Journal of Bioinformatics and Computational Biology

Cascading classifier application for topology prediction of transmembrane beta-barrel proteins

--Manuscript Draft--

Manuscript Number:	JBCB-1097R2
Full Title:	Cascading classifier application for topology prediction of transmembrane beta-barrel proteins
Article Type:	Research Paper
Corresponding Author:	H B Kazemian London Metropolitan University UNITED KINGDOM
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	London Metropolitan University
Corresponding Author's Secondary Institution:	
First Author:	H B Kazemian
First Author Secondary Information:	
Order of Authors:	H B Kazemian
	CEDRIC MAXIME GRIMALDI
Order of Authors Secondary Information:	
Abstract:	Membrane proteins are a major focus for new drug discovery. Transmembrane beta-barrel proteins play key roles in the translocation machinery, pore formation, membrane anchoring and ion exchange. Given their key roles and the difficulty in membrane protein structure determination, the use of computational modeling is essential. This paper focuses on the topology prediction of transmembrane beta-barrel proteins. In the field of bioinformatics, many years of research has been spent on the topology prediction of transmembrane alpha-helices. The efforts to TMB (transmembrane beta-barrel) proteins topology prediction have been overshadowed and the prediction accuracy could be improved with further research. Various methodologies have been developed in the past for the prediction of TMB proteins topology, however the use of cascading classifier has never been fully explored. This research presents a novel approach to TMB topology prediction with the use of a cascading classifier. The MATLAB computer simulation results show that the proposed methodology predicts transmembrane beta-barrel proteins topologies with high accuracy for randomly selected proteins. By using the cascading classifier approach the best overall accuracy is 75.2% which includes a TMB topology prediction of 82%. The accuracy of 82% is achieved using a two-layers cascading classifier.

Journal of Bioinformatics and Computational Biology

CASCADING CLASSIFIER APPLICATION FOR TOPOLOGY PREDICTION OF TRANSMEMBRANE BETA-BARREL PROTEINS

HASSAN B. KAZEMIAN

School of Computing and Digital Media, Intelligent Systems Research Centre
London Metropolitan University
London, UK
h.kazemian@londonmet.ac.uk

CEDRIC MAXIME GRIMALDI

School of Computing and Digital Media, Intelligent Systems Research Centre
London Metropolitan University
London, UK
rx8879@gmail.com

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Membrane proteins are a major focus for new drug discovery. Transmembrane beta-barrel proteins play key roles in the translocation machinery, pore formation, membrane anchoring and ion exchange. Given their key roles and the difficulty in membrane protein structure determination, the use of computational modelling is essential. This paper focuses on the topology prediction of transmembrane beta-barrel proteins. In the field of bioinformatics, many years of research has been spent on the topology prediction of transmembrane alpha-helices. The efforts to TMB (transmembrane beta-barrel) proteins topology prediction have been overshadowed and the prediction accuracy could be improved with further research. Various methodologies have been developed in the past for the prediction of TMB proteins topology, however the use of cascading classifier has never been fully explored. This research presents a novel approach to TMB topology prediction with the use of a cascading classifier. The MATLAB computer simulation results show that the proposed methodology predicts transmembrane beta-barrel proteins topologies with high accuracy for randomly selected proteins. By using the cascading classifier approach the best overall accuracy is 76.3% with a precision of 0.831 and recall or probability of detection of 0.799 for TMB topology prediction. The accuracy of 76.3% is achieved using a two-layers cascading classifier.

Keywords: Support vector machine, Deep learning, Neural networks, K-nearest neighbours, Cascading classifier, beta-barrel, Topology prediction

1. Introduction

Transmembrane topology prediction problem will be discussed within this paper. A recent paper embarks upon a NN (Neural Network) technique and its comparison with hybrid-two-level NN-SVM (Support Vector Machines) methodology to classify inter-class and intra-class transitions to predict the number and range of beta membrane spanning

regions (Kazemian et al., 2016). The computer simulation results demonstrate a significant impact and a superior performance of NN-SVM tests with a five residue overlap for signal protein over NN with and without redundant proteins for prediction of transmembrane beta-barrel spanning regions. The efforts to beta-barrel topology prediction have been overshadowed and the accuracy of prediction could be improved. Recent studies focus on alpha-helix transmembrane regions prediction with the use of SVM- genetic algorithm (Kazemian et al., 2013) or adaptive neural fuzzy inference system (Kazemian et al., 2014) for example. TMB topology prediction is understudied, and the aim of this paper is to evaluate the performance of a cascading classifier in the prediction of TMB topologies. Datasets used for transmembrane beta-barrel proteins are usually of small size. A recent publication by Sharma et al. (2016) evaluates the performance of various machine learning techniques based on small datasets with varying dimensionalities. From their study, they concluded that KNN (*k*-nearest neighbours), SVM and linear discriminant have the best predictive accuracy on small datasets. One of the latest methods used for predicting transmembrane beta-barrel topologies is BOCTOPUS (Hayat et al., 2012). In 2016, Hayat et al. introduced BOCTOPUS2 (Hayat et al., 2016), an improved version of BOCTOPUS. The correct topology is predicted correctly in 69% of the proteins with BOCTOPUS2. It is more than 10% improvement compared to BOCTOPUS, the earlier method.

This research takes the transmembrane beta-barrel topology prediction further by applying novel techniques such as DNN, KNN and SVM as part of a cascading classifier. The computer simulation results show new results for TMB topologies prediction using a cascading classifier.

2. Machine Learning approaches

The cascading classifier presented as part of this paper consists of combinations of various machine learning techniques including KNN, DNN and SVM. The model allows to use two methods or even three methods. Use KNN =1 or 0, use SVM = 1 or 0 and use NN = 1 or 0 are the list of parameters. Each machine learning techniques have their own characteristics that are summarized in this chapter.

2.1. *K-Nearest Neighbours (KNN)*

Among all machine learning algorithms, Nearest Neighbours algorithms are the simplest. KNN is particularly well suited for multi-modal classes as well as applications in which an object can have many class labels. Datasets used for transmembrane beta-barrel proteins are usually of small size. In the MATLAB implementation presented as part of this paper, KNN is one of three machine learning techniques that can be used as part of the cascading classifier within a combination. At the input layer, a sliding encoding window will be used on each amino acid sequence. Prediction is based on the topology characteristic of the central residue in the window. A binary array of size 20 is used to encode each window position at the start of the implementation. Several model parameters

have been modified such as the tie-breaking algorithm, the nearest neighbour search method, *k* value, maximum data points in node, tie inclusion flag, distance metrics and exponent.

2.2. *Deep Neural Network (DNN)*

Deep neural networks have become popular machine learning tools in recent years. In a recent paper, Heffernan et al. (2015) achieved a secondary structure prediction accuracy of 82% by using a deep learning neural network. Deep neural networks are able to learn complex patterns. For the MATLAB implementation presented as part of this paper, when a DNN is used part of a combination, the pattern recognition network variable is used for the creation of a pattern recognition neural network. An input layer, a hidden layer and an output layer are used to define the neural network. At the input layer, a sliding encoding window is used on each amino acid sequence.

2.3. *Support Vector Machines (SVM)*

Another machine learning technique used as part of the cascading classifier is SVM. In the field of machine learning, a Support Vector Machine is a supervised learning technique that can be used for both classification and regression. SVMs don't over generalize in general whereas the neural networks can lead to over generalization often (Mitchell, 1997). The performance of SVM depends largely on the kernels chosen. The best kernel of choice for a specific problem had to be researched for this implementation. Smola et al. (Smola, 1998) provided an explanation of the relation between the standard regularization theory and the SVM kernel method. Other problems of SVMs, for the training and testing phases include size and speed. For a similar generalization performance, other neural networks are faster than SVMs (Haykin, 1998).

2.4. *Ensemble methods*

There has been an increasing use of ensemble learning methods in recent research in computational biology. Ensemble learning combines multiple learning algorithms in order to improve the overall prediction accuracy (Dietterich, 2000). It is one of the most promising solutions for many biological problems. Ensemble of SVMs has been evaluated in a study for robust microarray data classification (Peng, 2006). When a comparison was done between a single SVM classifier and the ensemble of bagging and boosting, the author observed that the proposed clustering based SVM ensemble obtained the best result. Cascading is a specific case of ensemble learning. It is based on concatenation of several classifiers using the information provided from the output of a given classifier as additional information for the next classifier in the cascade. Kazemian et al. presented in a recent study (Kazemian et al., 2014) a cascaded SVM-NN classification methodology for signal peptide discrimination and cleavage site identification. The overall accuracy achieved was 91.5% based on cross-validation tests using the SVM-NN model. A study on the TMB

topology prediction using a cascading classifier has not yet been evaluated, hence the interest of this paper.

3. Data preparation

3.1. *Datasets*

There are a number of databases that are available and are repositories for the structures and sequences of transmembrane proteins. TOPDB (Topology data bank of transmembrane proteins) contains a comprehensive list of transmembrane proteins with topology information (Tusnády, 2008). It is the most complete and comprehensive collection of transmembrane protein datasets containing experimentally derived topology information. The database collects the details of various experiments carried out to learn about the topology of particular transmembrane proteins. The experimental techniques include fusion with reporter enzymes, glycosylation studies, protease accessibility and immunolocalization. It has a total of 4190 transmembrane proteins obtained from the literature and from public databases available on internet. Data derived from literature cannot be collected automatically but data based on 3D structures generates semi-automatic and continuously updated information for the database. For each protein in the database, the most probable topology consistent with the collected experimental constraints is also calculated using multiple transmembrane topology prediction algorithms for alpha-helices and beta-barrel transmembrane proteins respectively. The web interface includes tools for extensive searching, relational querying and data browsing as well as visualization tools for topology data. The beta-barrel TOPDB entries can be downloaded directly from the web interface. The topdb_bp.txt file contains 123 TMB sequences. The table below represents a sample of 15 TMB out of the 123 TMB proteins used for this implementation.

Table 1. Sample of 20 TMBs selected out of 123 TMBs available in TopDB database

	ID	Description	Organism
1	**BP00056**	Outer membrane usher protein faeD precursor	Escherichia coli
2	**BP00086**	Maltoporin precursor	Escherichia coli
3	**BP00115**	Outer membrane protein A precursor	Escherichia coli
4	**BP00124**	Outer membrane pore protein E precursor	Escherichia coli
5	**BP00193**	Major outer membrane protein P.IA precursor, PIA	Neisseria gonorrhoeae
6	**BP00272**	Major outer membrane protein P.IB precursor, PIB	Neisseria gonorrhoeae
7	**BP00273**	Major outer membrane protein P, PIA	Neisseria meningitidis serogroup B
8	**BP00274**	Outer membrane protein class 2	Neisseria meningitidis
9	**BP00310**	Sucrose porin precursor	Salmonella typhimurium
10	**BP00320**	Alpha-hemolysin precursor (Alpha-toxin) (Alpha-HL)	Staphylococcus aureus
11	**BP00339**	Porin	Rhodobacter blasticus
12	**BP00345**	Outer membrane protein F precursor (Porin ompF) (Outer membrane protein 1A) (Outer membrane protein IA) (Outer membrane protein B)	Escherichia coli
13	**BP00346**	Ferrichrome-iron receptor precursor (Ferric hydroxamate uptake) (Ferric hydroxamate receptor)	Escherichia coli
14	**BP00359**	Outer membrane porin protein 32 precursor (OMP32)	Delftia acidovorans
15	**BP00364**	Outer membrane protein tolC precursor	Escherichia coli

The BOCTOPUS2 dataset is the second dataset used for the implementation. It is the dataset that was used for the training/testing of the software/predictor BOCTOPUS2 which is a transmembrane beta-barrel topology prediction tool (Hayat, 2012). It is available on the website of BOCTOPUS2 server. The BOCTOPUS2 dataset consists of 42 TMB sequences.

3.2. *Data collection*

All residues in the data sets were annotated as either "I" (Inner-loop), "O" (Outer loop) or "M" (Transmembrane beta-strand) based on the coordinate of the C-alpha atoms and membrane boundaries obtained from the OPM (Orientations of Proteins in Membranes) database (Lomize, 2006). Residues located within the membrane boundaries but do not belong to a transmembrane beta-strand are labelled as "I" or "O" based on the location of the initial residue. On the BOCTOPUS2 server website, supplementary information is provided with the BOCTOPUS2 dataset (42 proteins) available in the file named boctopus2_crossvalidation_dataset.xlsx and the sequences and their annotation used for training/testing BOCTOPUS2 named boctopus2_dataset_sequenceannotation.txt.

For the implementation, data was curated manually, and two files were created for use in MATLAB: boctopus2Sequence.txt that list all sequences and boctopus2Labels.txt that list all labels. The load file created out of those two files was named boctopus2dataset.mat. Regarding TOPDB dataset, the data files were also manually curated. A download for sequences and topologies is available on the Topology Data Bank of Transmembrane Proteins website. All TOPDB entries are available in one file called top_all.txt. The file was divided manually into two separate files named TOPBPLabels.txt and TOPBPSequence.txt. The observed topology represented with an X corresponds to the signal peptide. For the implementations, the signal peptide was ignored. The process of curation was similar to the BOCTOPUS2 dataset. The load file created out of those two files was named TOPBPdataset.mat. It contains 123 proteins.

3.3. *Data pre-processing*

In order to train the cascading classifier, datasets needed to be created and formatted for MATLAB. Structure arrays were created with a small program. A 1x42 structure array with three fields (header, sequence, and topology) was created for BOCTOPUS2 and a 1x123 structure array was created for the TOPDB dataset. The name of the structure arrays and the load files for the implementation are called TOPBPdataset.mat and Boctopus2dataset.mat. The fields include 'header' which corresponds to the annotation of a given protein sequence, 'sequence' which represents the protein sequence and 'topology' which represents the predicted topology.

4. Computer simulation implementation and results

The computer simulation was executed in MATLAB. The approach is classified as an experimental study. A cascading classifier is built and parameters are modified and optimized in order to achieve the best possible performance in the context of TMB topology prediction.

4.1. *Creating and training the cascading classifier*

Data division for the cascading classifier is defined by the model. Data division is based on two parameters corresponding to the fraction of first level training set and fraction of second level training set. The fraction of first level training set parameter defines how many data will be used in total for training. So, if we choose a fraction of first level training set equals to 0.8, it means 80% of data will be used for training and 20% for testing. From the selected 80%, the fraction of second level training set defines which part will be used by the first layer and which part will be used by the second layer. For fraction of first level training set equals 0.8 and the fraction of second level training set equals 0.5, if we have a total a 100 data, 80 will be used for training in total. Among them, 50%=40 will be used for the first layer and 50%=40 will be used for the second layer. When not using DNN, the fraction of the first level can be set manually.

The model consists of two levels. Several selected models will be trained at the first level. The selected algorithms (KNN, SVM or DNN) are trained to predict the values of the class. In case two or more classifiers are selected, a second level SVM classifier is trained to predict the value of the class based on the probability predicted by those two or more models at the first level. A block diagram of the cascading classifier is presented in Fig. 1 below:

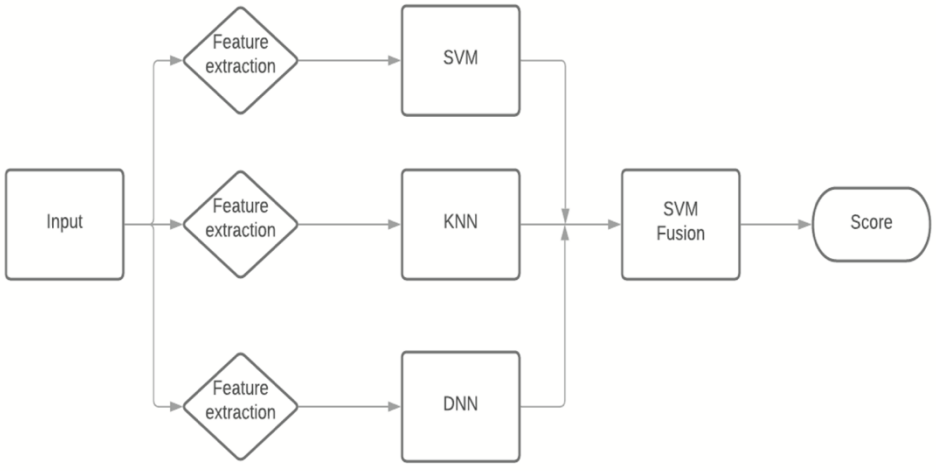

Fig. 1. Block diagram of the cascading classifier.

This is a cascading classifier as the output of the first layer corresponds to the input of the second layer. In case, only one model is initially selected at the first level, the second layer classifier will not be trained.

The model allows to use a single method, two methods or even three methods. KNN =1 or 0, SVM = 1 or 0, and DNN = 1 or 0 are the list of parameters. If a parameter is equals to 0, it will not be used in the cascading classifier. All combinations are possible except for combinations when all of them are equal to zero. If only one of them is set to one, the application will work as one classifier. The selected classifier (equals to one) will be trained and will show the results. If at least two parameters are set to one, it means that several classifiers will be trained and then combined together by one more probability classifier. When multiple classifiers are chosen, a second level SVM will be trained.

4.2. *Modifying the cascading classifier parameters*

The model has four different basic configurations. The parameters of each machine learning algorithms can be modified.

When KNN has been used as part of the cascading classifier, various parameters have been modified. One parameter is the *k* value (number of nearest neighbours). The classifier performs better with more than one neighbour and the best results are reached when the *k* value equals eleven. The tie-breaking algorithm has been modified using the smallest index among tied groups, using the class with the nearest neighbour among tied groups and using a random tie breaker. The tie inclusion flag as well as the maximum number of data points in each lead node of the *k*d-tree have been modified. Nearest neighbour search method has been modified by creating and using a *k*d-tree to find the nearest neighbours and by using the exhaustive search algorithm. Distance metrics used include the City block distance, Chebyshev distance, one minus the sample linear correlation between observations, one minus the cosine of the included angle between observation and the Euclidean distance. The Euclidean distance is the most widely used distance metric in KNN and is the default distance metric used during the implementation. In Cartesian coordinates, if $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ are two points in Euclidean n-space and n a positive integer, then the distance (d) from x to y or y to x is given by the Pythagorean formula:

$$d(x, y) = d(y, x) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \quad (1)$$

When DNN has been used as part of the cascading classifier, various hidden layer sizes have been used from two to 1000. Various training algorithms have been used with the computation in order to evaluate the accuracy of prediction. The scaled conjugate gradient is the default training algorithm available in MATLAB. Other available trainings used included resilient backpropagation algorithm, Conjugate Gradient with Powell/Beale Restarts, Fletcher-Powell Conjugate Gradient in which weight and bias values are updated according to the conjugate gradient backpropagation with Fletcher-Reeves updates (Fletcher, 1964), Polak-Ribiére Conjugate Gradient Training algorithm in which weight and bias values are updated according to the conjugate gradient backpropagation with

Polak-Ribiére updates, Levenberg-Marquardt algorithm, One Step Secant algorithm, and Variable Learning Rate Backpropagation algorithm. The Scaled Conjugate Gradient provides the best results.

The transfer function was modified. Log-Sigmoid allows the signals received from the input layer to be transformed in each hidden layer. Log-Sigmoid was used and provided better results than the hyperbolic tangent sigmoid transfer function. Training occurred according to the training parameters defined such as maximum number of epochs, maximum time to train in seconds, minimum performance gradient or maximum validation failures for example. When one of the conditions defined is met, the training process will stop. It can be for example that the maximum number of epochs also referred as the number of repetitions is reached, the maximum amount of time is exceeded or the performance is minimized to the goal defined. The number of validation checks are used to terminate the training. The gradient will become very small as the training reaches a minimum of the performance. The value of the minimum performance gradient was set to 1e-6. If the magnitude of the gradient was less than 1e-6, the training would have stopped. This was not applicable for this implementation. The number of validation checks represents the number of successive iterations that the validation performance fails to decrease. When the number reached six, which was the value set for the maximum validation failures, the training stopped.

Various functions were used for the data division for the DNN. One problem that occurs during neural network training is data overfitting, where the network tends to memorize the training examples without learning how to generalize to new situations. The default method for improving generalization is called early stopping and consists in dividing the available training data set into three subsets. The training set which is used for computing the gradient and update the network weights and biases, the validation test whose error is monitored during the training process because it tends to increase when data is over-fitted and the last subset is the test set whose error can be used to assess the quality of the division of the data set. One configuration used for data division included division into three sets using random indices. The ratio that is used by default is 0.6/0.2/0.2. It corresponds to the ratio for training, testing and validation. The data is randomly divided so that 60% of the samples are assigned to the training set, 20% to the validation set, and 20% to the test set. Other configurations included data divided into three sets using blocks of indices or into three sets using specified indices. The data division is an automatic process that happens when the network is trained.

When SVM has been used as part of the cascading classifier, the optimization routine is a parameter that was modified. It is specified as Iterative Single Data Algorithm by default or using quadratic programming to implement $L1$ soft-margin minimization by quadratic programming or Sequential Minimal Optimization (Fan et al., 2005). The default is Iterative Single Data Algorithm, if the expected proportion of outliers in the training data is set to a positive value for two-class learning. Other parameters modified included Box constraint that helps prevent overfitting, Cache size, flag to clip alpha coefficients, tolerance for gradient difference, feasibility gap tolerance, maximum number of numerical

optimization iterations, Kernel offset parameter, Kernel scale parameter, Karusch-Kuhn-Tucker complementary conditions violation tolerance, number of iterations between optimization diagnostic message output, expected proportion of outliers in training data, store support vectors, their labels, number of iterations between reductions of active set, flag to standardize data, verbosity level and kernel function using Gaussian or Radial Basis Function (RBF), linear kernel or polynomial kernel.

Polynomial kernel function order is another parameter that has been modified. The default value is three. For this implementation polynomial SVMs have been used as layer one and two of the cascading classifier. The polynomial kernel is a kernel function that is used often with SVMs or kernelized models. It represents the similarity of vectors (training sample) in a feature space over polynomials of the original variables, allowing learning of non-linear models. The polynomial kernel looks not exclusively at the given features of input samples for determination of their similarity, but it looks also at combinations of these. For regression analysis, the combinations refer to interaction features. The feature space of a polynomial kernel equals that of polynomial regression, but without the combinatorial blow-up in the number of parameters to be learned.

The order of polynomial in mathematics refers also to the degree of a polynomial, that is, the largest exponent (for a univariate polynomial) or sum of exponents (for a multivariate polynomial) in any of its monomials.

The following names are given to polynomials according to their degree. Degree 0 corresponds to non-zero constant, degree one is linear, degree two is quadratic, degree three is cubic, degree four is quartic and so on. In general, SVM works well on small datasets. A recent study evaluated the performance of SVMs with linear, quadratic and cubic kernels in the problem of recognizing 3D objects from 2D views. The paper indicates that the degree of the polynomial order plays a minor role in the final results (Dos Santos et al., 2002).

4.3. *Results of cascading classifier*

Multiple runs have been executed in MATLAB using different parameters configurations. Various ratio combinations such as 50:50, 60:40, 70:30, 75:25 and 80:20 have been used for the split between training and testing data in the fraction of first level training set. Best results for the cascading classifier were obtained using a split with 80% of data used for training and 20% used for testing in the fraction of first level training set and 42% in the fraction of second level training set. Best results are obtained with parameters configured to a window size of 55, hidden layer size of 55, log sigmoid transfer function, scaled conjugate gradient for the training function, Mean squared normalized error performance function and data division into three sets using random indices for the DNN part of the cascading classifier and *k*-value of eleven, exhaustive nearest neighbour search method, random tie-breaking algorithm for the KNN part of the cascading classifier. For the SVM part of the cascading classifier, a polynomial kernel function. The model was set to store the support vectors, their labels and the estimated α coefficients. The default values were used for the box constraint, cache size, solver, tolerance to gradient difference,

feasibility gap tolerance, maximal number of optimization iterations, kernel offset parameter, kernel scale, Karush-Kuhn-Tucker complementarity conditions violation tolerance, number of iterations between optimization diagnostic message output, expected proportion of outliers in training data, polynomial kernel function order, number of iterations between movement of observations from active to inactive set, flag to standardize predictor data and verbosity level.

The performance of the model was evaluated with the use of ROC curves, confusion matrix and bar charts. The Receiver Operating Characteristic (ROC) is a plot of the true positive rate (sensitivity) versus the false positive rate (1 - specificity). Fig. 1 represents the confusion matrix for the cascading classifier. We examine the confusion matrix by considering the outputs of the trained classifier and comparing them to the expected results (targets). Green blocks (diagonal cells) show the number (and percentage) of class samples/residues positions in the dataset that were correctly classified. Red blocks show misclassifications (false positives in the upper right, false negative in the lower left). For example, based on fig. 2, 704 examples of class three are wrongly classified as class one. Grey blocks give the percentages of correct classification in relation to the respective class.

Fig. 2. Confusion matrix

The receiver operating characteristic for each output class is plotted with the plot receiver operating characteristic. When the curve goes to the left and top edges of the plot it means that the classification is better. The sensitivity measures the proportion of actual positives that are correctly identified as such. The false positive is also known as the fall-out. Fall-out is closely related to specificity and is equal to (1 - specificity). The steepness

of ROC curve is also important since it is ideal to maximize the true positive rate while minimizing the false positive rate.

The ROC curve (Fig. 3) is thus the sensitivity as a function of the fall-out. A perfect predictor would be described at 100% sensitive. The closer the ROC curve is to the upper left corner (100% sensitivity, 100% specificity), the higher the overall accuracy.

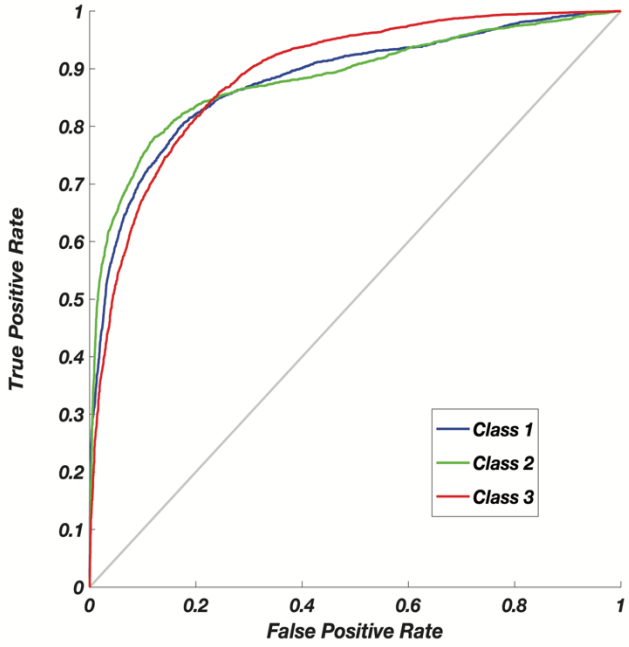

Fig. 3. ROC curve

Topologies predictions can be evaluated in more detail by calculation of prediction quality indices (Kabsch, 1983). Fig. 4 indicates how well a particular topology was predicted and wether overprediction or underprediction have occurred. The blue column represents the computed fraction of correct predictions when a given topology is observed (in other words, the number of residues correctly predicted for topology I, O or M, divided by the number of residues observed) . The red column represents the computed fraction of correct predictions when a given topology is predicted (in other words, the number of residues correctly predicted, divided by the number of residues predicted).Those quality indices are important for the interpretation of the prediction accuracy.The fraction correct of predicted (red column) is practically useful in predicting unknown topologies. Results indicate that overprediction and underprediction is limited, which suggests that the cascading classifier can effectively predict. Bar (x, y) is the MATLAB function that was used to create Fig. 4.

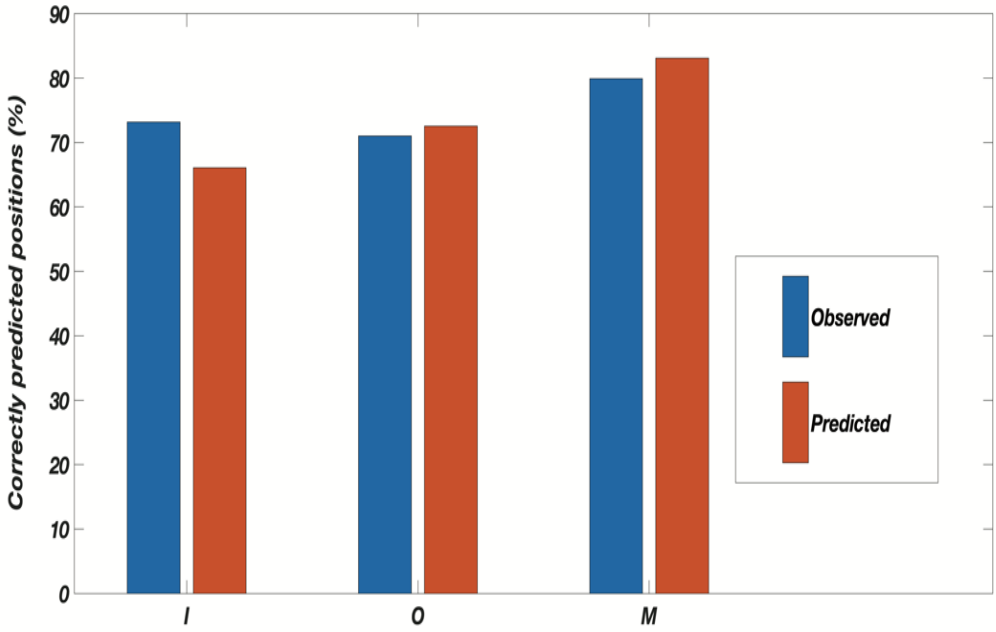

Fig. 4. Prediction quality indices

5. Conclusion

This paper discusses the applications of a cascading classifier that consists of multiple machine learning algorithms including KNN, DNN and SVM for TMB topology prediction. Training and testing were performed on curated TOPDB and BOCTOPUS2 datasets. The model allows to use a single method, two methods or even three methods by selecting KNN =1 or 0, SVM = 1 or 0, and DNN = 1 or 0. The computer simulation results using the TOPDB dataset that include 123 TMB sequences respectively generates an overall topology prediction accuracy of 76.3% with a precision of 0.831 and recall or probability of detection of 0.799 for TMB topology prediction. The accuracy of 76.3% is for one scenario combination where layer one includes SVM, KNN and DNN, and layer two include SVM. When used individually, best accuracy from DNN, SVM and KNN are 70%, 64% and 71.8% respectively. The output of layer one is the input of layer two in the cascading classifier. The algorithms were optimized by varying the parameters. Various ways to improve the performance of the deep neural network part of the cascading classifier were implemented including checking for overfitting. It is important to ensure that the deep neural network does not over-fit. Few techniques were used to avoid overfitting including early stopping. It precipitates the training of the deep neural network leading to a reduction

in error in the test set. Checking for overfitting was one way to improve the performance of the DNN. Few other parameters were modified for the DNN including the number of layers in the neural network, the activation function used, the training function and/or performance function. The scaled conjugate gradient backpropagation has been proven to provide the best accuracy for the implementation. It provides faster training with excellent test efficiency. Few parameters have also been modified for the KNN part of the cascading classifier. The use of the Euclidean distance has been proven to provide the best accuracy for the implementation. Various values of *k* have also been used. A value of *k*=11 has been proven to provide the best accuracy for the implementation. SVMs were initially designed for binary classification. The multi-class classification problem was decomposed into a series of binary problems. The implementation of the SVMs as part of the cascading classifier in MATLAB used the support vector machine template. Various parameters of SVMs have been used and tuned in order to improve the performance. Parameter C represents the error penalty for misclassification for SVMs. The parameter C is defined as box constraint in MATLAB and various values were used. Increasing Box constraint decreases the number of support vectors, but it also increases training time. During SVMs training, SVMs kernels and its parameters have very important role for classification accuracy. Various kernels were used. The most efficient kernel was the polynomial kernel with polynomial order three. It is aligned with the findings of the literature on this topic. The polynomial degree parameter controls the flexibility of the decision boundary. Higher degree kernels yield a more flexible decision boundary. For SVM, in order to avoid overfitting, a soft Margin needs to be chosen instead of a hard one. Gamma (γ) is an important parameter and it controls overfitting in SVM. Gamma is not technically an SVM parameter. It is a parameter of the Kernel. Gamma (γ) is referred as the kernel scale parameter in MATLAB. Various kernel scale parameters have been used and best results are obtained with kernel scale parameter equals 1.

Overall, by constructing and using various machine-learning frameworks as part of the cascading classifier, a system has been developed and could predict the TMB topologies with significant robustness in comparison to other classifiers. In comparison, PRED-TMBB2 is a method recently developed based on Hidden Markov Models yielding 76% accuracy for correct topology predictions (Tsirigos, 2016). BOCTOPUS2 (Hayat et al., 2016) is another recent study that has the topology predicted correctly only in 69% of the proteins. The cascading classifier has also a better prediction accuracy compared to BOCTOPUS (Hayat et al., 2012). The method presented in this paper is innovative and represents an improvement in the prediction of beta-barrel transmembrane topology prediction. Many important biological processes are mediated by transmembrane proteins. The methodology could be applied to any TMB proteins and could potentially help to identify new targets for antibiotics, vaccines and antimicrobials.

Acknowledgments

This research work has been carried out at Intelligent Systems Research Centre, School of Computing and Digital Media, London Metropolitan University. The project ‘Cascading

classifier application for topology prediction of transmembrane beta-barrel proteins' is available on Gitlab (https://gitlab.com/CMG1101/cascading-classifier-application-for-topology-prediction-of-transmembrane-beta-barrel-proteins)

.

References

1. Dietterich TG, Ensemble Methods in Machine Learning, *Multiple Classifier Systems*, pp. 1–15, 2000.
2. Dos Santos EM, Gomes HMA, Comparative Study of Polynomial Kernel SVM Applied to Appearance-Based Object Recognition, *Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, pp. 408-418, 2002.
3. Fan RE, Chen PH, Lin CJ, Working set selection using second order information for training support vector machines, *Journal of Machine Learning Research* **6**:1889–1918, 2005.
4. Fletcher R, Reeves CM, Function minimization by conjugate gradients, *The Computer Journal* **7**:149–154, 1964.
5. Hayat S, Elofsson A, BOCTOPUS: improved topology prediction of transmembrane β barrel proteins, *Bioinformatics* **28**:516–522, 2012.
6. Hayat S, Peters C, Shu N, Tsirigos KD, Elofsson A, Inclusion of dyad-repeat pattern improves topology prediction of transmembrane β-barrel proteins, *Bioinformatics* **32**:1571–1573, 2016.
7. Haykin S, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
8. Heffernan *et al*., Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning, *Scientific reports* **5**:11476, 2015.
9. Kabsch W, Sander C, How good are predictions of protein secondary structure?, *FEBS Letters* **155**:179–182, 1983.
10. Kazemian HB, White K, Palmer-Brown D, Applications of evolutionary SVM to prediction of membrane alpha-helices, *Expert Systems with Applications* **40**:3412–342, 2013.
11. Kazemian H, Yusuf SA, An ANFIS approach to transmembrane protein prediction. IEEE World Congress on Computational Intelligence (IEEE WCCI 2014), IEEE International Conference on Fuzzy Systems, Beijing China, pp.1360-1365, 2014.
12. Kazemian HB, Yusuf SA, White K, Signal peptide discrimination and cleavage site identification using SVM and NN, *Computers in Biology and Medicine* **45**:98–110, 2014.
13. Kazemian H, Yusuf SA, White K, and Grimaldi CM, NN approach and its comparison with NN-SVM to beta-barrel prediction, *Expert Systems with Applications* **61**:203–214, 2016.
14. Kecman V, Huang TM, Vogt M, Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance, *Support Vector Machines: Theory and Applications*, pp. 255–274. Berlin: Springer-Verlag, 2005.
15. Lomize MA, Lomize AL, Pogozheva ID, Mosberg HI, OPM: orientations of proteins in membranes database, *Bioinformatics* **22**:623–625, 2006.
16. Mitchell TM, *Machine Learning*, McGraw-Hill, 1997.
17. Peng Y, A novel ensemble machine learning for robust microarray data classification, *Computers in Biology and Medicine* **36**:553–573, 2006.
18. Reich JC, An Iterative Feature Perturbation Method for Gene Selection from Microarray Data, PhD Thesis, University of South Florida, Tampa, FL, USA**,** 2010.
19. Sharma S, Sharma V, Performance of Various Machine Learning Classifiers on Small Datasets with Varying Dimensionalities, *Circulation in Computer Science* **1**:30-35, 2016.

20. Smola AJ, Schölkopf B, Müller KR, The connection between regularization operators and support vector kernels, *Neural Networks* **11**:637–649, 1998.
21. Tsirigos KD, Elofsson A, Bagos PG, PRED-TMBB2: improved topology prediction and detection of beta-barrel outer membrane proteins, *Bioinformatics*, **32**:i665–i671, 2016.
22. Tusnády GE, Kalmár L, Simon I, TOPDB: topology data bank of transmembrane proteins. *Nucleic Acids Research* **36**:234–239, 2008.

Responses to reviewers' comments

Reviewer #1:

Authors seem to have answered most of the concerns from original submission. Few unanswered questions as below.

1) This article uses cascading classifier with KNN, DNN and SVM components for TMB topology prediction. The unanswered question is regarding the other well- known / popular machine learning approaches. What if KNN and/or DNN and/or SVM components of cascading classifier are replaced with any other machine learning approach e.g. HMM, etc. There are many ML approaches available to test. Will that change the performance of TMB topology prediction?

The cascading classifier has been built using only KNN, DNN and SVM at the first level and SVM at the second level. Technically, it is possible to include HMM by creating an additional code, but this is not in the scope of this paper. SVM and KNN were chosen based on a recent study (Sharma et al, 2016) demonstrating that SVM and KNN provide the best results when using small datasets. Deep neural networks have become popular machine learning tools in recent years and the use of DNN within a cascading classifier has not been studied in this context, that is why it was decided to include DNN in the classifier.

2) It will be valuable for the article and very useful for the reader to present the experimental workflow in the form of a flow-diagram.

A block-diagram (Figure 1) has been added to the paper. The figure is 300 dpi and CMYK.

3) The figures appear to be low-resolution, some compression in making PDF is probably affection figure quality. It's hard to read the axes-labels. Please see the attached document with a presentable example of Fig. 4 from this manuscript.

We have decided to remove the previous validation checks (Figure 1) and replace it with the block diagram (Figure 1). The current figures properties have been updated with MATLAB and Adobe Illustrator and they are 300dpi and CMYK. The figure numbers and references in the paper have also been updated, as requested.

4) Authors mentioned about GitLab, no link to GitLab page was found.

We have created a Gitlab project. Project ID: 19396702. Please see the link below. The project is public. An access to Gitlab project can be requested. The URL below is sometimes unstable.

https://gitlab.com/CMG1101/cascading-classifier-application-for-topology-prediction-of-transmembrane-beta-barrel-proteins

References :

1. Sharma S, Sharma V, Performance of Various Machine Learning Classifiers on Small Datasets with Varying Dimensionalities, *Circulation in Computer Science* **1**:30-35, 2016.