Intelligent Consensus Modeling for Proline Cis-Trans Isomerization Prediction

Paul D. Yoo, Sami Muhaidat, Kamal Taha, Jamal Bentahar, Abdallah Shami

Abstract—Proline cis-trans isomerization (CTI) plays a key role in the rate-determining steps of protein folding. Accurate prediction of proline CTI is of great importance for the understanding of protein folding, splicing, cell signaling, and transmembrane active transport in both the human body and animals. Our goal is to develop a state-of-the-art proline CTI predictor based on a biophysically motivated intelligent consensus modeling through the use of sequence information only (i.e., position specific scores generated by PSI-BLAST). The current computational proline CTI predictors reach about 70–73% Q2 accuracies and about 0.40 Matthew Correlation Coefficient (Mcc) through the use of sequence-based evolutionary information as well as predicted protein secondary structure information. However, our approach that utilizes a novel decision tree-based consensus model with a powerful randomized-metalearning technique have achieved 86.58% Q2 accuracy and 0.74 Mcc, on the same proline CTI dataset, which is a better result than those of any existing computational proline CTI predictors reported in the literature.

Index Terms—proline cis-trans isomerization; machine-learning; intelligent systems; ensemble methods;

1 Introduction

It remains an importance and relevant problem to accurately predict proline cis-trans isomers of proteins based on their amino acid sequences only. The importance of the cis-trans isomerization (CTI) as rate-determining steps in protein folding reactions has been well reported in the literature [1–3]. Prolyl CTI can be catalyzed by prolyl isomerizes, an enzyme found in both prokaryotes and eukaryotes that interconverts the cis-trans isomers of peptide bonds with the amino acid proline [4]. These enzymes are involved not only in the catalysis of folding [4–5] but also in regulatory process [6–7]. The CTI of prolyl peptide bonds has been suggested to dominate the folding of the alpha subunit of tryptophan synthase from Escherichia coli (aTS) [8]. A CTI, which is necessary to achieve the final conformational state of the prolyl bonds for such proteins, has often been found as the rate-limiting step in vitro protein folding [9].

The international research effort called Human Genome Project started in 1990s has produced a massive amount of biological data, and consequently, accurate and efficient computational modeling methods that can find useful patterns from the massive data have gained much attention. The first attempt to predict the CTI of proline using a computational model from amino acid sequences was made by Frömmel and Preissner in 1990 [10]. They had taken adjacent/local residues (±6) of prolyl residues and their physicochemical properties into account, and found six different patterns that allow one to assign correctly about 72.7% (176 cis-prolyl residues in their relatively small dataset of 242 Xaa-Pro bonds of known cis-prolyl residues), where by no false positive one is predicted.

Since Frömmel and Preissner's seminal work, support vector machines (SVMs) seemed to be the most suitable for proline CTI prediction task. The first SVM-based computational predictor was built by Wang et al [11]. They constructed a SVM with polynomial kernel function and used amino acid sequences as input, and achieved the Q2 accuracy of 76.6%. Song et al [12] also built a SVM with radial basis function, and used evolutionary information represented in position-specific-scoring matrix (PSSM) scores generated by PSI-BLAST [13] and predicted secondary structure information obtained from PSI-PRED [14] as input. They reached the Q2 accuracy of 71.5%, and Mcc of 0.40. Pahlke et al's [15] showed the importance of protein secondary structure information in the prediction of proline CTI residues. Their computational algorithm called COPS—the first attempt to predict for all 20 naturally occurring amino acids whether the peptide bond is a protein is in *cis* or *trans* conformation—used secondary structure information of amino acid triplets only. Most recently, Exarchos et al [16] used a SVM with a wrapper feature selection algorithm, on evolutionary information (i.e., PSSM scores), predicted secondary structure information, real-valued solvent, and accessibility level for each amino acid, and the physicochemical properties of the neighboring residues as input. They achieved 70% accuracy in the prediction of the peptide bond conformation between any two amino acids only.

As seen above, the recent development of computational modeling for proline CTI prediction has mostly been based on SVM and its variants, and evolutionary (i.e., PSSM scores), and secondary structure information as input. These models showed about 70–73% Q2 accuracies and 0.40 Mcc. This observation is aligned with the results of other computational biology studies [17–21]. SVMs showed great results

- *P.D. Yoo, S. Muhaidat and K. Taha are with Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi. E-mail: {paul.yoo, sami.muhaidat, kamal.taha}@kustar.ac.ae*
- *J. Bentahar is with Concordia Institute for Information Systems Engineering, Concordia University, Canada, E-mail: bentahar@ciise.concordia.ca*
- *A. Shami is with the Department of Electrical and Computer Engineering, The University of Western Ontario, Canada, E-mail: ashami@eng.uwo.ca*

in the prediction/classification tasks in the fields of computational biology and bioinformatics [17–21].

In this paper, we introduce a novel approach that utilizes biophysically-motivated intelligent consensus model (Method I) with a powerful randomized metalearning technique (Method II) through the use of sequence information only (i.e., PSSMs generated by PSI-BLAST) for the accurate and efficient prediction of proline CTI residues. The proposed model has been built based on the idea of RandomForest data modeling [22], and evolutionary information, and its predictive performance is compared with the most widely used SVM and its variants on the same dataset as used in Song et al's study.

2 Methods

Our experiment consists of four consecutive phases. First, collect and pre-process the proline CTI data. Second, construct each model and tune its parameters. In this phase, a standard SVM (a.k.a. Lib-SVM), a SVM variant, the proposed consensus models with Method I and II are constructed through a set of experiments that help to choose a proper kernel function and other parameters. Third, the predictive performance of the proposed methods is compared with those of SVM_{LIB} (Lib-SVM) and SVM_{ADA} (Adaboosted-Lib-SVM) for Q2 accuracy, Sensitivity (Sn), Specificity (Sp), Mathew's correlation coefficient (Mcc), Type I and II Error Rates, and StDev for Model stability/generalization ability on the proline CTI dataset built in the first phase. Lastly, we then compare those results with the consensus results from literature.

2.1 Evolutionary Dataset Construction

To make a fair comparison with existing proline CTI prediction models, we have chosen Song et al's [12] dataset. The dataset has 2424 non-homologous protein chains, obtained from the Culled PDB list provided by PSICES server [23]. All the tertiary structures in the dataset were determined by X-ray crystallography method with resolution better than 2.0Å and R-factor less than 0.25. In addition, the sequence identity of each pair of sequences is less than 25%, and the protein chains with sequence length shorter than 60 amino acids were excluded in the dataset. In total, there are 609,182 residues, and every sequence contains at least one *proline* residue. The PDB codes, CisPep PDB codes, proline *cis* peptide records, corresponding *dihedral* angles and protein sequences of the 2424 protein chains used in this study are available on request.

In addition, evolutionary information in the form of PSSMs was included in the windows as direct input. Evolutionary information in form of PSSMs is the most widely used input form for protein structure prediction in 1D, 2D and 3D, as well as other computational/structural proteomic prediction/classification tasks [14–21]. The idea of using evolutionary information in the form of PSSMs was first proposed by Jones et al [24], and it has improved its prediction accuracy about 3–5% in their prediction tasks.

To generate PSSM scores, we used the *nr* (non-redundant) database and *blastpgp* program obtained from NCBI [25]. We run *blastpgp* program to query each protein in our dataset against the *nr* database to generate the PSSMs with the following setup: 1) three iterations, 2) cutoff e-value of 0.001. Finally, the PSSM scores were scaled to the range between 0–1 by the following standard logistic function:

$$f(x) = \frac{1}{1+\exp(-x)},$$

where x is the raw profile matrix value. The scaled PSSM scores were used as direct input to the learning models. A PSSM is generated for each protein sequence, and has a $M\times 20$ matrix, where M is the target sequence length, and 20 is the number of amino acid types. Each element of the matrix represents the log-odds score of each amino acid at one position in the multiple alignments. The window size $2l+1$ indicates the scope of the vicinity of the target prolyl peptide bonds, determining how much neighboring sequence information is included in the prediction. We selected the windows size (l) of 9, and built our models as it produced the best predictive results, aligned with Song et al's experimental result.

When a large difference between positive and negative samples is observed in training set, data imbalance problem exists [26]. Our dataset is composed of 1,265 *cis* and 27,196 *trans* residues. There are two general approaches to reduce such imbalance problem. First, increasing the number of under-samples by random resampling. Second, decreasing the number of over-samples by random removal. In this study, we adopted the first approach, and made 1 to 1 ratios between the sizes of positive (*cis*) and negative (*trans*) training samples.

2.2 Protein Secondary Structure Information

The recent computational proteomic studies report that protein secondary structure information is useful in various protein sequence-based classifications/predictions [14–21]. Although the mutations at sequence level can obscure the similarity between homologs, the secondary-structure patterns of the sequence remain conserved. That is because changes at the structural level are less tolerated. The recent studies mostly use the probability matrix of secondary structure states predicted from PSI-PRED [14]. PSI-PRED is a well-known computational predictor, and it predicts protein secondary structures in three different states (α-helix, β-sheet, and loop). However, there is one significant limitation with using predicted secondary structure information. The best secondary-structure prediction model still cannot reach the upper boundary of its prediction accuracy. In other words, it is not good enough yet to be used as a cofirmation tool. It shows about 75–80% Q3 accuracies only. Clearly, incorrectly predicted secondary structure information if presented in input dataset of a computational prediction/classification model leads to the poor learning and, eventually to the incorrect prediction of proline CTI residues. Although predicted secondary information may be useful in some extent, it should not be used if one attempts to reach better than 80% Q2 accuracy. We therefore, used evolutionary information in the form of PSSMs obtained from protein amino-acid sequences only. To achieve above 80% Q2 accuracy, we believe that accurate and correct information encoding presented in input dataset is critical, especially if used with intelligent/model-free modeling like machine-learning. In other words, noise presented in input dataset could lead to significant degrading in the performance of the models.

2.3 Method I: Intelligent Voting

This new intelligent voting/consensus approach to RandomForest data modeling combines a number of methods and procedures to exploit homology information effectively. If we take a large collection of weak learners, each performing only better than chance, then by putting them together, it is possible to make an ensemble learner that can perform arbitrarily well. Randomness is introduced by bootstrap resampling [27] to grow each tree in the ensemble learner, and also by finding the best splitter at each node within a randomly selected subset of inputs. Method I grows many decision trees (DTs) [28] as in Figure 1. To classify a new input vector *x*, put the input vector down each of the DTs in the ensemble learner. Each DT is trained on a bootstrap sample of the training data.

To estimate the performance of the ensemble learner, Method I performs a kind of cross-validation by using Out-of-Bag (OOB) data. Since each DT in the ensemble grows on a bootstrap sample of the data, the sequences left out of the bootstrap sample, the OOB data, can be used as legitimate test set for that tree. On average $1–e^{-1} ≅ 1/3$ of the training data will be OOB for a given tree. Consequently, each PSSM in the training dataset will be left out of 1/3 of the trees in the ensemble, and use these OOB predictions to estimate the error rate of the full ensemble.

Like CART [29], Method I uses the *gini* index for determining the final class in each DT. The gini index of node impurity is the measure most commonly chosen for classification-type problems. If a dataset *T* contains examples from *n* classes, gini index *G(T)* is defined as:

$$G(T) = 1 - \sum_{j=1}^{n} (P_j)^2,$$

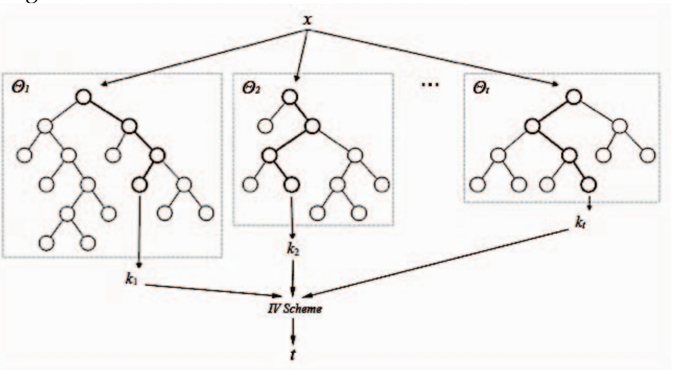
where p_j is the relative frequency of class *j* in *T*. If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *G(T)* is defined as:

$$G_{split}(T) = \frac{N_1}{N} G(T_1) + \frac{N_2}{N} G(T_2).$$

The attribute value that provides the smallest $G_{split}(T)$ is chosen to split the node.

Figure 2 shows the key three steps of the Method I Ensemble. First, a random seed is chosen which pulls out at random a collection of samples from the training dataset while maintaining the class distribution. Second, with this selected dataset, a random set of attributes from the original dataset is chosen based on user defined values. All the input variables are not considered because of enormous computation and high chances of overfitting. In a dataset where M is the total number of input attributes in the dataset, only R attributes are chosen at random for each tree where R<M.

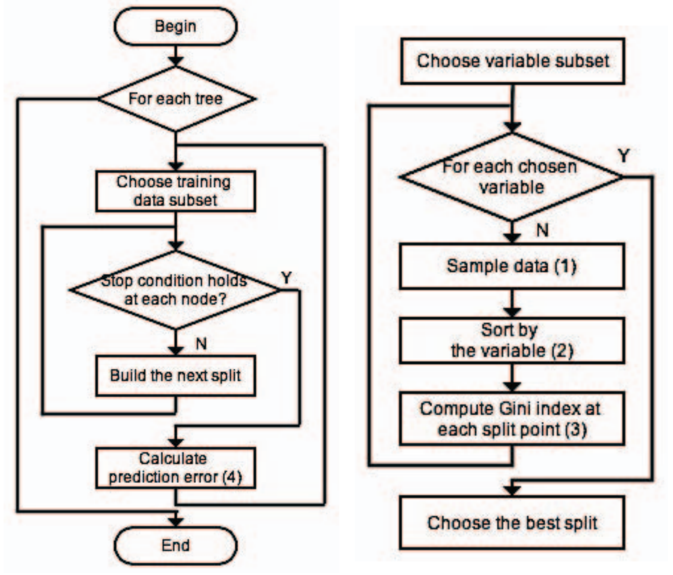
Figure 1. A General Architecture of Method I Ensemble



The collection of decision trees (DTs) *{h(x, Θ$_k$), k = 1...}*, where the $Θ_k$ are independently, identically distributed random DTs, and each DT casts "a unit vote" for the final classification of input *x*.

Third, the attributes from this set create the best possible split using the gini index to develop a DT model. The process repeats for each of the branches until the termination condition stating that leaves are the nodes that are too small to split. In this study, Method I Ensemble was constructed and implemented with the *Weka* RF package [30].

Figure 2. A Flowchart of Method I Ensemble



The left-hand side shows the main flow of the Method I Ensemble while the right-hand side flowchart is the expantion of the process, Build The Next Split, of the main flowchart of left-hand side.

2.4 Method II: Randomized Metalearning

Method II builds an ensemble of randomized base classifiers (i.e., Method I), and averages their classification. Each one is based on the same input data, but uses a different ramdom-number seed. Some leanring algorithms already have a built-in random component. For example, when learning multiplayer perceptrons using the backpropagation algorithm, the initial network weights are set to small randomly chosen values. The learned classifier depends on the random numbers because the algorithm may find a different local minimum of the error function. One way to make the outcome of classification more stable is to run the learner several times with different random number seeds (i.e., initial weights) and combine the classifiers' predictions by voting or averaging. Learning in Method I builds a randomized DT in each iteration of the bagging algorithm, and often produces excellent predictors. Although bagging [27] and randomization yield similar results, it sometimes pays to combine them because they introduce randomness in different, perhaps complementary, ways. Randomization demands more work than bagging because the learning algorithm must be modified, but it can profitably be applied to a greater variety of learners.

The Method II input vector is formed by three seeds, and ten number of iterations. The steps involved in decision-making are depicted in Figure 2. Tables 1 and 2 show their experimental results related to Methods I and II predictions. Using Method II to combine all the DTs, the system reduced Type I error rate significantly as depicted in Table 2.

2.5 Model Validation and Testing

For the system model to be useful, it must be validated to ensure that it emulates the actual system in the desired manner. This is especially true for empirical models, such as statistical machine-learning models, which primarily rely on observed data rather than analytical equations derived from first principles. The validation of these models using problem-specific information, such as theoretic relationships or experimental knowledge, should be performed. There are several methods to perform the validation task. The most common statistical methods are resubstitution, cross-validation, bootstrapping, and their variants.

To accurately assess the predictive performance of each model, we adopted a cross-validation scheme for our model evaluation. First, we apply the holdout method to our proline CTI dataset. However, the holdout method has a key drawback in that the single random division of a sample into training and testing sets may introduce bias in model selection and evaluation. Since the estimated classification rate can be very different depending on the characteristic of the data, the holdout estimate can be misleading if we happen to get an unfortunate split. Hence, in our experiment, we adopted multiple train-and-test experiments to overcome the limitation of the holdout method. We created 7 to 11-fold dataset, and only one of each fold was used for testing. The result of each fold is provided in Tables 1 and 2.

2.6 Parameter Tuning

All of the stages contain parameters or variables that need to be given appropriate values. Some of these parameters are so delicate that they have to be selected by an expert in the field, and kept constant thereafter. However, profoundly more interesting are the parameters the system is able to learn autonomously from training with available data. In our experiments, we used a semi-autonomous approach. We first used the Weka's meta-learner, *CVParameterSelection* searches, and again checked the neighboring values of the best parameters found by the search. The list of the full parameters that we have used in our experiments is provided with Table 1.

3 Model Evaluation and Analysis

The performance of the models used in this study are measured by the accuracy (Q2: the proportion of true-positive and true-negative residues with respect to the total positives and negatives residues), the sensitivity (Sn: also called *recall*, the proportion of correctly predicted isomerization residues with respect to the total positively identified residues), the specificity (Sp: also called *precision*, the proportion of incorrectly predicted isomerization residues with respect to the total number of proline isomerization residues), and Mathew's correlation coefficient (Mcc: a correlation coefficient between the observed and predicted binary classifications, between −1 and +1). In Mcc, a coefficient of +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation. Hence, a high value of Mcc means that the model is regarded as a more robust prediction model. The above measures can be obtained using the following.

$$Q2 = \frac{TP+TN}{TP+TN+FP+FN},$$

$$Sp = \frac{TN}{TN+FP},$$

$$Sn = \frac{TP}{TP+FN},$$

$$Mcc = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}},$$

where TP is the number of true postives, FN is the number of false negatives or under-predictions, TN is the number of true negatives, and FP is the number of false positives or over-predictions. We adopted the polynomial kernel function and radial basis function (rbf kernel) to construct the SVM classifiers, which is aligned with the existing proline CTI prediction studies [12].

$$K(\vec{x}_i \cdot \vec{x}_j + 1)^d,$$

$$K(\vec{x}_i \cdot \vec{x}_j) = \exp(-r\|\vec{x}_i - \vec{x}_j\|^2),$$

where the degree d needs to be tuned as for polynomial function, and the gamma and the regulator parameters for RBF need to be regulated. See the footnote of Table 1 for the parameters settings used for this study. For the optimal learning of the prediction models, the most suitable data fold for each model should be sought.

Table 1 shows the comparison of our proposed methods results with those of SVM_{LIB} and its variant, SVM_{AB}. The best score in each category is underlined, and the best fold scores in each model are bolded. As seen from the table,

Table 1. Model Performance Comparisons in Different Fold

Models	Fold	Ac	Sp	Sn	Mcc
SVM_{LIB}	7	0.7613	0.5604	0.9621	0.5712
	8	0.7633	0.5623	0.9645	0.5757
	9	**0.7672**	**0.5722**	**0.9622**	**0.5813**
	10	0.7606	0.5584	0.9628	0.5703
SVM_{AB}	7	0.7645	0.5647	0.9643	0.5775
	8	0.7647	0.5644	0.9650	0.5781
	9	**0.7653**	**0.5656**	**0.9650**	**0.5796**
	10	0.7564	0.5483	0.9645	0.5639
Method I	7	0.8032	0.6510	0.9554	0.6379
	8	0.8080	0.6644	0.9514	0.6440
	9	0.8077	0.6611	0.9544	0.6461
	10	**0.8150**	**0.6698**	**0.9599**	**0.6592**
	11	0.8107	0.6680	0.9533	0.6504
Method II	7	0.8452	0.7399	0.9504	0.7076
	8	0.8575	0.7643	0.9506	0.7289
	9	**0.8658**	**0.7816**	**0.9500**	**0.7443**
	10	0.8589	0.7688	0.9489	0.7311

The parameters of each model were given the following values: **Method I** (debug: false, maxDepth: 0, numExecutionSlots: 1, numFeatures: 0, numTrees: 13, printTrees: false, seed: 1), **Method II** (the same setting for Method I, and for Method II, seed: 3 and iteration: 10), **SVM_{LIB}** (SVMType: C-SVM, cacheSize: 40.0, coef0: 0.0, cost: 13, debug: false, degree: 3, eps: 0.0010, gamma: 0.0, kernelType: rbf, loss: 0.1, normalize: false, nu: 0.5, seed: 1, shrinking: true), and **SVM_{AB}** (the same as SVM_{LIB}'s, and for Adaboost, numIterations: 14, seed: 1, weightThreshold: 100)

SVM models and Method II perform better on 9-fold while Method I performs better on 10-fold. The proposed methods (Methods I and II) performed a far better then SVM models. Method I and II achieved 81.5%, and 86.58% Q2 accuracies respectively, while SVM models achieved about 76% Q2 accuracy only.

Table 1 also shows that our proposed methods are superior to SVM models in terms of Sp and Mcc, which indicate the model robustness and stableness. The Q2 accuracy of 86.58% that we achieved on proline CTI prediction is a far better than those of any existing computational proline CTI predictors reported in the literature. The best Q2 accuracy that we have found in the literature was about 73% on the same dataset as used in this research.

The performance of each model is measured by Types I and II Error rates as well, since incorrectly predicted residues can be as valuable, as are the correctly predicted residues for further modification of the model. Type I Errors mean experimentally verified trans residues that are predicted (incorrectly) to be cis residues; type II errors indicate experimentally verified cis residues that are predicted (incorrectly) to be trans residues. Method II shows the lowest Type I Error Rate (0.21833) while SVM_{AB} reaches the lowest Type II Error Rate (0.035). Although our proposed methods seem to be not very useful in improving Type II Error Rate, it reduces Type I Error Rate effectively. Interestingly, Type I Error Rates are worse with SVMs while Type II Error Rates are worse with our proposed methods.

Table 2. Type I and II Errors in Different Fold

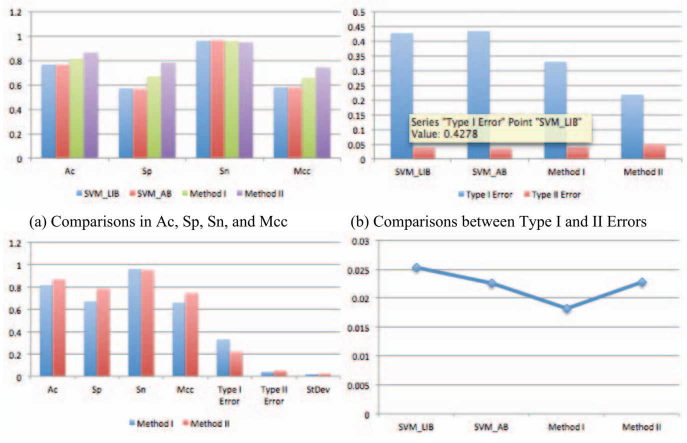
Models	Fold	Type I	Type II	StDev
SVM_{LIB}	7	0.4396	0.0379	0.0268
	8	0.4378	0.0355	0.0229
	9	**0.4278**	**0.0378**	**0.0253**
	10	0.4416	0.0372	0.0318
SVM_{AB}	7	0.4353	0.0357	0.0219
	8	0.4356	0.035	0.0233
	9	**0.4344**	**0.035**	**0.0226**
	10	0.4517	0.0355	0.0352
Method I	7	0.349	0.0446	0.0105
	8	0.3356	0.0486	0.014
	9	0.3389	0.0456	0.012
	10	**0.3302**	**0.0401**	**0.0182**
	11	0.332	0.0467	0.0306
Method II	7	0.2601	0.0496	0.0114
	8	0.23575	0.049375	0.01523
	9	**0.21833**	**0.05**	**0.022672**
	10	0.2312	0.0511	0.026547

Type I Errors mean experimentally verified trans residues that are predicted (incorrectly) to be cis residues; type II errors indicate experimentally verified cis residues that are predicted (incorrectly) to be trans residues.

StDev provides a good idea on model generalization ability. Although nonparametric machine-learning models have been proved to be useful in many different applications, their generalization capacity has often been shown to be unreliable because of the potential for overfitting. The symptom of overfitting is that the model fits the training sample too well, and thus the model output becomes unstable for prediction. On the other hand, a more stable model, such as a linear model, may not learn enough about the underlying relationship, resulting in underfitting the data. It is clear that both underfitting and overfitting will affect the generalization capacity of a model. The underfitting and overfitting problems in many data-modeling procedures can be analysed through the well-known bias-plus-variance decomposition of the prediction error. The best generalization ability came from Method I, where a multiple DTs have been built.

The idea used in Method II using a different ramdom-number seed seems to be useful in better learning; however, not enough to improve its generalization ability. Method I shows the best StDev value of 0.0182, while other models reach about 0.022–0.025. (a) of Figure 3 depicts the performance comparisons of four different models in Q2, Sp, Sn, and Mcc. As you can see, Method II outperforms other models in Q2, Sp, and Mcc, and no significant differences observed in Sn, while Method I and II improve Sp and Mcc significantly. As in (b) of Figure 3, Type II Error Rates are much lower than Type I Error Rates in general, and Method I and II effectively reduce Type I Error Rate. (c) of Figure 1, shows that Method II improves in Q2, Sp, and Mcc and Type I Error Rate. However, no significant improvement observed in Sn and Type II Error Rate and StDev. This result indicates that our proposed methods could be improved by reducing the errors of experimentally verified cis residues that are predicted (incorrectly) to be trans residues.

Figure 3. Model Performance Comparisons



(a) Comparisons in Ac, Sp, Sn, and Mcc (b) Comparisons between Type I and II Errors

(c) Comparisons between Method I and II (d) Comparisons in StDev

(d) of Figure 3 compares the model generalization ability and stability. Method I clearly outperforms other models. Again, using a different random-number seed does not really make the outcome of classification more stable, which contradicts to the findings in [31].

Although our proposed methods have shown to be useful for proline CTI prediction tasks, we suggest the following to be taken into account for the sake of improvement. First, since our methods use PSSMs only as input, homology information presented in PSSMs may not have enough information to reach the upper-boundary accuracy. Recent studies suggest that global sequence homology is seen as a strong indicator for the occurrence of prolyl cis residues [32], meaning that accurate descriptors of CTI residues and their corresponding encoding schemes must be identified. Second, solvent accessibility as a new possible input feature of proline CTI must be well examines as proline *cis* residues are more frequently found in surface accessible areas compared to *trans* residues [32]. Computational machine-learning approaches build their models based on input data only. Clearly, missing useful information in input dataset leads to misclassification.

4 Conclusion

In this paper, we have presented a novel ensemble method to predict proline CTI residues in proteins. The proposed models are trained using a RF-like ensemble method, which grows a multiple trees, and chooses the classification having the most votes over all the trees in the forest, and build an ensemble of randomized base classifiers using using a different ramdom-number seed, and averages their classification. On average, our methods are able to predict proline CTI with the Q2 accuracy of 86.58%, a far better than any existing proline CTI predictors reported in the literature. Experimental results on proline CTI prediction could be subjective as other existing prediction methods usually do predictions on their own dataset. However, our experiments were able to demonstrate that the proposed methods can achieve a test error better than the most widely used computational models, SVM and its variants, in the literature on the same dataset used in [12]. It has also demonstrated that pure evolutionary information in the format of PSSM scores as input works greatly in reducing error rate during the model learning process, meaning that noise presented (i.e., predicted secondary information) in input dataset may lead to significant degrading in the performance of the models.

Acknowledgment

We are grateful to Dr. Song at the ARC Centre of Excellence in Structural and Functional Microbial Genomics at Monash University, Australia for providing us the proline cis-trans isomerization dataset, as well as her invaluable discussions.

References

[1] U. Reimer, G. Scherer, M. Drewello, S. Kruber, M. Schutkowski, G. Fischer, Side-chain effects on peptidyl-prolyl cis/trans isomerization, J. Molecular Biology, 279, pp. 449-460, 2003

[2] B. Eckert, A. Martin, Balbach J, F.X. Schmid, Prolyl isomerization as a molecular timer in phage infection, Nat Struct Mol Biol. 12, pp. 619–623, 2005

[3] W.J. Wedemeyer, E. Welker, H.A. Scheraga, Proline cis-trans isomerization and protein folding, Biochemistry, 41, pp. 14637–14644, 2002

[4] F.X. Schmid, Prolyl isomerases, Adv. Protein Chem. 59, pp. 243–282, 2001

[5] J. Balbach, F.X. Schmid, In Mechanisms of Protein Folding (ed. Pain, R.H.), pp. 212–237, Oxford Univ. Press, Oxford, UK, 2000

[6] M.B. Yaffe et al., Sequence-specific and phosphorylation-dependent proline isomerization—a potential mitotic regulatory mechanism, Science, 278, pp. 1957–1960, 1997

[7] G. Fischer, T. Aumuller, Regulation of peptide bond cis/trans isomerization by enzyme catalysis and its implication in physiological processes, Rev. Physiol. Biochem. Pharmacol., 148, pp. 105–150, 2004

[8] Y. Wu, R. Matthews, A Cis-Prolyl Peptide Bond Isomerization Dominates the Folding of the Alpha Subunit of Trp Synthase, a TIM Barrel Protein, J. Mol. Biol., 322, pp. 7–13, 2002

[9] F.X. Schmid, Prolyl isomerase: enzymatic catalysis of slow protein-folding reactions, Annu. Rev. Bio- phys. Biomed. 22, pp. 123–142, 1993

[10] C. Frömmel, R. Preissner, Prediction of prolyl residues in cis-con formation in protein structures on the basis of the amino acid sequence, FEBS Lett, 277, pp. 159–163, 1990

[11] M.L. Wang, W.J. Li, W.B. Xu, Support vector machines for predic tion of peptidyl prolyl cis/trans isomerization, J Peptide Res. 63, pp. 23–28, 2004

[12] J. Song et al., Prediction of cis/trans isomerization in proteins using PSI-BLAST profiles and secondary structure information, BMC Bioinformatics, 7(124), 2006

[13] S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman, Basic local alignment search tool, Journal of Molecular Biology 215(3), pp. 403–410, 1990

[14] T.N. Petersen, C. Lundegaard, M. Nielsen, H. Bohr, J. Bohr, S. Brunak, G.P. Gippert, O. Lund, Prediction of Protein Secondary Structure at 80% Accuracy, PROTEINS: Structure, Function, and Genetics, 14, pp.17–20, 2000

[15] D. Pahlke, D. Leitner, U. Wiedemann, D. Labudde, COPS-cis/trans peptide bond conformation prediction of amino acids on the basis of secondary structure information, Bioinformatics, 21, pp. 685–686, 2005

[16] K. Exarchos et al., Prediction of cis/trans isomerization using feature selection and support vector machines, J. Biomedical Informatics, 42, pp.140–149, 2009

[17] P.D. Yoo, B.B. Zhou, A.Y. Zomaya, Machine Learning Techniques for Protein Secondary Structure Prediction: An Overview and Evaluation." Journal of Current Bioinformatics, 3(2), pp. 74–86, 2008

[18] P.D. Yoo, A. Sikder, J. Taheri, B.B. Zhou, A.Y. Zomaya, DomNet: Protein Domain Boundary Prediction Using Enhanced General Regression Network and New Profiles, IEEE Transactions on NanoBioscience, 7(2), pp.172–181, 2008

[19] P.D. Yoo, S. Ho, B.B. Zhou, A.Y. Zomaya, SiteSeek: Protein Post-Translational Modification Analysis Using Adaptive Locality-Effective Kernel Methods and New Profiles, BMC Bioinformatics, 9, 272, 2008

[20] P.D. Yoo, B.B. Zhou, A.Y. Zomaya, A modular kernel approach for integrative analysis of protein inter-domain linker regions, BMC Genomics BMC Genomics, 10(3): S21, 2009

[21] P.D. Yoo, Y.S. Ho, J. Ng., et al. Hierarchical Kernel Mixture Models for the prediction of AIDS disease progression using HIV structural gp120 profile, Journal of BMC Genomics, 11(4), S22, 2010

[22] L. Breiman, Random Forests, Machine Learning, 45, pp. 5–32, 2001

[23] PISCES: a protein sequence culling server, Available at [http://dunbrack.fccc.edu/PISCES.php]

[24] D.T. Jones, Protein secondary structure prediction based on position-specific scoring matrices, J Mol Biol, 292, pp.195–202, 1999

[25] NCBI FTP website [ftp://ftp.ncbi.nlm.nih.gov/blast/db/]

[26] Qian J, Lin J, Luscombe NM, Yu H, Gerstein M: Prediction of regulatory networks: genome-wide identification of transcription factor targets from gene expression data. Bioinformatics 2003, 19:1917-1926

[27] L. Breiman, Bagging predictors, Machine Learning 24(2), pp. 123–140, 1996

[28] P. Argentiero , R. Chin and P. Beaudet, An automated approach to the design of decision tree classifiers, IEEE Trans. Patt. Analy. Mach. Intell., vol. PAMI-4, pp.51–57, 1982

[29] X.D. Wu, V. Kumar, The top ten algorithm in data mining, Chapman & Hall, CRC, London, 2009

[30] G. Holmes, A. Donkin, I.H. Witten, Weka: A machine learning workbench, Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, 1994

[31] M.M.S. Lira, R.R.B. de Aquino, A.A. Ferreira, M.A. Carvalho, O.N. Neto, G.S.M. Santos, Combining Multiple Artificial Neural Networks Using Random Committee to Decide Upon Electrical Disturbance Classification, International Joint Conference on Neural Networks (IJCNN), pp. 2863–868, Aug. 12–17, 2007

[32] S. Lorenzen, B. Peters, A. Goede, R. Preissner, C. Frömmel, Conservation of cis prolyl bonds in proteins during evolution, Proteins, 58, pp. 589–595, 2005

Paul D. Yoo received his PhD in Engineering and IT from the University of Sydney (USyd) in 2008. He was a Research Fellow in the Centre for Distributed and High Performance Computing, at USyd from 2008 to 2009, and PHD Researcher (Quantitative Analysis) at the Capital Markets CRC, administered by the Australia Federal Dept., for Education, Science and Training, from 2004 to 2008. He is currently an Assistant Professor of Intelligent Systems and Data Mining in ECE dept., Khalifa University, Abu Dhabi. His recent research is centred on the application of various computer science and mathematical methods to the discovery of the syntactic and semantic patterns in nucleic acid and amino acid sequences. Paul holds over 30 prestigious journal and conference publications and is currently actively involved in technical program committees, and review panels of the intelligent systems/machine-learning areas for international conference and journal publications such as IEEE, ACM and ISCB.

Sami Muhaidat received the Ph.D. degree in Electrical and Computer Engineering from the University of Waterloo, Ontario, in 2006. From 2007 to 2008, he was an NSERC postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. From 2008 to 2012, he was an Assistant Professor in the School of Engineering Science, Simon Fraser University, BC, Canada. He is currently an Assistant Professor at Khalifa University and a Visiting Reader in the Faculty of Engineering, University of Surrey, UK. Sami's research focuses on advanced digital signal processing techniques for image processing and communications, machine learning, cooperative communications, vehicular communications, MIMO, and space-time coding. He has authored more than 100 journal and conference papers on these topics. Sami is an active Senior IEEE member and currently serves as an Editor for IEEE Communications Letters and an Associate Editor for IEEE Transactions on Vehicular Technology. He was the recipient of several scholarships during his undergraduate and graduate studies. He was also a winner of the 2006 NSERC Postdoctoral Fellowship competition.

Kamal Taha is an assistant professor in ECE Dept., at Khalifa University, Abu Dhabi, since 2010. He received his Ph.D. in Computer Science from the University of Texas at Arlington, USA, in March 2010. He has over 30 refereed publications that have appeared in journals, conference proceedings, and book chapters. He worked as an Instructor of Computer Science at the University of Texas at Arlington from August 2008 to August 2010. He worked as Engineering Specialist for Seagate Technology (a leading computer disc drive manufacturer in the US) from 1996 to 2005. His current research interests include bioinformatics databases (mediators, ontologies), information retrieval in semi-structured data, keyword search in XML documents, recommendation systems and social networks, and knowledge discovery and data mining. He serves as a member of the Program Committee of a number of international conferences and he is a reviewer for a number of academic journals and conferences. He was selected by Marquis Who's Who to be included in the 2011–2012 (11th) Edition of Who's Who in Science and Engineering. He is a member of the IEEE.

Jamal Bentahar received his M.Sc. in 2001 from National Engineering School of Computer Science and Systems Analysis, Morocco and Ph.D. in 2005 from Laval University, Canada in Computer Science and Software Engineering. He is currently Associate Professor in the Concordia Institute for Information Systems Engineering at Concordia University, and is member of the Professional Engineers of Ontario. Before joining Concordia University in July 2006 as Assistant Professor, he was NSERC Postdoctoral Fellow at Simon Fraser University (2005). He is Associate Editor of the Journal of Service Oriented Computing and Applications, Springer and serves on the Editorial Board of the International Journal of Information Technology and Web Engineering, IGI Global. His research interests include intelligent systems, multi-agent systems, web services, and model checking.

Abdallah Shami (M03, SM09) received a B.E. Degree in Electrical and Computer Engineering from the Lebanese University, Beirut, Lebanon in 1997, and a Ph.D. Degree in Electrical Engineering from the Graduate School and University Center, City University of New York, New York, NY in September 2002. In September 2002, he joined the Department of Electrical Engineering at Lakehead University, Thunder Bay, ON, Canada as an Assistant Professor. Since July 2004, he has been with The University of Western Ontario, London, ON, Canada where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. His current research interests are in the areas of wireless/optical networking, big data, and biomedical informatics.