Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 8 February 2019
Seminar training paper

Subject Section

Prediction of protein-protein interactions using support vectors

Stefan Dvoretskii [1,*]

[1] Chair of Bioinformatics, Department of Informatics, Technical University of Munich, 85478, Germany

[*] To whom correspondence should be addressed.

Associate Editor: Jonas Reeb

Received on 4 Jan 2019; revised on XXXXX; accepted on XXXXX

Abstract

In the world of intensive development of personalised medicine and targeted drugs' development it is becoming increasingly important to have information about various aspects about protein activity in an organism, one of such being protein-protein interaction - in which proteins engage constantly. The other important motivation behind studying protein-protein interactions, especially *in silico*, is the ongoing extension of our knowledge of the interactome of the various organisms and hence the increase of complexity of finding and organising the links between and inside the protein-protein interaction networks. This work tries to approach the problem of predicting interactions between proteins from the very start - by training a classifier (support vector machine) that defines whether a protein is *interactive*, i.e. whether it engages in an interaction with at least some other protein (that is, interacts with one protein or more). The training involved using a dataset of 905 unique proteins with known interaction class (i.e. it was known each of these proteins whether it interacts or does not interact with other proteins) represented with their FASTA sequence. There was no additional information apart from FASTA sequence and interactivity class, so the trained model is *sequence-based*. Used conversions include ProtVec representation and SMOTE oversampling, which are closer discussed in Methods section.

Results. Final model achieves the overall accuracy of 63 ± 5%, being barely distinguishable from the picked baseline model - achieved by random shuffling - with the overall accuracy of 56 ± 5%. The model achieves an impressive average precision of 84%.

Contact: stefan.dvoretskii@tum.de

1 Introduction

During the existence of proteins, for many interaction with other proteins plays an important part (O. et al., 2016). The interactions vary in their nature, and can be divided in physical and functional - former for interactions happening between two proteins in a physical proximity, latter for proteins taking part in a *same pathway* (i.e., tightly interconnected group of biochemical reactions) at different steps/in different roles (Wu et al., 2010). Further example of grouping proteins into distinct clusters by their characteristics is the obligation of the interaction between two proteins - can they exist outside the interaction with each other? If yes (and this group of protein interactions is called "non-obligate"), division according to the duration of interaction between proteins is possible - i.e. groups could be "permanent" (reacting forever after they "meet") and transient (temporary interaction based on the setting). As the reader may guess there are some differences between each of these groups; we are however just showing that protein-protein interactions can vary in many ways in their setting, durability, nature and in many other features as well. The areas of biology, for which interaction between proteins can be especially crucial, include interaction between proteins in signaling pathways(Pawson and Nash, 2000) and so-called protein aggregation diseases, among which are Alzheimer and Kuru disease (Aguzzi and O'connor, 2010).
While various experimental methods exist for discovering and/or measuring protein-protein interactions, such as dual expression recombinase based single vector system (JP et al., 2008) or phage display (GP and VA, 1997), there are also many *in silico* methods based on various approaches: e.g. genomic neighbourhood, interaction inheritance or even text mining (O. et al., 2016).

Classic, widely used tools for assessing the protein-protein interaction *in silico* that are open for the public, include STRING database (Jensen et al., 2008) and Struct2Net MIT webserver (Singh et al., 2010). For the approaches based on computational algortihms, there are of course differences in the amount and, most importantly, type of data (so-called "features" in machine learning paradigm), that are used to assess the prediction between the proteins. Some of the recent efforts to build a computational algorithm for predicting the protein-protein interactions include Z.H. et al., 2014, Emamjomeh et al., 2014 or Göktepe and Kodaz, 2018, who have even done it based only sequence information. It is however to notice that while all the listed tools are made to predict interactions between two proteins (and networks of protein interactions (Barabasi and Oltvai, 2004), in general), our work was focused around predicting the binary interativity class of a protein. Having also mentioned the possibility of taking various information about the (potentially) interacting proteins in account, as an input into our model, it is to say that we have made predictions about whether a protein interacts only based on its *sequence* - "interacts" here meaning whether a given protein interacts with at least some other proteins.

2 Materials and Methods

2.1 Dataset

We used a dataset of 1187 protein sequences with known binding class (binds/doesn't bind). Sequences were supplied in FASTA format. After filtering for full duplicates, the dataset comprised of 905 unique FASTA sequences with a known binding class. It is to say that the original dataset was *imbalanced* with appx. 2:1 bind to non-bind class ratio.

2.2 Data preparation

The sequences were converted to ProtVec representation (Asgari and Mofrad, 2015) of length 100, summarizing 3-gram vectors of the same length along sequence with step 1. Figure 1 shows the schema of converting FASTA sequences to ProtVec. The pre-trained dictionary of 3-gram ProtVec representations was also taken from the supplement data of Asgari and Mofrad, 2015.

The dataset was then splitted in test dataset, comprising one fourth of the original dataset (227 sequences with known binding class) and training dataset (the rest - i.e., 678 sequences).

2.3 Creating oversampling pipeline

To avoid class bias in predictions, we used SMOTE oversampling algorithm to create new observations of the underrepresented class, using an "average" of 5 adjacent observations of the same class (Chawla et al., 2002). We used `imblearn.over_sampling.SMOTE` implementation of the algorithm.

2.4 Optimizing the model

As a model, Support Vector Machine was utilized, based on Support Vector methods(Cortes and Vapnik, 1995). For training the model, Python's `sklearn` (Pedregosa et al., 2011) module was used. For the actual SVM, `sklearn.svm.SVC` class was used. To achieve the best parameters for the prediction task, we analysed several metrics using sklearn.

2.4.1 Picking C and kernel

First of all, we compared various combinations of kernel and regularisation parameter C using *sklearn's* `GridSearchCV` functionality with three cross-validation folds. Figure 2 shows the results of the grid search.

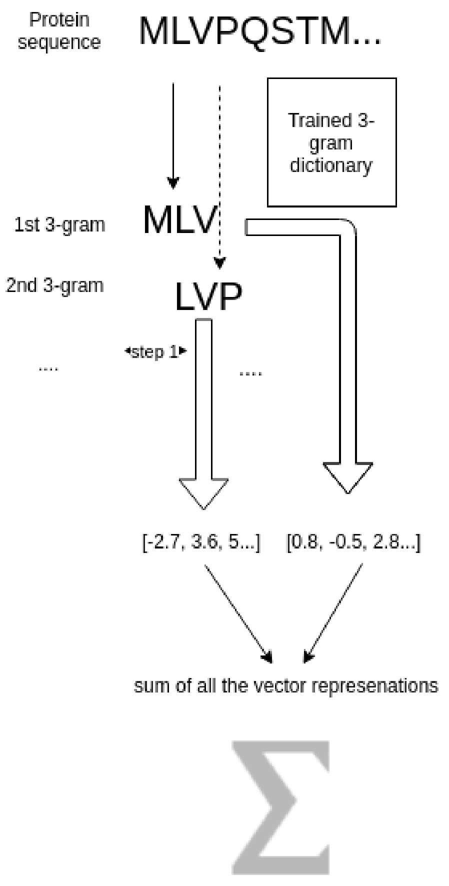

Fig. 1. The algorithm of converting FASTA protein sequences to their ProtVec representation. A sliding window of 3 positions slides along the sequence with step of 1 position. Then, the ProtVec representations of resulting 3-grams are being looked up in a trained dictionary. After that, the ProtVec representation of a sequence is achieved by summing up all the partial 3-gram ProtVec representations.

The method we used to pick *best measurement* was to *maximize train* score and, at the same time, *minimize the difference* between train and test scores to avoid overfitting.
In case of current pipeline, we picked RBF kernel with C=0.3.

2.4.2 Selecting gamma

In the next step, we have assessed the `gamma` parameter of the SVM; for that, `sklearn.model_selection.validation_curve()` function was used. Figure 3 shows the validation curve of various values of the `gamma` parameter

2.5 Running the model

Using the best parameters, we fitted the training dataset into a `Pipeline()` instance and then ran the prediction on the test dataset using `SVC().predict()`.

2.6 Evaluation

2.6.1 Standard error

The standard error was calculated as the standard deviation of the distribution of performance measurements of 1000 bootstrap resamplings

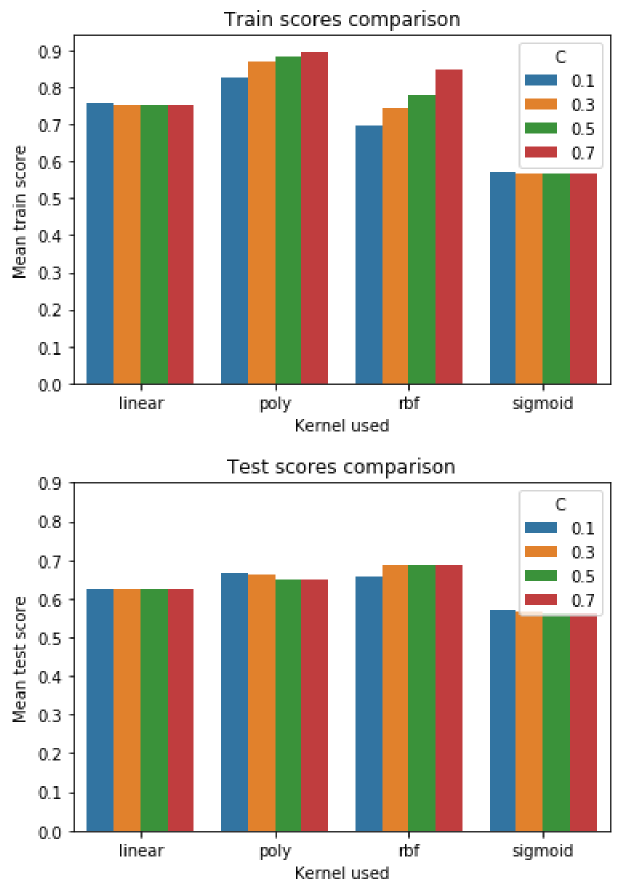

Fig. 2. Comparison of various classifier parameters combinations with regard to their scores. Each bar corresponds to a kernel, with color pointing on the C value used. Heights of the bars stand for their training or test score - accuracy - in accordance with the graphs ordering.

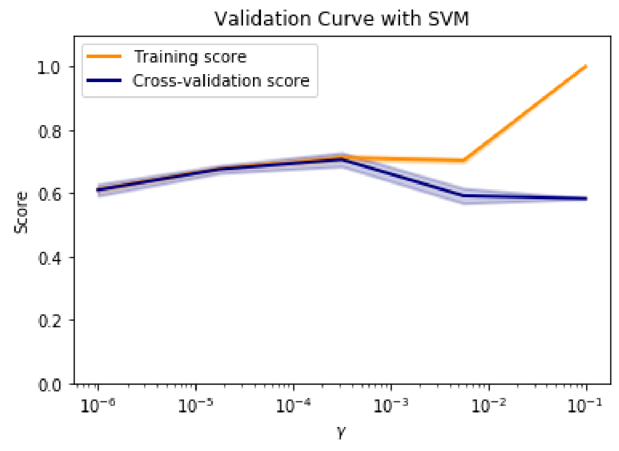

Fig. 3. Validation curve of the gamma parameter. The value of gamma inreasing the test score, but minimizing the difference between test score and train score was picked (in this case $\gamma = 5 * 10^{-3}$)

of the *test* dataset, i.e.

$$\mathrm{SE} = \mathrm{SD}(resamplings) \tag{1}$$

building on the assumption, that the vector of performance measurements follows a Gaussian distribution.

2.6.2 Confusion matrix and performance measurement

To have the first overview of the model's performance, we first constructed a confusion matrix of two rows and two columns, according to the binary classification origin of the prediction task. The matrix was constructed after fitting the predictor `SVC()` with the best parameters (see method) on the training dataset and during the prediction of the test partition. As a measurement of model's performance, *overall accuracy* (defined as fraction of correctly predicted samples) was used (calling `sklearn.metrics.accuracy_score()`).

2.6.3 Comparison to other models

To further assess the performance of the model, it was compared to random baseline. The random baseline was constructed by shuffling the *target* column of `test` dataset with random seed and then taking the performance measurement. We haven't compared the model to other state-of-art model because of the lack thereof.

2.6.4 Precision-recall

To compute the average precision-recall score, `sklearn.metrics.average_precision_score()` was used. To draw the precision recall curve, we have used `sklearn.metrics.precision_recall_curve()`. As a statistic of the curve we used *average precision* (or AP), defined as

$$\text{AP} = \sum_n \Delta R_n \dot{P}_n \tag{2}$$

The exact form of the curve as well as average precision value are discussed in Results section.

3 Results and Discussion

3.1 Standard error

Table 1 shows standard errors to expect when working with performance measurements used when evaluating the model's performance, measured in percent.

Table 1. Standard errors of performance measurements.

Performance measurement	Standard error(%)
Overall accuracy	5,27
F1 score	5.70
Overall accuracy (baseline)	5.72

Baseline model (random shuffling)'s errors are given with a corresponding note

We notice that the computed standard errors are relatively high, stably overcoming a 5 per cent mark, which for instance increases the range of confidence interval and generally suggests a possibility of improvement in the direction of predictions' stability.

3.2 Accuracy and F1 score

Model achieved overall accuracy of $63\pm5\%$. The random baseline model (constructed by shuffling the *test* dataset's `target` vector) achieved $56\pm5\%$ overall accuracy. Told that, we notice that there is a certain probability of performing more accurate predictions using the proposed model rather than the baseline model.
Next, we have calculated the F1 score of the model, which lied around $68\pm6\%$. Table 2 gives an overview of the scores assessed during model's evaluation.

Table 2. Performance measurements overview

Performance measurement	Value
Overall accuracy	$63\pm5\%$
F1 score	$68\pm6\%$
Overall accuracy (random baseline)	$56\pm5\%$

Summary of performance measurements of the model and its random baseline.

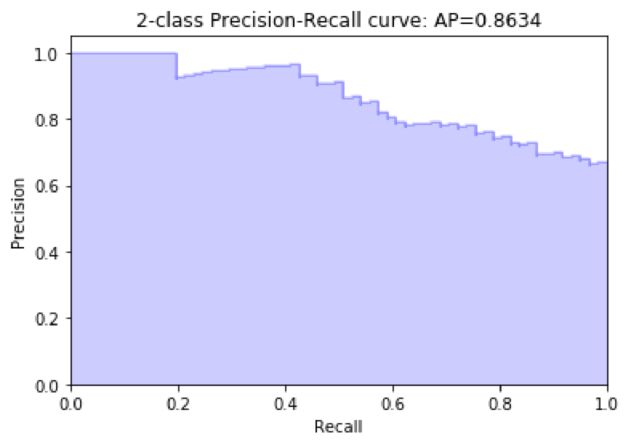

Fig. 4. Second class precision-recall curve of the model. As a summarizing metric of the curve, Average Precision was used, which is marked with `AP` at the top of the graph (see corresponding section in Results and Discussion section for the exact definition of average precision)

Because of the lack of any models pursuing the same task (assess the binary interactability class of a protein) no comparisons to state-of-art models from another authors was made.
Given the accuracy measurements, we notice that it is in the range (especially taking in account the standard error) of a class bias on the test dataset. That could suggest that in some cases, the model achieves the same accuracy of predictions as a classifier with just one of the two classes (labeling all observations with just one class) would do. That doesn't say anything good in the way to positively distinguish our model from the random baseline, however it is also nothing negative that would neglect our proposed model in that this fact pertains.

3.3 Precision and recall

The final metrics using which we have undertaken an assessment of the resulting model's performance were precision and recall. For that, we have drawn a second-class precision-recall curve, shown in Figure 4.

As a summary metric of the curve, we have used Average Precision, defined as following

$$\sum AP = \Sigma_n (P_n - P_{n-1}) R_n \tag{3}$$

where P_n and R_n are precision and recall at the n-th threshold correspondingly, with $P_n - P_{n-1}$ representing the ΔP, i.e. change of precision since the last threshold.
Told this, it is to notice that our model achieves a gratifying precision-recall measurement (comparison to other state-of-the-art methods is provided e.g. by Göktepe and Kodaz, 2018). The curve as well as the area under it seem to be especially convincing, suggesting that we can use the model with right cutoff for recall as well as specificity purposes.

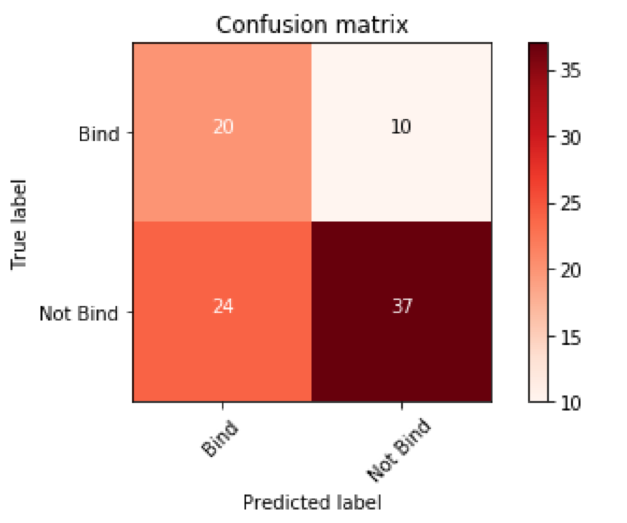

Fig. 5. Confusion matrix of the `SVC().predict()` run on the test dataset.

3.4 Confusion matrix

Figure 5 shows the resulting confusion matrix.

We notice that it corresponds to and therefore supports the results of other performance assessment methods, especially precision-recall curve.

4 Conclusion

We have constructed a model based on support vector approach that can distinguish between interactive and non-interactive "standalone" proteins with more or less good performance.

We notice that especially the precision-recall curve with average precision of 86 per cent could be an argument of usability of the model in some settings. One must also say that this is a novel type of model in the field in relation to the goal of it (as already mentioned in Introduction, most of the models focus on more exact goals like predicting interaction between two proteins or the interface(s) of interaction).

There is no clear way to distinguish the performance of this model from random baseline (see Results); however we suggest that this model still has a place to be in it's appropriate scientific field, at least because it presents a solution to a novel prediction task in the field of computational approach to the prediction of protein-protein interactions.

Acknowledgements

Author is thankful for the supervision of Jonas Reeb. He also thanks the organizational commitee of the Bachelor Bioinformatics course between Technical University of Munich and Ludwig-Maximilians-University Munich for creating the Problem Based Learning module, as well as the Rostlab team for organizing one of the subcourses including lectures and supply materials from which the author has learned about the topics involved in working on the project. Furthermore, the author thanks the course of Problem Based Learning at the Rostlab in summer term 2018/winter term 2018-2019 for their presentations delivering some important insights in Machine Learining, Statistics and Scientific Working.

Funding

This work has been supported by Good Will, Motivation to Succeed and Wish to Know. Other sources of support may exist, however without scientific evidence. It is nevertheless clear that no financial funding was provided.

References

Aguzzi, A. and O'connor, T. (2010). Protein aggregation diseases: pathogenicity and therapeutic perspectives. *Nat Rev Drug Discov.*, 9(3):237–48.

Asgari, E. and Mofrad, M. R. K. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE*, 10(11):1–15.

Barabasi, A.-L. and Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5:101.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Emamjomeh, A., Goliaei, B., and A. Torkamani, R. Ebrahimpour, N. M. A. P. (2014). Protein-protein interaction prediction by combined analysis of genomic and conservation information. *Genes Genet. Syst.*, 89 (6):259–272.

Göktepe, Y. E. and Kodaz, H. (2018). Prediction of protein-protein interactions using an effective sequence based combined method. *Neurocomputing*, 303:68 – 74.

GP, S. and VA, P. (1997). Phage display. *Chem. Rev.*, 97 (2):391âŁ"410.

Jensen, L. J., Kuhn, M., Stark, M., Chaffron, S., Creevey, C., Muller, J., ..., and Bork, P. (2008). String 8 - a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(suppl. 1):D412–D416.

JP, L., LK, B., and JH, P. (2008). Dual expression recombinase based (derb) single vector system for high throughput screening and verification of protein interactions in living cells. *Nature Precedings*.

O., K., N., T., and A., G. (2016). Predicting proteinâ˘'protein interactions from the molecular to the proteome level. *Chem. Rev.*, 116:4884â˘'4909.

Pawson, T. and Nash, P. (2000). *Genes Dev.*, 14(9):1027–47.

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

Singh, R., Park, D., Xu, J., Hosur, R., and Berger, B. (2010). Struct2net: a web service to predict protein-protein interactions using a structure-based approach. *Nucleic acids research*, 38(Web Server issue):W508–W515.

Wu, G., Feng, X., and L., S. (2010). A human functional protein interaction network and its application to cancer data analysis. *Genome Biol.*, 11(5):R53.

Z.H., Y., S., L., X., G., X., L., and Z., J. (2014). Large-scale protein-protein interactions detection by integrating big biosensing data with computational model. *Bio. Med. Res. Int.*