Journal of Bioinformatics and Computational Biology
Vol. 16, No. 4 (2018) 1850014 (18 pages)

DOI: 10.1142/S0219720018500142

World Scientific
www.worldscientific.com

Training host-pathogen protein–protein interaction predictors

Abdul Hannan Basit[*,†,‡], Wajid Arshad Abbasi[*,§], Amina Asif[*,¶], Sadaf Gull[*,‖] and Fayyaz Ul Amir Afsar Minhas[*,**]

[*]*Department of Computer and Information Sciences*
Biomedical Informatics Research Laboratory
Pakistan Institute of Engineering and Applied Sciences (PIEAS)
Nilore, Islamabad 44000, Pakistan

[†]*Department of Electrical Engineering*
Pakistan Institute of Engineering and Applied Sciences (PIEAS)
Nilore, Islamabad 44000, Pakistan
[‡]*hannan.ahb@gmail.com*
[§]*wajidarshad@gmail.com*
[¶]*a.asif.shah01@gmail.com*
[‖]*sadafzakarkhan@gmail.com*
[**]*afsar@pieas.edu.pk*

Received 8 September 2017
Revised 26 March 2018
Accepted 8 May 2018
Published 31 July 2018

Detection of protein–protein interactions (PPIs) plays a vital role in molecular biology. Particularly, pathogenic infections are caused by interactions of host and pathogen proteins. It is important to identify host–pathogen interactions (HPIs) to discover new drugs to counter infectious diseases. Conventional wet lab PPI detection techniques have limitations in terms of cost and large-scale application. Hence, computational approaches are developed to predict PPIs. This study aims to develop machine learning models to predict inter-species PPIs with a special interest in HPIs. Specifically, we focus on seeking answers to three questions that arise while developing an HPI predictor: (1) How should negative training examples be selected? (2) Does assigning sample weights to individual negative examples based on their similarity to positive examples improve generalization performance? and, (3) What should be the size of negative samples as compared to the positive samples during training and evaluation? We compare two available methods for negative sampling: random versus *DeNovo* sampling and our experiments show that *DeNovo* sampling offers better accuracy. However, our experiments also show that generalization performance can be improved further by using a *soft DeNovo* approach that assigns sample weights to negative examples inversely proportional to their similarity to known positive examples during training. Based on our findings, we have also developed an HPI predictor called HOPITOR (Host-Pathogen Interaction Predictor) that can predict interactions

[**] Corresponding author.

between human and viral proteins. The HOPITOR web server can be accessed at the URL: http://faculty.pieas.edu.pk/fayyaz/software.html#HoPItor.

Keywords: Host–pathogen interactions; interaction predictor; protein–protein interactions; negative sampling.

1. Introduction

Proteins are complex molecules that take part in virtually all life processes in living organisms[1] such as metabolism, signaling, and structural organization.[2] Protein sequences are composed of long chains of 20 amino acids.[1] The sequence of a protein determines its three-dimensional structure, and, consequently, its specific functions.[1,3] Proteins rarely act alone: more than 80% of proteins operate in complexes formed through interactions of proteins.[1,2,4,5] Therefore, to understand functional mechanisms of proteins, it is very important to study the protein–protein interactions (PPIs).[6] A special type of PPIs is host–pathogen interactions (HPIs) that involve interactions between proteins from a pathogen (virus or bacteria) and its host.[7] According to the World Health Organization, each year more than 17 million people are killed by infectious diseases.[8,9] To fight these infectious diseases, it is important to identify HPIs as it is a key step in drug design and biological discovery of disease mechanisms.[6]

Conventional wet lab techniques are expensive and time-consuming, making it almost impossible to assess all possible combinations of protein interactions between a pathogen and its host.[10,11] Therefore, computational approaches are used to predict HPIs.[10,11] For example, if we want to find possible interactions of only 2000 host proteins with 500 pathogen proteins, the possible host-pathogen combinations turn out to be one million. For this reason, there is a shortage of experimentally verified HPI data. Computational studies are vital to increase the available PPI data and eventually add to the pace of drug design research.[11]

Most available computational methods predict HPIs that use sequence and structural similarity-based techniques.[7,11–14] However, due to limited availability of structural information[15] and missing data,[16] sequence-only based methods are better for generalized predictors. In machine learning techniques for PPIs, feature vectors are extracted from protein sequences and then a machine learning model learns from available data to predict unknown interactions.[11–13,17,18] A binary classifier uses a training data set of known interacting and noninteracting protein pairs.[19] The positive samples in training of the classifier are obtained from biochemical experiments, whereas, negative samples are typically generated computationally.[11,13,19,20] Machine learning models for predicting PPIs include Support vector machines (SVMs),[13,17,18] Random Forest (RF),[21] Gradient Boosting Machine (XGBoost),[22] Neural Networks, etc.

In this work, we discuss three questions related to the development of machine learning models for prediction of HPIs, and their performance evaluation as discussed

below. We also present a new prediction method called HOPITOR which performs significantly better than previous techniques.

The first question that arises in the development of a machine learning model for HPIs is how to generate negative samples for training a host-pathogen interaction predictor. Ben-Hur *et al.*[20] discuss the available methods to choose negative examples and conclude that negative samples should be selected uniformly at random. Although random sampling may contaminate the negative examples with interacting proteins, this contamination is likely to be minor. More recently, a method called *DeNovo* negative sampling has been proposed by Eid *et al.*[13] to replace random sampling. *DeNovo* sampling is a dissimilarity-based negative sampling criterion that considers sequence similarities of viral proteins interacting with a host protein in generating negative examples.[13] In *DeNovo* sampling, pairs of host and pathogen proteins whose sequences are similar to known positive examples are excluded from the set of negative examples. The experiments by Eid *et al.* show that *DeNovo* sampling is more effective than random sampling in training HPI predictors in terms of generalization performance. However, Eid *et al.* use a balanced dataset with an equal number of positive and negative proteins selected with *DeNovo* sampling which does not reflect the true use of an HPI prediction method as the number of pairs of host and pathogen proteins can be much larger than possible protein interactions. We use a simple SVM to compare these two methods for sampling negative examples on an imbalanced dataset.

The second question is: whether assigning sample-level weights to negative classification examples based on their similarity to positive examples can help improve generalization performance of HPI predictors or not? Training examples can be assigned sample level weights to reflect domain knowledge or confidence in their labeling or noise contamination. Eid *et al.*[13] and others[17–19] do not assign sample level weights to individual training examples. The idea of using weighted training samples is widely implemented in other fields of science[24–26] Ravikant *et al.*[27] used the idea of weighted samples in an energy-based docking method for PPIs. Inspired by the success of *DeNovo* sampling in selection of negative examples, we hypothesize that assigning sample weights for negative examples are inversely proportional to their similarity to known positive examples that can help in improving classifier generalization. This approach can be thought of as a *soft DeNovo* approach for handling negative examples in training HPI predictors. It generalized the concept of *DeNovo* sampling: negative examples that are similar to known positive examples have a lesser impact on the decision boundary of a classifier in comparison to more dissimilar ones to ensure good generalization. Using such sample level weighting can reduce the impact of false positives during training. We test this hypothesis by comparing the performance of different classifiers with and without sample weights.

The third question considered in this study is: what is the effect of training and test data sizes on prediction performance? Should we use balanced training and evaluation sets with an equal number of positive and negative examples or should we

use as much available data as possible? Eid *et al.*[13] and others[17,18] use a balanced set in their evaluation. However, as discussed above, this approach does not simulate real-world use of HPI predictors because the number of possible interactions between proteins is much smaller than all possible pairs of those proteins.[23] Therefore, we propose that the entire data should be used for training an HPI predictor and experimentally show that this approach offers improved prediction performance.

The rest of the paper is organized as follows: Section 2 describes the methods used in the study, Section 3 gives results and discussion about different computational experiments whereas Sec. 4 presents our conclusions.

2. Methods

An overview of the methodology is presented in the flowchart given in Fig. 1. The step-wise development is given below.

2.1. *Datasets and preprocessing*

HPIs are obtained from supplementary data provided by Eid *et al.*[13] This dataset has originally been compiled using VirusMentha.[28] After removing duplicates, it has 4971 unique interactions between 2237 human proteins and 337 viral proteins. The viral proteins are divided into 10 groups based on their biological families as shown in Table 1. The partitioned data was obtained from Eid *et al.*[13] on request.

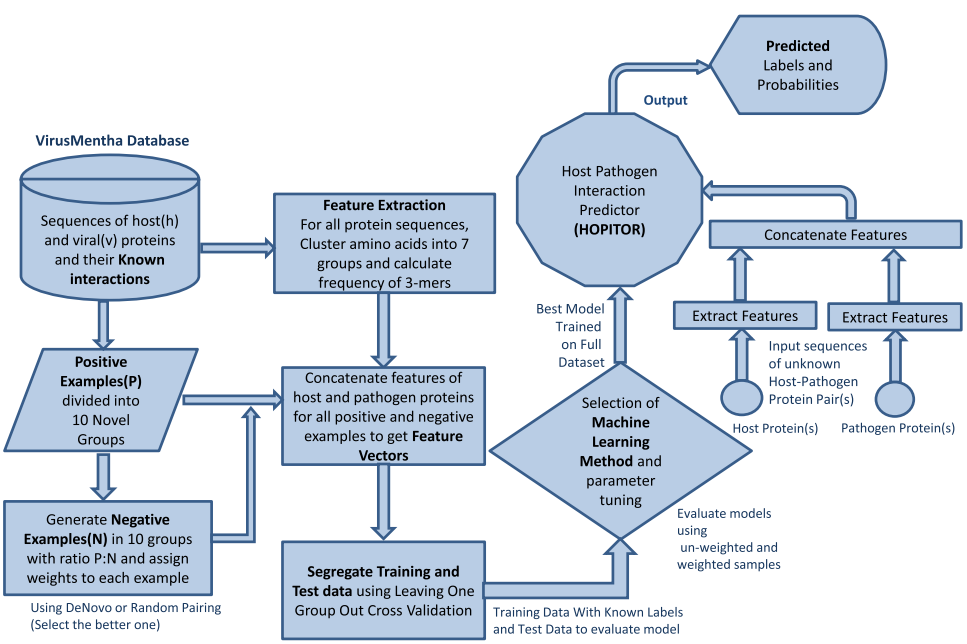

Fig. 1. Flowchart for developing HOPITOR.

Table 1. Partitioned viral families.

Group no.	Family name	No. of viral proteins	Positive examples	*DeNovo* negative examples	Ratio of negative to positive examples
1	Paramyxoviridae	12	762	1797	2.35
2	Filoviridae	4	114	592	5.20
3	Bunyaviridae	3	159	508	3.20
4	Flaviviridae	14	291	25953	89.2
5	Adenoviridae	22	88	3453	39.2
6	Orthomyxoviridae	32	664	5004	7.50
7	Chordopoxviridae	26	194	4158	21.4
8	Papillomaviridae	40	245	6665	27.2
9	Herpesviridae	134	1001	25505	25.5
10	Retroviridae	50	1399	8062	5.76
	Total	**337**	**4917**	**81697**	**16.6**

2.2. *Generating negative samples*

Machine learning-based PPI predictors require both positive and negative datasets for their training. Because the interaction data are available for positive class only, the generation of negative examples is the first step. We evaluate two different techniques for generating negative samples: random versus *DeNovo* sampling.

2.2.1. *Random negative sampling*

Ben-Hur *et al.*[20] argue that, even though random sampling may contaminate the dataset, this contamination is not likely to effect classifier performance much. This method is illustrated in Fig. 2(a). First, a random host protein and a random pathogen protein are chosen; then, it is checked whether the chosen pair is part of the set of known interactions; if not, then the randomly chosen samples are selected as negative examples. The solid connectors represent the positive HPIs and the dotted connectors show the randomly selected negative HPIs. Pathogen protein 1 can be paired with host proteins a and c as a negative example; however, it cannot be paired negatively with host protein b since there exists a positive sample between 1 and b.

2.2.2. *DeNovo negative sampling*

Eid *et al.*[13] hypothesize that viral proteins with high sequence similarity can interact with similar host proteins. Based on this hypothesis, the authors argue that random negative sampling will result in a large number of false negatives samples. To mitigate this issue, Eid *et al.*[13] propose dissimilarity-based negative sampling criterion called *DeNovo* negative sampling.

In *DeNovo* sampling, pairwise sequence similarity of viral proteins is first determined. If two viral proteins are more similar than a cut-off value, a host protein that interacts with one of them cannot be paired with the second one to form a negative example. Dissimilarity scores are used to compare the similarities between viral proteins. These scores are obtained by taking the complement of normalized bit

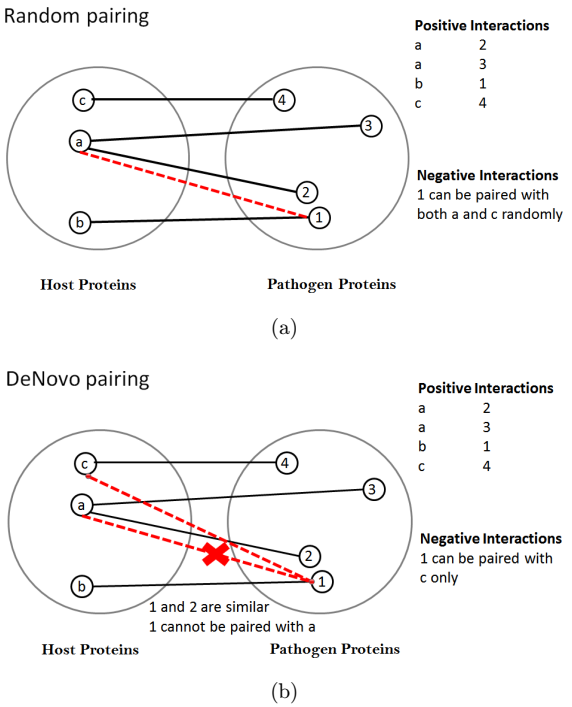

Fig. 2. Illustration of (a) Random sampling and (b) *DeNovo* sampling.

scores from the all-versus-all pairwise global alignment of viral proteins. At a dissimilarity threshold T, the negative samples that do not fulfill the criterion are filtered out, and random sampling is done over the rest of the negative examples. We use $T = 0.7$ in this study. This method is illustrated in Fig. 2(b). The complete details of *DeNovo* pairing technique is given in Eid *et al.*[13] In Fig. 2(b), pathogen protein 1 cannot be paired with host protein b because it interacts with it. Moreover, it also cannot be paired with host protein a as 1 and 2 are similar to each other, i.e. their dissimilarity distance is less than T, and 2 interacts with a. However, viral proteins 1 and 4 have dissimilarity distance $\geq T$, and they have a positive example with human proteins b and c respectively, therefore, 1 can only be paired with c as a negative example. Table 1 shows the number of *DeNovo* negative examples for each viral family in our evaluation. For a fair comparison, an equal number of negative proteins were selected based on random sampling as well.

2.3. *Feature extraction*

In this work, we have used clustered tripeptide composition features which were also used by Eid *et al.*[13] and originally proposed by Shen *et al.*[17] and others.[12,18] In this approach, the 20 amino acids are first clustered into seven groups based on their physiochemical properties that affect protein interactions such as side chain volume

and dipoles.[17] The seven clusters are $\{A, V, G\}$, $\{I, L, F, P\}$, $\{Y, M, T, S\}$, $\{H, N, Q, W\}$, $\{R, K\}$, $\{D, E\}$ and $\{C\}$. After clustering, the frequency of all possible 3-mers is calculated in each protein sequence to get a $7^3 = 343$ dimensional protein-level feature vector. The individual feature vectors in an example (h, v) comprising of a host protein h and pathogen protein v are first normalized to unit norm and concatenated into a single 686-dimensional feature vector.

2.4. *Classification models*

We evaluate three different regularized machine learning models in our study: SVM, Random Forest (RF) and Gradient Boosting Machine (XGBoost).[22] We use Scikit-learn 0.19[29] in Python 2.7[30] to train and evaluate SVM and RF models and python-based xgboost 0.7 API for training and testing of XGBoost. For notation, we assume that we are given sets of positive and negative examples, P and N, respectively, with a total of $n = |P| + |N|$ examples. An example is represented by its feature vector $\mathbf{x_i}$ and associated label $y_i \in \{+1, -1\}$ for $i = 1, \ldots, n$. Below we discuss classifiers used in this study.

2.4.1. *Support vector machines (SVMs)*

We have used SVMs[31] with a radial basis function kernel. SVM is a large-margin classifier with the decision function $f(\boldsymbol{x}) = \mathbf{w}^T \boldsymbol{x} + b$ where $\mathbf{w}$ and b are weight and bias terms of the classifier. The optimization problem of an SVM tries to minimize the number of margin violations and misclassifications over training data while maximizing its regularization or margin:

$$\min_{\mathbf{w},\mathbf{b},\xi} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{n} c_i \xi_i, \tag{1}$$

$$\text{such that for all } i = 1, \ldots, n : y_i(\mathbf{w}^{\boldsymbol{T}} \mathbf{x_i} + \mathbf{b}) \geq 1 - \xi_i, \quad \xi_i \geq 0. \tag{2}$$

Here, ξ_i is the extent of margin error for the ith example. The hyper-parameter c_i is the margin violation penalty of the ith example and it determines the relative significance of its margin violation and maximization of the margin. Typically, the margin violation penalties of all examples are set equal to C to reduce the number of hyper-parameters. However, these margin violation penalties can be used to assign sample weights to individual examples as well. The above equation can be expressed as the dual form:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x_i^T} \mathbf{x_j}, \tag{3}$$

$$\text{such that : } \sum_{i=1}^{n} y_i \alpha_i = 0, \quad \text{and}, \quad 0 \leq \alpha_i \leq c_i. \tag{4}$$

In (3), the dot product term $\mathbf{x}_i^T\mathbf{x}_j$ can be replaced by a generalized dot product or kernel function, $K(\mathbf{x}_i, \mathbf{x}_j)$ to make the classifier nonlinear. In this study, radial basis function is used which is given as: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_j - \mathbf{x}_j\|^2)$. We optimize the SVM kernel parameters using grid search and select $C = 10, \gamma = 0.1$.

2.4.2. *Random forest (RF)*

Random Forest (RF) is an ensemble learning technique that works by creating several decision trees on arbitrary subsample sets of input features during training. In testing, it produces a probability value for a test example to belong to the positive class by taking the mode of classes of individual trees.[21] We optimize the hyper-parameters of the RF ensemble using grid search with respect to maximum number, depth and splitting features of decision trees.

2.4.3. *Gradient boosting machine (XGBoost)*

XGBoost is an ensemble learning technique that operates by combining weak decision tree learners in an iterative manner through boosting. This technique uses gradient boosting to learn a decision function that minimizes the average value of classification loss on training data based on a combination of decision trees trained in an incremental manner over residual errors of the previous stage.[32]

We use the grid search to optimize XGBoost parameters with respect to maximum depth, objective function, learning rate, booster subsample and number of boosting iterations. The optimal parameters are: learning rate = 0.1, max depth = 10, no. of estimators = 100 and subsample = 1.

2.5. *Soft DeNovo: Weighting of negative examples in training*

In training machine learning models for classification, training examples can be assigned a sample weight based on confidence about the correctness of their labels, the degree of noise in the sample or class sizes. Sample weighting can be used to include domain-specific knowledge about examples in training a machine learning model. For example, in a support vector machine, each example can be assigned its own margin violation penalty c_i instead of having a global C hyper-parameter as discussed earlier. One advantage that this flexibility allows is to handle class imbalance by assigning a larger margin violation penalty to the minority class examples in comparison to examples from the majority class. Similar sample weighting schemes are also available for both random forest and XGBoost classifiers as well.[21,32]

In this work, we propose to use sample weights to reflect our confidence in the correctness of labels of negative training examples based on their sequence similarity to positively labeled training examples. To test our hypothesis that sample weighting based on this approach will improve classification performance, we compare various classifiers with and without sample weighting as discussed below.

2.5.1. *Unweighted training samples*

In this approach,[31] all examples are weighted equally during training. However, to handle class imbalance, all examples in a class are assigned the same weight which is inversely proportional its prior class probability.

2.5.2. *Soft DeNovo*

In this approach,[24] a separate value of sample weight is assigned to each training example. As positive training examples are experimentally validated, therefore, all positive examples are assigned a sample weight of 1.0. On the other hand, negative training examples are computationally generated, therefore, negative examples are assigned weights between 0 and 1 depending upon their sequence similar to known positive examples. Specifically, we set the sample weight of a negative example comprising of a host protein h and viral protein v to the dissimilarity score of v with its most similar viral protein that interacts with the host protein h. This idea is illustrated in Fig. 3. Mathematically, the sample weight of a negative example consisting of a host protein h and viral protein v can be written as: $\min_{v' \in \{v''|(h,v'') \in P\}} D_{vv'}$ where $D_{vv'}$ is the normalized dissimilarity score between viral proteins v and v'. This method of assigning sample weights can be thought of as a *soft DeNovo* approach for handling negative examples. It generalizes the *DeNovo* sampling concept proposed by Eid *et al.* negative examples that have high sequence similarity with positive examples should have a smaller effect on determining the classification boundary in comparison to more dissimilar ones to ensure good generalization by reducing the impact of potential false positives. This allows the classifier to construct a decision boundary through an area of low data density. Assigning weights to negative samples in this manner can also reduce the impact of the method used for selecting negative examples on generalization performance of an HPI predictor.

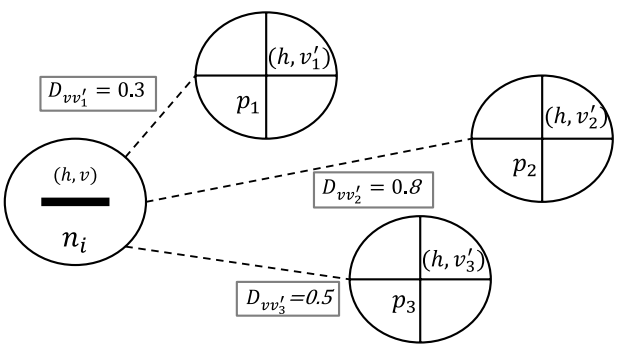

Fig. 3. An example of selecting the weight, c_i of a negative sample. Consider the pair $(\boldsymbol{h}, \boldsymbol{v})$ between host protein $\boldsymbol{h}$ and viral protein $\boldsymbol{v}$ such that the host protein $\boldsymbol{h}$ interacts with $\boldsymbol{v_1'}, \boldsymbol{v_2'}$, and $\boldsymbol{v_3'}$. Similarity of $\boldsymbol{v}$ is maximum with $\boldsymbol{v_1'}$. Therefore, dissimilarity distance $\boldsymbol{D_{vv_1'} = 0.3}$ is set as the sample weight c_i.

To handle class imbalance, the sample weight of an example is then multiplied by its class-level weight which is inversely proportional to its prior class probability.

2.6. *Performance evaluation*

Machine learning models in this study have been evaluated using leave-one-group-out cross-validation as implemented by Eid *et al.*[13] In this evaluation, a machine learning model is trained on examples from all but one of the 10 groups listed in Table 1 and tested on the held-out group or family. This gives a more realistic assessment of generalization performance in scenarios where the predictor will be used for identifying protein interactions of a novel viral family.

The cross-validation performance of different machine learning models is evaluated using area under the Receiver Operating Characteristic curve (AUC-ROC) and precision-recall curves (AUC-PR).[11,33] For this purpose, True positive rate (TPR) (or recall), False Positive Rate (FPR) and precision are defined as follows: $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$, $\text{FPR} = \frac{\text{FP}}{\text{TP+FN}}$, $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$.

AUC-ROC is the area under the plot between TPR and FPR at various thresholds. The ROC curve tells us how good a predictor is at detecting true positives at a given rate of false positives. The ROC curve is not sensitive to class imbalance. To report our results on imbalanced datasets, we have used the area under the precision-recall curve (AUC-PR). The weighted averages of AUC-ROC and AUC-PR for all groups with respect to the number of examples in them are reported for comparison.

2.7. *Biological validation*

To test the generalization performance of our prediction model, we test it on three different viral species (Human respiratory syncytial virus (HRSV), Measles virus, and Rabies virus) from three different families. Specifically, we use our trained model to identify the interactions of all proteins in a viral proteome with human STAT1 and STAT2 proteins. It is important to note that these examples are not part of our training data and one of the viral families is completely novel for the classifier. We rank all putative interactions of the viral proteins with their human host proteins and compare them to the literature as discussed in the results section.

3. Results and Discussion

3.1. *Negative sampling*

In order to select the optimal method for generating negative examples, we set-up two SVM models to test Random and *DeNovo* negative sampling. The first model is trained with negative examples generated by using Random negative sampling whereas the second model is trained with negative examples generated by using *DeNovo* negative sampling at $T = 0.7$. Eid *et al.*[13] use the same number of positive

and negative examples. However, to simulate the real word scenario where the number of positive examples is expected to be much smaller than negative examples, we select the entire set of *DeNovo* negative examples instead of using the balanced set. Both classifiers are tested on positive and *DeNovo* negative examples using leave one group out cross-validation. The weighted average AUC-ROC for the model trained with random negative sampling is 0.47 whereas, the AUC-ROC for the model trained with *DeNovo* negative sampling is 0.68. Moreover, the average AUC-PR for random sampling is 0.05, while it is 0.39 for *DeNovo* sampling.

The detailed group-wise results are shown in Table 2. These significantly improved results show that *DeNovo* sampling is indeed better than Random sampling. Therefore, we choose *DeNovo* negative sampling for building our HPI predictor and comparing classifiers as discussed in the next section. This result is in agreement with the findings of Eid *et al.* However, unlike the work by Eit *et al.*, we have used an imbalanced dataset with precision-recall scores. As a consequence, our result further generalizes the conclusion that *DeNovo* sampling should be used in training HPI predictors.

3.2. *Classifier comparison*

We compare the performance of three different classifiers (SVM, RF, and XGBoost). The cross-validation results of these three classifiers are tabulated in Table 3. It can be seen that XGBoost outperforms both SVM and RF both in terms of AUC-ROC and AUC-PR. As a consequence, we use XGBoost for our final model.

3.3. *Effect of sample weighting*

In order to analyze the effect of sample weighting, we compare the three classification techniques (SVM, RF, and XGBoost) with and without sample weighting. The results are shown in Table 4 and are discussed below. It is interesting to note that

Table 2. Comparison of random versus *DeNovo* sampling for SVM classifier.

	Random sampling		*DeNovo* sampling	
Group	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
1	0.43	0.244	0.96	0.95
2	0.272	0.101	0.984	0.973
3	0.38	0.203	0.952	0.943
4	0.53	0.015	0.277	0.011
5	0.487	0.027	0.989	0.917
6	0.405	0.09	0.877	0.736
7	0.412	0.03	0.962	0.778
8	0.437	0.023	0.932	0.661
9	0.478	0.034	0.816	0.358
10	0.353	0.106	0.773	0.517
Score	**0.47**	**0.05**	**0.68**	**0.39**

Table 3. Comparison of SVM, RF and XGBoost for *DeNovo* sampling.

	SVM		RF		XGBoost	
Group	ROC	PR	ROC	PR	ROC	PR
1	0.96	0.95	0.998	0.996	0.999	0.999
2	0.984	0.973	0.998	0.99	1.000	1.000
3	0.952	0.943	0.999	0.995	1.000	1.000
4	0.277	0.011	0.269	0.008	0.305	0.01
5	0.989	0.917	0.999	0.977	0.999	0.99
6	0.877	0.736	0.957	0.825	0.999	0.998
7	0.962	0.778	0.998	0.979	0.999	0.994
8	0.932	0.661	0.962	0.653	0.994	0.805
9	0.816	0.358	0.894	0.345	0.906	0.52
10	0.773	0.517	0.951	0.844	0.97	0.911
Score	**0.68**	**0.39**	**0.73**	**0.44**	**0.76**	**0.53**

Table 4. Comparison of SVM, RF, and XGBoost with weighted training samples. The last row is added from Table 3 for easier comparison. The best performance scores are highlighted in bold.

Classifier	SVM		RF		XGBoost	
Viral family group	ROC	PR	ROC	PR	ROC	PR
1	0.994	0.992	0.996	0.989	0.999	0.999
2	0.999	0.998	0.995	0.979	1.000	1.000
3	0.970	0.964	0.998	0.993	1.000	1.000
4	0.269	0.011	0.274	0.008	0.289	0.008
5	0.995	0.957	0.997	0.936	0.999	0.991
6	0.964	0.908	0.974	0.877	0.999	0.994
7	0.986	0.932	0.994	0.961	0.998	0.988
8	0.972	0.722	0.98	0.743	0.994	0.870
9	0.910	0.570	0.927	0.633	0.925	0.620
10	0.896	0.667	0.962	0.863	0.957	0.863
With sample weighting	**0.73**	**0.5**	**0.75**	**0.54**	**0.76**	**0.56**
Without sample weighting	0.68	0.39	0.73	0.44	**0.76**	0.53

sample weighting improves prediction performance for all classification techniques especially in terms of the Area under the PR curve. This confirms our hypothesis that the proposed soft *DeNovo* technique for selecting negative examples improves prediction performance by reducing the impact of potential false positives during training.

For SVM, the AUC-PR score for un-weighted training samples is 0.39. It improves to 0.5 when sample weights are used in training. Similarly, the AUC-ROC score improves from 0.68 (without sample weights) to 0.73 with sample level weighting. For RF, the AUC-ROC score for weighted training samples is 0.75 as compared to 0.73 for un-weighted samples. Similarly, the AUC-PR improves

from 0.44 to 0.54 as a consequence of sample weighting. For XGBoost, the weighted AUC-PR improved from 0.53 to 0.56 when the classifier was trained with weighted samples. However, the weighted AUC-ROC remained the same for both unweighted and weighted training samples. It is also important to note that accuracy of group 4 is consistently low for all classifiers. This is because the viral proteins in this group belonging to Flaviviridae, are very dissimilar to the rest of the protein families.

3.4. *Effect of training and test data sizes*

We evaluate the performance of our best performing classifier (XGBoost) with *DeNovo* sampling of negative examples with sample level weights using a balanced ($|P| = |N| = 4971$) and the full data set ($|P| = 4971, |N| = 81,697$). The objective of this experiment is to see the effect of training and test data sizes on classification performance. The negative examples in the balanced set are a random subset of examples from the set of 81,697 *DeNovo* negative examples stratified with respect to their groups. The results of leave one group out cross validation on all training and test combinations of balanced versus full data sets are given in Table 5. It is interesting to note that restricting testing to the balanced test set gives much higher performance scores in comparison to the more realistic full dataset testing. This clearly shows that restricting to a balanced test set for performance evaluation can lead to inflated accuracy values. Furthermore, using the full training set with sample weighting can improve prediction performance over the full test set in comparison to training the classifier on a balanced training set. Therefore, we conclude that HPI predictors should be trained and evaluated using the full set of negative examples rather than restricting to a balanced set. Not only does this give a more realistic assessment of the real world use of HPI predictors, but it can also lead to better generalization performance.

3.5. *Web server implementation*

We train our final HPI predictor using XGBoost on the full dataset using *DeNovo* negative sampling and sample level weighting. We have developed a web server of our HPI predictor for the biologists to check whether a human protein interacts with a viral protein or not. It is called Host-Pathogen Interaction Predictor (HOPITOR). It can be accessed through http://faculty.pieas.edu.pk/fayyaz/software.html#

Table 5. Comparison of AUC-PR and AUC-ROC (in parenthesis) scores for XGBoost trained and tested with balanced and full datasets using leave one group out cross-validation. The best performance scores are highlighted in bold.

	Balanced training set	Full training set
Balanced test set	0.95 (0.94)	0.95 (0.94)
Full test set	0.50 (0.75)	0.56 (0.76)

HoPItor for free. Due to computational requirements, the server is limited to a single protein pair for testing at one time by a single user. However, we also provide the source code of the method for large-scale use.

3.6. *In silico predictions and biological validation*

We use HOPITOR for *in silico* predictions for biologically validated host-pathogen interactions that are not a part of our training set to ensure its generalization performance. The sequences of proteins discussed below are taken from the UniProt[34] repository. Specifically, we use three different viral proteomes to identify their interactions with human STAT1 and STAT2 proteins which are part of the human immune response to pathogens.

3.6.1. *Human STAT2 and HRSV proteins*

Human respiratory syncytial virus (HRSV, family: *Paramyxoviridae*) is the main cause of lower respiratory tract infections.[35,36] HRSV has 11 proteins out of which nonstructural proteins (NS1 and NS2) specifically degrade human STAT2 protein.[37,38] We tested all proteins in the HRSV proteome for interaction with human STAT2 protein on HOPITOR. The predicted probability of interaction between NS1 and human STAT2 is 99% and it ranks at the top among all 11 viral proteins. HOPITOR also predicts 90% probability of NS2 and human STAT2 proteins to interact with each other.

3.6.2. *Human STAT1 and measles proteins*

Measles virus (family: *Paramyxoviridae*) phosphoprotein P blocks the phosphorylation of human STAT1 protein by interacting with it.[39,40] We tested the interactions between all proteins in the Measles virus proteome and human STAT1. The predicted probability of interaction between STAT1 and measles virus protein P is 96.1%. This interaction is ranked second among seven proteins of measles virus. The highest ranked protein is nucleoprotein with probability 96.5%.

3.6.3. *Human STAT1 and rabies proteins*

Rabies virus (family: *Rhabdoviridae*) phosphoprotein interacts with human STAT1 protein to inhibit interferon signal transduction pathways.[41] We tested human STAT1 protein for interaction with all proteins in the rabies virus proteome. The predicted probability of interaction between STAT1 with rabies phosphoprotein is 91.4%. This prediction is ranked second among five proteins of rabies virus. It is important to note that *Rhabdoviridae* family was not part of our training set.

These *in silico* prediction experiments show that HOPITOR can be used as a reliable tool to predict human and viral protein interactions.

4. Conclusions

In this paper, we tried to answer three questions that arise while developing a host-pathogen protein–protein interaction predictor. Below are the conclusions from our findings and contributions:

(1) *DeNovo* sampling is better than random sampling for generating negative examples. We suggest that HPI predictors be trained using *DeNovo* negative examples.
(2) We propose a *soft DeNovo* approach for handling negative examples in training HPI predictors that offers better generalization performance by assigning lower weights to negative training examples that are similar to positive ones.
(3) HPI predictors should be trained using the complete set of negative examples instead of restricting to a balanced dataset.

We have also developed a new HPI predictor called HOPITOR using the XGBoost classifier based on these findings. Our computational results show that HOPITOR can be used for practical applications.

Acknowledgments

The authors thank Fatma-Elzahraa Eid, Virginia Virginia Polytechnic Institute and State University, USA for providing the relevant data for this study and continuous support throughout this study. We are also grateful to Dr. Brian Geiss at Colorado State University, USA in assisting us with the biological validation of our approach.

Abdul Hannan Basit and Amina Asif are supported by PIEAS MS Fellowship and IT and Telecom Endowment Fund, respectively. Wajid A. Abbasi and Sadaf Khan are supported by grant under indigenous 5000 Ph.D. fellowship scheme from the Higher Education Commission (HEC) of Pakistan (PIN: 213-58990-2PS2-046, PIN: 315-12753-2EG3-197). This work is also supported by National Research Program for Universities (NRPU), Higher Education Commission, Pakistan (Project ID: 6085).

References

1. Kessel A, Ben-Tal N, *Introduction to Proteins: Structure, Function, and Motion*, CRC Press, 2010.
2. Braun P, Gingras A-C, History of protein-protein interactions: From egg-white to complex networks, *Proteomics* **12**:1478–1498, 2012.
3. Rigden DJ, *From Protein Structure to Function with Bioinformatics*, Springer Science & Business Media, 2008.
4. Berggård T, Linse S, James P, Methods for the detection and analysis of protein-protein interactions, *Proteomics* **7**:2833–2842, 2007.
5. Waugh DF, Protein-Protein Interactions, in Anson ML, Bailey K, Edsall JT (eds.), *Advances in Protein Chemistry*, Academic Press, pp. 325–437, 1954.

6. Dömling A, *Protein-Protein Interactions in Drug Discovery*, John Wiley & Sons, 2013.
7. Dyer MD, Murali TM, Sobral BW, Computational prediction of host-pathogen protein–protein interactions, *Bioinformatics* **23**:i159–i166, 2007.
8. WHO Press release. *WHO* Available at http://www.who.int/whr/1996/media_centre/press_release/en/. Accessed: 16th October 2016.
9. WHO Press release. *WHO* Available at http://www.who.int/whr/1996/media_centre/press_release/en/index1.html. Accessed: 16th October 2016.
10. Pitre S *et al.*, Computational methods for predicting protein–protein interactions, in Werther M, Seitz H (eds.), *Protein–Protein Interaction*, Springer, Berlin Heidelberg, pp. 247–267, 2008.
11. Nourani E, Khunjush F, Durmus S, Computational approaches for prediction of pathogen-host protein-protein interactions, *Front. Microbiol.* **6**:94, 2015.
12. Dyer MD, Murali TM, Sobral BW, Supervised learning and prediction of physical interactions between human and HIV proteins, *Infect Genet Evol* **11**:917–923, 2011.
13. Eid F-E, ElHefnawi M, Heath LS, DeNovo: Virus-host sequence-based protein–protein interaction prediction, *Bioinformatics* **32**:1144–1150, 2016.
14. Davis FP, Barkan DT, Eswar N, McKerrow JH, Sali A, Host–pathogen protein interactions predicted by comparative modeling, *Protein Sci* **16**:2585–2596, 2007.
15. Berman H, Henrick K, Nakamura H, Markley JL, The worldwide protein data bank (wwPDB): Ensuring a single, uniform archive of PDB data, *Nucleic Acids Res* **35**: D301–D303, 2007.
16. Kshirsagar M, Carbonell J, Klein-Seetharaman J, Techniques to cope with missing data in host–pathogen protein interaction prediction, *Bioinformatics* **28**:i466–i472, 2012.
17. Shen J *et al.*, Predicting protein–protein interactions based only on sequences information, *Proc Natl Acad Sci USA* **104**:4337–4341, 2007.
18. Cui G, Fang C, Han K, Prediction of protein-protein interactions between viruses and human by an SVM model, *BMC bioinformatics* **13**:1, 2012.
19. Abbasi WA, Minhas FUAA, Issues in performance evaluation for host-pathogen protein interaction prediction, *J Bioinform Comput Biol* **14**:1650011, 2016, doi: 10.1142/S0219720016500116.
20. Ben-Hur A, Noble WS, Choosing negative examples for the prediction of protein-protein interactions, *BMC Bioinformatics* **7**:1–6, 2006.
21. Breiman L, Random forests, *Mach Learn* **45**:5–32, 2001.
22. Chen T, Guestrin C, *XGBoost: A Scalable Tree Boosting System*, ACM Press, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
23. Park Y, Marcotte EM, Revisiting the negative example sampling problem for predicting protein–protein interactions, *Bioinformatics* **27**:3024–3028, 2011.
24. Wu Y, Liu Y, Adaptively weighted large margin classifiers, *J Comput Graph Stat* **22**: 416–432, 2013.
25. Chen H, Wang J, Yan X, A fuzzy support vector machine with weighted margin for flight delay early warning, *Fifth Int Conf Fuzzy Systems and Knowledge Discovery, 2008 (FSKD '08)*, pp. 331–335, 2008.
26. Yang X, Song Q, Wang Y, A weighted support vector machine for data classification, *Int J Pattern Recognit Artif Intell* **21**:961–976, 2007.
27. Ravikant DVS, Elber R, Energy design for protein-protein interactions, *J Chem Phys* **135**(6):065102, 2011, doi: 10.1063/1.3615722.
28. Calderone A, Licata L, Cesareni G, VirusMentha: A new resource for virus-host protein interactions, *Nucleic Acids Res* **43**:D588–D592, 2015.
29. Pedregosa F *et al.*, Scikit-learn: Machine learning in python, *J Mach Learn Res* **12**: 2825–2830, 2011.

30. Welcome to Python.org. *Python.org* Available at: https://www.python.org/. Accessed: 16th October 2016.
31. Ben-Hur A, Weston J, A user's guide to support vector machines, *Methods Mol Biol* **609**:223–239, 2010.
32. Friedman JH, Greedy function approximation: A gradient boosting machine, *Ann Statist* **29**:1189–1232, 2010.
33. Davis J, Goadrich M, The relationship between precision-recall and ROC curves, *Proc 23rd Int Conf Machine Learning*, pp. 233–240, 2006.
34. UniProt. Available at: http://www.uniprot.org/. Accessed: 12th May 2016.
35. Sigurs N, Epidemiologic and clinical evidence of a respiratory syncytial virus-reactive airway disease link, *Am J Respir Crit Care Med* **163**:S2–S6, 2001.
36. Tripp RA, Oshansky C, Alvarez R, Cytokines and respiratory syncytial virus infection, *Proc Am Thorac Soc* **2**:147–149, 2005.
37. Elliott J *et al.*, Respiratory syncytial virus NS1 protein degrades STAT2 by using the Elongin-Cullin E3 ligase, *J Virol* **81**:3428–3436, 2007.
38. Lo MS, Brazas RM, Holtzman MJ, Respiratory syncytial virus nonstructural proteins NS1 and NS2 mediate inhibition of stat2 expression and alpha/beta interferon responsiveness, *J Virol* **79**:9315–9319, 2005.
39. Devaux P, Priniski L, Cattaneo R, The measles virus phosphoprotein interacts with the linker domain of STAT1, *Virology* **444**:250–256, 2013.
40. Caignard G *et al.*, Measles virus V protein blocks Jak1-mediated phosphorylation of STAT1 to escape IFN-α/β signaling, *Virology* **368**:351–362, 2007.
41. Vidy A, Chelbi-Alix M, Blondel D, Rabies virus P protein interacts with STAT1 and inhibits interferon signal transduction pathways, *J Virol* **79**:14411–14420, 2005.

Abdul Hannan Basit is currently working for Larkana Institute of Nuclear Medicine & Radiotherapy (LINAR), Larkana, Pakistan as a biomedical engineer. Basit received his MS degree in Systems Engineering from Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan under PIEAS MS Fellowship program. His MS thesis research focuses on "Machine learning in Biomedical Informatics".

Wajid Arshad Abbasi is currently working as a visiting research scholar under the International Research Support Initiative Program (IRSIP) of the Higher Education Commission (HEC) of Pakistan, to carry out part of his PhD research at the Department of Computer Science at Colorado State University (CSU), USA. He is pursuing his PhD in the Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. He has also been awarded the indigenous PhD fellowship by the HEC. His primary area of research is "Machine Learning in Bioinformatics". His webpage is available at http://faculty.pieas.edu.pk/fayyaz/wajid/index.html.

Amina Asif is a PhD student at the Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. Her primary area of research is Applied Machine Learning.

Sadaf Gull is a PhD scholar in the Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. She is doing her PhD under indigenous PhD fellowship by Higher Education Commission (HEC). Her area of research is "Machine Learning in Biomedical Informatics".

Fayyaz Ul Amir Afsar Minhas is a principal scientist with the Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. Dr. Minhas received his PhD degree in Bioinformatics from Colorado State University, USA on a Fulbright Scholarship. He has also been awarded the National Youth Award by the Government of Pakistan for his contributions to science and technology. His lab focuses on applications of machine learning in Bioinformatics and the analysis of biomedical data. His webpage is available at http://faculty.pieas.edu.pk/fayyaz/.