 6 pages.* https://doi.org/10.1145/1234567890

2 Materials & Methods

2.1 Tools Used

For the realization of this project, we made use of the *Python* language and we had *scikit-learn*, *NumPy* and *Pandas* as our only dependencies, which makes it possible to use machine learning algorithms, essential to solve the problem mentioned earlier. We also made use of the resources provided by *Google Colab*, in order to carry out this project efficiently, since the notebook can be shared among several people.

2.2 PPI and Embedding Datasets

For this project we had two different types of datasets provided to us: one with the protein-protein interactions[1] that occur and do not occur and the other with the vector embeddings for each protein, both of them with datasets from four different species (*Homo sapiens*, *Saccharomyces cerevisiae*, *Escherichia coli* and *Drosophila melanogaster*) and the fifth one (All) with data from all these species.

Table 1. Proteins and Protein-Protein Interactions present in each dataset for each species.

	One aspect		All aspects	
	Proteins	Pairs	Proteins	Pairs
D. melanogaster	455	364	287	200
E. coli	371	734	263	420
H. sapiens	7093	30826	6718	29672
S. cerevisiae	3776	27898	2888	16904
All	11695	59822	10156	47196

Regarding the PPI interaction datasets, the data is also divided regarding the Gene Ontology (GO) aspects. GO is divided into three smaller branches that characterize a protein: Molecular Function, Cellular Component and Biological Process. With this, the PPI datasets either have proteins with one annotation of each GO aspect and at least one aspect with one leaf-class annotation (One aspect, e.g., *All1*) or proteins with one annotation of each GO aspect and each aspect with at least one leaf-class annotation (All aspects, e.g., *All3*). For each entry of any of these datasets we have a pair of proteins together of a binary representation of the PPI, i.e., 1 if the proteins interact and 0 if the proteins do not interact. The number of proteins and the number of PPI of each dataset is present in Table 1.

Regarding the embeddings, we do not have the separation of One and All aspects since the proteins of All aspects are included in the One aspect datasets. The embeddings themselves are vectors of 200 dimensions that represent each protein of each dataset.

2.3 Embedding Combination Operations

Different operations were performed on the embeddings of the interacting proteins, i.e., for each pair of proteins, an operation was performed in order to combine their pair of embeddings. These operations were Hadamard, min, max, concatenation and addition. In order to compute them we used the NumPy library.

2.3.1 *Hadamard product.* The Hadamard product, also known as the element-wise product, is a binary operation that takes two matrices of the same dimensions and produces another matrix of the same dimension as the operands, where each element i, j is the product of elements i, j of the original two matrices.

2.3.2 *Min.* The Min compares the two embeddings and returns a new embedding containing the element-wise minima.

2.3.3 *Max.* The Max compares the two embeddings and returns a new embedding containing the element-wise maxima.

2.3.4 *Concatenation.* Concatenation joins the two embeddings returning a new embedding containing the two embeddings.

2.3.5 *Addition.* The Addition adds the two embeddings element-wise.

2.3.6 *Mean.* The Mean computes the arithmetic mean which is the sum of the elements along the axis divided by the number of elements.

2.4 Supervised Machine Learning Models

In order to create models to classify the data available, we decided to use several algorithms shown below. Given that these are essential for this type of classification problems.

2.4.1 *Decision Trees.* A decision tree is a non-parametric supervised learning method that uses tree-like structures composed of decision rules (nodes), that are applied recursively to a feature space of a dataset forming branches until reaching the terminal nodes, i.e., the class labels[10].

2.4.2 *Random Forest.* A Random Forest is an automatic learning algorithm that operates by building a combination of decision trees, to perform classification and regression tasks. That is, the algorithm creates several decision trees and combines them in order to obtain predictions with greater accuracy and precision[10].

2.4.3 *Gaussian Naive Bayes.* Naive Bayes is based on Bayes' theorem and follows the Gaussian normal distribution. This model can be adjusted by finding the mean and standard deviation from the points inside each label. In each point, the Gaussian Naive Bayes classifier is calculated using the distance of the mean of each class, which is divided by the standard deviation of that class[11].

2.4.4 *Multi-Layer Perceptron.* A Multi-Layer Perceptron consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training[12].

2.4.5 *Support Vector Machines.* Support Vector Machines approach is to try to find a hyper-plane that creates a boundary between data categories. This is done by plotting the data into a N-dimensional space and maximizing the distance between both classes with a set of hyper-planes[12].

2.4.6 *Differences between the models.* In order to compare the models described above, we can start by comparing the most similar, decision trees and random forest. Decision Trees are usually simpler to understand and implement however, they can generate complex trees that do not generalize well and are unstable. Whereas Random Forest models are less prone to overfitting and, in most cases, more accurate than Decision Tree models. However, their implementation is more complex, and their predictions are slower.

Furthermore, Gaussian Naive Bayes models, unlike Random Forest models are extremely fast and in comparison, with decision trees require smaller training datasets. They do, on the other hand, assume the independence and the equal importance of the features, assumptions that are not always verified.

Multilayer Perceptrons are distinguished by their ability to learn models in real-time. Notwithstanding, these models are sensitive to feature scaling, require adjusting several hyperparameters and have a varying validation accuracy.

Lastly, and contrary to the models mentioned before, Support Vector Machine Models aren't appropriate for non-linear problems, nor for a large number of features. Still, these models are not biased by outliers, do not suffer from overfitting problems, and are quite memory efficient.

2.5 Evaluation Metrics

While training the models, there needs to be a way to evaluate their performance. Therefore, the next metrics are used to verify if the models obtained really are efficient and that they can predict correctly the data provided.

2.5.1 Accuracy. This metric is one of the most intuitive performance measures, and basically provides information about the ratio of correctly predicted instances to the total number of instances.

2.5.2 Precision. Ratio of correctly predicted positive instances to the total of positive instances.

2.5.3 Recall. Represents the ratio of correctly predicted positive observations to the total of observations in the real class.

2.5.4 F1 Score. This metric is the weighted average of Recall and Precision, usually, it is one of the most valuable to help evaluate the models generated. Principally in cases where the datasets have an uneven class distribution.

For the evaluation of the models obtained, the metric mainly used during the project realization was accuracy. However, having high accuracy does not mean that the model is performing well. However, the dataset used in this project is symmetric, where the values of the target class (Interaction) are evenly distributed. Therefore, having high accuracy represents that the model does have a high performance.

2.6 Training and testing models across the same species

When training the models, firstly we decided to train and test All1 dataset with all five classifiers: Decision Trees, Random Forest, Gaussian Naive Bayes, Multi-Layer Perceptron, and Support Vector Machines. However, in the other species datasets, we only trained for the best three models since we noticed that the better scores came from the classifiers Random Forest, Multi-Layer Perceptron and Support Vector Machines.

We then proceeded to use *StratifiedKFold* to shuffle the data provided, and split it into several parts, in the case of this project, the number of splits chosen was 10. Having divided the data into several training and test sets, it is time to train the model, using the training set and then evaluating the model on the test set, this process repeats for the number of

splits we have. This process is done for each embedding combination operation, so at the end we have five different models trained by the data of the same species.

We also decided to maximize the scores of the Random Forest classifier. In other words, we computed the default Random Forest, without any parameters, for the All1 dataset, then we compared the scores with the best parameters found with GridSearchCV, which does an exhaustive search over specified parameter values. Considering this, we used the parameters outputted by GridSearchCV for all the next Random Forest classifiers in all other species datasets.

2.7 Training and testing with different species

We also decided to check the ability of a model to make predictions about a species when trained with a different species. Following our previous results, we used the Random Forest algorithm trained for each individual species and then used another's species dataset to test and be made predictions of.

2.8 Training with pairs of species

Lastly, we aimed to test a model's performance when training it with two different species and testing it in a third one. Correspondingly to what was done prior, the Random Forest algorithm was employed.

3 Results & Discussion

3.1 Best Algorithm for each Species

Table 2. Best Classifiers for each dataset. For each species, the accuracy scores for the train and test are also present.

Species	Acc. Train	Acc. Test	Classifier
D. melanogaster	1.0	0.973	Random Forest
E. coli	1.0	0.9324	Multi-Layer Perceptron
H. sapiens	0.831	0.7733	Support Vector Machines
S. cerevisiae	0.8865	0.8548	Support Vector Machines
All	0.89	0.7277	Random Forest

When computing the best algorithm for each species, we determined that the operation is independent when comparing algorithms. That said, it made sense to only compute the same operation, in this case, Hadamard, for the best three algorithms: Random Forests, Multi-Layer Perceptron and Support Vector Machines.

Table 2 shows us the results of the best performing algorithm regarding accuracy for each species. By analyzing the results, we noticed that Support Vector Machines is the best algorithm in general, since the difference between the accuracy of train and test is smaller than Random Forests, in other words, it's less likely to overfit.

In general, the differences in scores between the top three models were minor. The major differences came in computational time, where Support Vector Machines underperformed in comparison with the others. Multi-Layer Perceptron performed the best in EC1 (*E. coli*) dataset, however in the other species datasets it gave high accuracy in train and low scores on accuracy in test and on precision, making it an inadequate algorithm for predicting PPIs across different species.

This insight made us realize that the best classifier, in general, would be the Random Forest.

3.2 Best Combination Method for each Species

For the different combination methods used to calculate the embeddings, we proceeded to obtain the values for all the combinations for each species. With that, the best one was the *Hadamard,* as seen in Table 3 and always followed by either the *Max.* or *Min. combinations*.

Table 3. Best embedding combination method for each species. For each species, the accuracy scores for the train and test are also present. The Random Forest algorithm was used for all the predictions.

Species	Acc. Train	Acc. Test	Operation
D. melanogaster	1.0	0.973	Hadamard
E. coli	0.997	0.8919	Hadamard
H. sapiens	0.9163	0.7473	Hadamard
S. cerevisiae	0.9273	0.8473	Hadamard

The models obtained show good results, and interestingly the best embedding combination operation is the *Hadamard*, we believe this is the case because of the way used to calculate these when using both vectors to obtain one with the same dimension, it can catch similarities between values, for example, big * big = bigger, small * small = smaller. For the discrepancies, when a negative number is multiplied with a positive number, the result is negative. Being able to identify these, helps the model predict correctly.

However, this seems to lead to a more rigid model that achieves good results within its species, but we also want to

test it on different species to understand if the models are capable of generalizing to different species.

3.3 Predicting PPIs across different species

Table 4. Results of predicting PPIs with a model trained with different species (of the One Aspect datasets). For each pair, the accuracy of the prediction is present along with the best method of combination. Random Forest algorithm was used for all predictions.

Train Species	Test Species	Operation	Accuracy
D. melanogaster	*E. coli*	Hadamard	0.827
	S. cerevisiae	Hadamard	0.7065
	H. sapiens	Max	0.5996
E. coli	*D. melanogaster*	Hadamard	0.8846
	S. cerevisiae	Hadamard	0.7754
	H. sapiens	Hadamard	0.6554
H. sapiens	*D. melanogaster*	Max	0.783
	E. coli	Min	0.7602
	S. cerevisiae	Hadamard	0.7458
S. cerevisiae	*D. melanogaster*	Min	0.7637
	E. coli	Hadamard	0.7452
	H. sapiens	Hadamard	0.6848

Regarding the species comparison, our *a priori* knowledge of the species and the machine-learning models made us expect two things: more closely related species should have higher accuracy models, and species with a larger dataset as training data should provide the best results (the number of proteins and PPI is present in Table 1). Regarding the species, we know that *D. melanogaster* and *H. sapiens* are pluricellular organisms while *S. cerevisiae* and *E. coli* are unicellular. *E. coli* is also the only prokaryotic organism. Besides the accuracy of the predictions, another interesting factor to explore is what is the best method of embedding combination for a given set of two species.

Table 4 shows us the results of these combinations for the One aspect datasets. Firstly, it is worth noting that *D. melanogaster* together with *E. coli* is the pair with the better results (with either one serving as training data) and that the worse results are obtained when we train the models with either *D. melanogaster* or *E. coli* and test with *H. sapiens.* The first observation came as a surprise since *E. coli* and *D. melanogaster* are relatively different organisms. However, these two have a similar number of proteins and PPI that could explain the good results observed. Another interesting analysis would be directly checking the common PPI of both species, yet this analysis is out of the scope of this work. The fact that *E. coli* and *D. melanogaster* produce the worst results when predicting *H. sapiens* PPIs was already expected due to the huge difference in the numbers of PPI that these species. Since *H. sapiens* has so much more PPIs, it is difficult to train a model that can predict such a high number of interactions with only a fraction of interactions being used to train it. The results of training a model with *H. sapiens* and then predicting the PPI of *E. coli* and *D. melanogaster* go in line with this theory since the larger number of train proteins increase the ability of the model to predict new interactions, furthermore, there is also a big chance that a good part of the PPI of these species have *H. sapiens* correspondents.

For the vector combination operations, the *Hadamard* multiplication is the most common method, although, there is also the presence of the *Min* and *Max* methods. One important thing to discuss is why these methods provide the best results, since, as we previously saw, Hadamard provided the best results for all the models within the same species. Our hypothesis relies on the fact that these methods are the best in species pairs that either have very dissimilar species or have a large discrepancy in the number of PPI. Contrary to the *Hadamard* product, each dimension of the resulting vector embeddings of both *Min* and *Max* methods does not rely on both the proteins of the interaction, which can lead to a relaxation of the model and therefore lead to better scores predicting interactions of species not so related and with a big difference in the number of proteins.

In Table S 1 we have the best scores with the All annotations dataset. Overall, in the predictions one interesting behaviour was found, with the best scores of Table 4 decreasing and the lowest ones increasing, but with the main conclusions still remaining the same. This was somewhat expected due to the lower scores observed with the predictions within each species, but also due to the best annotation of proteins that can increase the accuracy in some species. Regarding the

vector embeddings combination methods, we can see the presence of more methods different than Hadamard, but still all of the different ones being either Min or Max.

3.4 Training with two species and testing with a different one

Table 5. Results of predicting PPIs with a model trained with a pair of species (of the One Aspect datasets). For each pair, the accuracy of the prediction is present along with the best method of combination. Random Forest algorithm was used for all predictions.

Train Species	Test Species	Operation	Accuracy
D.melanogaster and *E.coli*	*H. sapiens*	Hadamard	0.6466
	S. cerevisiae	Hadamard	0.758
D.melanogaster and *H.sapiens*	*S. cerevisiae*	Hadamard	0.758
	E. coli	Hadamard	0.8106
D.melanogaster and *S.cerevisiae*	*H. sapiens*	Hadamard	0.6571
	E. coli	Hadamard	0.7752
E. coli and *H. sapiens*	*D.melanogaster*	Min	0.6593
	S. cerevisiae	Hadamard	0.7237
E.coli and *S.cerevisiae*	*D.melanogaster*	Max	0.783
	Homo sapiens	Hadamard	0.6571
H.sapiens and *S.cerevisiae*	*D.melanogaster*	Max	0.783
	E. coli	Hadamard	0.7752

Subsequently, two species are used to train the model, then we proceed to test said model with the remaining species. In Table 5 shown above, we see the results of each trio, (for example, we trained a model with *D. melanogaster* and *S. cerevisiae*, and then tested it in *H. sapiens* and *E. coli*).

Following the *a priori* knowledge described above, it would be expected that by combining the most differently related species we would obtain models that generalize better for the remaining species. The complexity of the organisms should also be taken into consideration, so it is somewhat expected that when we test models in *H. sapiens*, which is a much more complex organism when comparing with the other three in study, we will obtain worse results.

However, as we can see from the findings in the last section, the results are greatly affected by how large the dataset used for training is.

As observed, the best result belongs to the test in E. coli when trained in *D. melanogaster* and *H. sapiens*, in general, *E. coli* has good results, which was expected from the smallest dataset because the training is always done with bigger datasets. The lowest accuracy appears when we test in *H. sapiens* after training in *D. melanogaster* and *E. coli*, this was already expected since we used the two smallest datasets and tested in the biggest one. It is also observed that in general, when tested in *H. sapiens* the result is lower when compared to the other tested species due to the same reason.

Finally, when we train in *S. cerevisiae* and *H. sapiens* we are using the two biggest datasets and then we test in the two smallest datasets (*E. coli* and *D. melanogaster*) the results are better comparing when we trained with both the smallest datasets and test in the two biggest ones.

Regarding the vector combination operations as previously seen, the *Hadamard* product obtains the best overall results. The exceptions being when the models are tested on *D. melanogaster*, where the best performances are found when using the *Min* or the *Max* combinations.

4 Conclusions

In summary, the models obtained for each species had as the best combination operation the *Hadamard*, we found this to be quite interesting, for species that differ from each other, obtained the best results through the same operation. This also applies to most of the cases when training and testing models on different species, the *Hadamard* was still the most recurring as the one achieving better results.

The training and testing with different species allowed us to see that good predictions can still be achieved (especially for the One Aspect datasets) but there is a dependency on the species and on the number of PPIs present. The dependency on the number of PPIs was perhaps the biggest inconvenience since it greatly affected our results. The results of the training with two species followed the previously observed trend, i.e., in general, the models with a larger number of PPIs allowed for better predictions.

To further this study, it would be interesting to train and test the different models and embedding combination operations on datasets of homogenous size, as it would allow us to: 1) access if our conclusions regarding the models' performance still hold and 2) highlight the effect of species similarity.

ACKNOWLEDGMENTS

We thank Professor Cátia Pesquita for the support given to us across this semester and for her persistence in helping us despite our difficulties with the projects.
We also thank Rita Sousa, for all the help given to us and the availability to answer any of our questions.

REFERENCES

1. Cardoso, C., Sousa, R. T., Köhler, S. & Pesquita, C. A Collection of Benchmark Data Sets for Knowledge Graph-Based Similarity in the Biomedical Domain. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 12124 LNCS 50–55 (2020).
2. Chen, M. *et al.* Multifaceted protein-protein interaction prediction based on Siamese residual RCNN. *Bioinformatics* **35**, i305–i314 (2019).
3. Li, L. P., Wang, Y. Bin, You, Z. H., Li, Y. & An, J. Y. PCLPred: A bioinformatics method for predicting protein–protein interactions by combining relevance vector machine model with low-rank matrix approximation. *Int. J. Mol. Sci.* **19**, (2018).
4. Yang, F., Fan, K., Song, D. & Lin, H. Graph-based prediction of Protein-protein interactions with attributed signed graph embedding. *BMC Bioinformatics* **21**, 1–16 (2020).
5. Yang, L., Han, Y., Zhang, H., Li, W. & Dai, Y. Prediction of Protein-Protein Interactions with Local Weight-Sharing Mechanism in Deep Learning. (2020) doi:10.1155/2020/5072520.
6. Koehrsen, W. Neural Network Embeddings Explained | by Will Koehrsen | Towards Data Science. https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526 (2018).
7. Reese, J. T. *et al.* KG-COVID-19: A Framework to Produce Customized Knowledge Graphs for COVID-19 Response. *Patterns* **2**, 100155 (2021).
8. Kamakoti, B. Introduction to Knowledge Graph Embeddings | by Balaji Kamakoti | Towards Data Science. https://towardsdatascience.com/introduction-to-knowledge-graph-embedding-with-dgl-ke-77ace6fb60ef (2020).
9. Xiao, Z. & Deng, Y. Graph embedding-based novel protein interaction prediction via higher-order graph convolutional network. *PLoS One* **15**, (2020).
10. 7 Types of Classification Algorithms - Analytics India Magazine. https://analyticsindiamag.com/7-types-classification-algorithms/.
11. Gaussian Naive Bayes. https://iq.opengenus.org/gaussian-naive-bayes/.
12. Advantages and Disadvantages of different Classification Models - GeeksforGeeks. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-different-classification-models/.

APPENDIX:

Table S 1. Results of predicting PPIs with a model trained with different species (of the All Aspect datasets). For each pair, the accuracy of the prediction is present along the best method of combination. Random Forest algorithm was used for all predictions.

Train Species	Test Species	Operation	Accuracy
D. melanogaster	E. coli	Hadamard	0.8024
	H. sapiens	Hadamard	0.6178
	S. cerevisiae	Hadamard	0.7256
E. coli	D. melanogaster	Hadamard	0.78
	H. sapiens	Hadamard	0.6745
	S. cerevisiae	Hadamard	0.741
H. sapiens	D. melanogaster	Max	0.77
	E. coli	Min	0.6333
	S. cerevisiae	Hadamard	0.735
S. cerevisiae	D. melanogaster	Max	0.765
	E. coli	Min	0.6595
	H. sapiens	Hadamard	0.7024