DNpro: A Deep Learning Network Approach to Predicting Protein Stability Changes Induced by Single-Site Mutations

Xiao Zhou and Jianlin Cheng*

Computer Science Department
University of Missouri
Columbia, MO 65211, USA
*Corresponding author: chengji@missouri.edu

Abstract—A single amino acid mutation can have a significant impact on the stability of protein structure. Thus, the prediction of protein stability change induced by single site mutations is critical and useful for studying protein function and structure. Here, we presented a new deep learning network with the dropout technique for predicting protein stability changes upon single amino acid substitution. While using only protein sequence as input, the overall prediction accuracy of the method on a standard benchmark is >85%, which is higher than existing sequence-based methods and is comparable to the methods that use not only protein sequence but also tertiary structure, pH value and temperature. The results demonstrate that deep learning is a promising technique for protein stability prediction. The good performance of this sequence-based method makes it a valuable tool for predicting the impact of mutations on most proteins whose experimental structures are not available. Both the downloadable software package and the user-friendly web server (DNpro) that implement the method for predicting protein stability changes induced by amino acid mutations are freely available for the community to use.

Keywords—Bioinformatics, deep learning, protein stability prediction, biological data mining.

INTRODUCTION

Single site amino acid mutations may have a significant impact on the stability of the structure of a protein. Since experimental determination of the stability change caused by mutations on proteins is time consuming and costly, computational prediction of the stability change induced by single-site mutations is useful for screening a large number of mutations for studying the structure and function of proteins. It can be used for protein engineering, protein design, mutagenesis analysis, and the study of the relationship between phenotypes and genotypes.

Recently, a variety of methods based on physical potentials [6, 7, 8, 9, 10], statistical potentials [11, 12, 13, 14, 15, 32, 36, 37], empirical potentials [16, 17, 18, 19, 20], machine learning [1, 2, 21, 25, 26] and combined approaches [31, 33, 34] have been developed to predict protein structure stability upon single-site amino acid mutation. The physical potential approaches, statistical potential approaches, and empirical potential approaches aim to approximate either the physical or pseudo-physical energy based on physical principles or their statistical approximation. Machine learning methods are data-driven knowledge-based methods that learn a function from the data to map the input information regarding a protein and its mutation to the energy change without the need of approximating the physics underlying mutation stability. This data-driven formulation of the problem makes it possible to apply a large array of machine learning methods to tackle the problem. Therefore, a wide range of machine learning methods has been applied to the same problem. E Capriotti et al., 2004 [2] presented a neural network for predicting protein thermodynamic stability with respect to native structure; J Cheng et al., 2006 [1] developed MUpro, a support vector machine with radial basis kernel to improve mutation stability prediction; iPTREE based on C4.5 decision tree revealed temperature is an important factor in mutation stability prediction [35]; R Casadio et al., 1995 [25] proved a radial basis function neural network can predict the free energy changes induced by mutations; M Masso et al., 2008 [31] combined random forest, regression tree and ada-boost on C4.5 decision tree with statistical potential approach to predict protein stability change. This array of machine learning tools with reasonable prediction accuracy are widely used by the community, which demonstrates the effectiveness of machine learning methods for protein stability prediction.

In addition to exploring various methods to further improve the accuracy of mutation stability prediction, reducing the amount of information required to make accurate prediction is also important for making the prediction methods generally applicable to any proteins that may not have the information required by existing methods. Many previous machine learning methods [2, 21, 25, 26, 31, 35] need sequence, solvent accessibility, pH value, temperature or tertiary structure information to make good predictions, which limits their use only to a small portion of proteins whose such information are available. J Cheng et al., 2006 [1] introduced a method encoding only sequence and mutation residue information, whose accuracy is comparable to existing methods. Following this direction, we design a new method based on cutting-edge deep learning networks to further improve protein stability changes upon single-site mutations using only protein sequence information.

Deep learning [22, 23, 24, 28] is the latest development in the field of machine learning, which has shown good performance in many complex machine learning tasks such as image processing, speech recognition, and protein structure prediction. Like traditional neural networks, deep learning

networks [22, 23, 24, 28] consists of multi-layer of nodes and the nodes between adjacent layers are fully connected. However, different from discriminative back-propagation supervised learning in neural networks, the weights between adjacent layers of deep learning networks can be learned mostly by un-supervised stochastic gradient descent or divergence-convergence algorithms in order to maximize the likelihood of input data. Therefore, deep learning networks can be trained on the almost unlimited amount of unlabeled data even though the labeled data can still be used to tune the weights of deep learning networks via back-propagation when available. This fundamental new training scheme helps avoid the vanishing gradient problem in traditional back-propagation and enables deep learning to use a large number of layers of nodes (i.e. deep networks) for complex learning tasks in contrast to typical two-layer or three-layer architecture used by traditional neural networks. And the deep architecture often makes deep learning capable of gradually transforming low-level input features into higher-level concepts from layer to layer, which often leads to better prediction performance.

As for this protein stability prediction task, a special kind of deep learning architecture - Deep Belief Network (DBN) [5] is developed, which stacks a number of layers of Restricted Boltzmann Machines [5] to predict protein stability changes upon mutations. A logistic regression [29] is added on the top layer of DBN, which allows the weights in the DNB that have been pre-trained by the unsupervised convergence-divergence algorithm to be fine-tuned by the supervised back-propagation method [22, 23]. The Dropout technique [3, 4, 27] is also applied in DBN to make the hidden layers more diverse to avoid overfitting. The deep learning model is tested in a 20-fold cross-validation on a standard protein stability benchmark. The results demonstrate this new deep learning method outperforms some widely used methods in protein stability prediction on a standard benchmark.

METHOD

Protein Stability Dataset

We use the S1615 dataset [1,2] in this work, which is widely used in the field of protein stability prediction [1, 2, 31, 35]. The mutations in the dataset were originally extracted from ProTherm [30] mutation energy database. The dataset has 1615 mutations obtained from 42 different proteins. PDB code of the mutated protein, secondary structure, solvent accessibility, original and mutated residues, temperature, pH and energy change are available in the dataset.

The energy change associated with each mutation in the dataset represents the change of protein stability caused by a single-site mutation. A positive energy change indicates a protein is more stable upon a mutation while a negative energy change indicates a decrease in structural stability. In a binary classification case, mutations with positive energy changes are labeled positively as well as all the others are labeled negatively. Dataset S1615 thus has 1168 negative data points and 447 positive examples, i.e. about 28% mutations increase the stability of the proteins.

Besides S1615, a redundancy-reduced subset of S1615 dataset that contains 388 unique mutations is also used to cross validate our model and optimize parameters. S388 has 340 negative examples and 48 positive ones. We carry out a 20-fold cross validation as in the previous work [1, 2, 31] to determine the good parameter values for the deep learning network. To avoid overfitting the S388 dataset during training, an independent dataset S65 is selected randomly from the S1615 dataset excluding the mutations in S388 dataset in order to blindly test the deep learning model trained on the S388 dataset. S65 has 52 negative examples and 13 positive examples. S65 is not used in any way until the final parameters and architecture of the deep learning model are determined by 20-fold cross validation on the S388 dataset

Encoding Scheme of Input Features

In order to make our method applicable to proteins without known tertiary structures, we only use the features that can be derived from protein sequence alone as input. Following the same feature extraction approach in [1], a window centered on the mutation site is used to capture the information regarding the mutation and its adjacent sequence environment. Each position in the window except the mutation site is represented by 20 binary numbers that represent 20 possible amino acids at the position. Only the number representing the amino acid at the position is set to 1, while all others are set to 0. For the mutation site in the window, the number representing the original residue is set to -1 and the number denoting the substitute residue is set to 1, while all other 18 numbers are set to 0. The size of the window is fixed at 7 because the previous work [1] demonstrated that the size yielded the best performance. As a result, a vector of 140 input features is generated for each mutation, which captures the information of the mutated residue and its three residue neighbors on each side.

After the encoding, the datasets consisting of both input features and binary class labels (positive or negative) are used to train and test our deep learning method - the Deep Belief Network (DBN) [22, 23] – for classifying mutations via 20-fold cross-validation and independent testing.

Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN) for Protein Stability Prediction

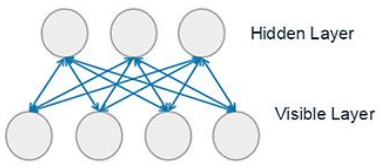

Figure 1. A Restricted Boltzmann Machine (RBM) model is comprised of a visible layer of nodes, a hidden layer of nodes, and the weighted connections between the nodes in two layers. The strength of each connection is quantified by an adjustable weight associated with it. As a simple example, in this model, there are 3 hidden units in the hidden layer and 4 visible units in the visible layer.

A Deep Belief Network (DBN) stacks several layers of Restricted Boltzmann Machines (RBMs) (**Figure 1**) [22] together to map input to output step by step. An RBM model is composed of one visible layer of nodes representing input features and one hidden layer of nodes representing hidden / latent variables. The nodes in the two layers are fully connected and the strength of connection between two nodes (v_i in the visible layer and h_j in the hidden layer) is measured by weights (ω_{ij}) of the connection between them. The probability of activating a visible or hidden unit is defined as:

$$p(v_i = 1|h) = \sigma\left(a_i + \sum_j h_j\, \omega_{ij}\right) \quad (1)$$

$$p\left(h_j = 1\middle|v\right) = \sigma\left(b_j + \sum_i v_i\, \omega_{ij}\right) \quad (2)$$

Here, v_i denotes the i^{th} unit in the visible layer and h_j the j^{th} unit in hidden layer. ω_{ij} is the weight of the connection between them , a_i and b_j are bias for the two nodes respectively. $\sigma(x)$ is a sigmoid function as follows:

$$\sigma(x) = 1/\left(1 + e^{(-x)}\right) \quad (3)$$

The energy of a joint configuration of all the nodes of an RBM is given by:

$$E(v,h) = -\sum_{i\in v} a_i\, v_i - \sum_{j\in h} b_j\, h_j - \sum_{i\in v, j\in h} v_i\, h_j \omega_{ij} \quad (4)$$

Thus, the probability of a joint configuration of the visible and hidden nodes is defined as:

$$p(v,h) = e^{-E(v,h)}/Z \quad (5)$$

Here Z is the normalization constant – the sum of the probabilities of all joint configurations:

$$Z = \sum_{v,h} e^{-E(v,h)} \quad (6)$$

The objective of training an RBM is to adjust its weights such that the probability of the visible layer $p(v)$ (i.e. the likelihood of the data) is maximized. $p(v)$ is simply calculated as the sum of the probabilities of all possible joint configurations over all hidden units:

$$p(v) = \frac{1}{Z}\sum_h e^{-E(v,h)} \quad (7)$$

One of the most popular un-supervised methods to train RBMs is Contrastive Divergence (CD) [5] algorithm that adjusts weight ω_{ij} by the difference between the current setting ($v_i h_j^{(t)}$) at time t and the next setting ($v_i h_j^{(t+1)}$) at time t+1.

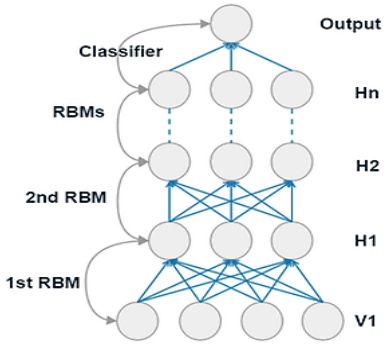

Figure 2. A Deep Belief Network (DBN) consisting of multiple RBMs with a logistic regression classifier on top of it. The output of each lower layer is used as the input for the next layer.

The same unsupervised CD algorithm can be used to pre-train all the layers of a DBN consisting of multiple RBNs step by step (see **Figure 2** for a DBN example). It iteratively trains every two adjacent layers as an RBM from bottom up while using the output of the lower layer as the visible input of the immediate upper layer. The input of the first layer is set as the original input data. Before pre-training, the weights and bias are initialized either randomly or to 0.

The unsupervised learning described above can map input features into high-level hidden features represented by the nodes in the higher layer. In order to make a DBN to classify data into different categories, a classifier needs to be added at the top of a DBN (**Figure 2**). In this work, a logistic regression is added at the top of DBN because it is easy to train. The weights between the logistic regression node and the hidden layers can be trained by the standard back-propagation algorithm - gradient descent with respect to the difference between the output of the DBN and the real labels of the data (i.e. classification error). Furthermore, the classification error is back propagated from the logistic regression node all the way down to the visible layer in order to fine tune all weights in the DBN in a supervised fashion. Although this hybrid training protocol of combining both unsupervised and supervised learning above works well in most situations, it still may be susceptible to the classic problem of overfitting data in machine learning. To reduce overfitting, we apply the dropout [3] training technique to train the deep learning network (**Figure 3**). During training, dropout randomly sets some units to 0 to temporarily drop them out of the deep network. When a unit is dropped out, all the connections associated with the unit are removed. Therefore, in each iteration of training, there is a different combination of active units. In addition to reducing overfitting, dropout also helps prevent units from co-adapting.

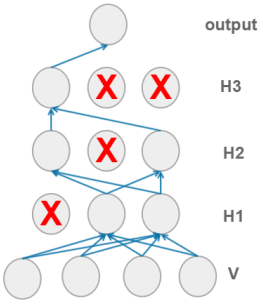

Figure 3. The dropout training protocol. Dropout stochastically sets some nodes (units) to 0 to disable them at a constant [3] or adaptive [4] rate in each epoch of training and fine-tuning. When a unit is dropped out, all the weights related to it will not be updated in this epoch.

The standard dropout approach sets the dropout rate to 0.5, which has been proved optimal for a wide range of networks and tasks. In this work, the dropout rate is applied to all layers except the input layer. In addition to the standard dropout approach, we test the adaptive dropout method. Instead of using a constant 0.5 dropout rate, the adaptive dropout (also called standout) regularizes its hidden units by selectively setting activities to 0 according to an adaptive rate determined by two user-defined parameters: α and β [4].

RESULTS

Parameter Estimation and Model Selection

A deep learning model has a number of parameters to determine, such as the number of hidden layers, the number of hidden nodes in each hidden layer, learning rates, and dropout rates. Choosing an appropriate deep learning model for a specific task is vital. Here, two datasets S388 and S65 are used for selecting models. We conduct a 20-fold cross-validation on the S388 dataset to test different sets of parameter values. The parameters that work well in the cross-validation are used to train deep learning models on whole S388 dataset, which are then tested on the independent test dataset S65.

Table 1. Eight sets of good parameters tested on S388 dataset according to 20-fold cross-validation. The first column lists the index of each parameter set, columns 2-6 the parameters (the number of hidden units in each hidden layer, learning rate in pre-training phase, learning rate in fine-tuning phase, dropout protocol, parameters of dropout protocol), and the last column the overall classification accuracy – the percent of correctly classified mutations among all the mutations.

Index of Parameter Set	# of Hidden Units in each Layer Separated by ","	Pre-Training Learning Rate	Fine-Tuning Learning Rate	Dropout Protocol	Dropout Parameters	Classification Accuracy
1	100,60,40	0.1	1.5	Standard	Rate = 0.5	0.8513
2	40,20	0.1	1.5	Standard	Rate = 0.5	0.8631
3	40,20	0.1	15	Standard	Rate = 0.5	0.8631
4	20,12	0.1	1.5	Standard	Rate = 0.6	0.8553
5	60,40,20	0.1	15	*Standout*	$\alpha = 1$, $\beta = 5$	0.8500
6	40,20,16	0.3	1.5	*Standout*	$\alpha = 2$, $\beta = 3$	0.8500
7	40,20	0.1	1.5	*Standout*	$\alpha = 1$, $\beta = 5$	0.8500
8	20,16	0.1	0.5	*Standout*	$\alpha = 2$, $\beta = 3$	0.8579

Table 2. The accuracy of five selected deep learning models on S65 dataset. These models are trained on S388 dataset. The index of each model corresponds to its index in Table 1, where the parameters of the model can be found.

Model Index	Model Name	Accuracy	Precision (P)	Precision (N)	Specificity	Sensitivity	Correlation Coefficient
1	DBN-1	0.877	0.857	0.879	0.981	0.462	0.571
2	DBN-2	0.831	0.600	0.873	0.923	0.462	0.426
3	DBN-3	0.846	0.800	0.850	0.981	0.308	0.433
4	DBN-4	0.846	0.615	0.903	0.903	0.612	0.519
8	DBN-8	0.862	0.750	0.877	0.962	0.462	0.515

We investigated on how different values of several key parameters (i.e. the number of hidden layers, number of hidden units in each layer, pre-train learning rate, fine-tune learning rate, and standard or standout dropout parameters) influence the performance of the deep learning models. The results of eight sets of good parameter values tested on S388 via 20-fold cross-validation are listed in **Table 1**. When different numbers of hidden layers are tested, 2 or 3 hidden layers work best. More than 3 layers of networks produce more complicated models with the similar accuracy. So we chose to use 2 or 3 layers in order to get simpler models. Among different numbers of hidden units tested, 12 to 100 hidden units work reasonably well. The fine-tuning learning rate may vary in a wide range and most of them achieve good performance. When training a Deep Belief Network without the dropout technique, the model is extremely biased toward the predominant negative class due to the highly imbalanced distribution of the two classes in the dataset. When dropout is applied to our method, the model begins to perform in a much more balanced way as reported in [3]. Both standard dropout [3, 27] and adaptive dropout [4] are used in our model. The dropout rates of 0.5 and 0.6 for the standard dropout protocol work similarly well. In the standout dropout scheme, the highest accuracy of 85.79% is achieved with alpha=2, and beta=3, which is similar to the performance of the standard dropout protocol. As shown in **Table 1**, most of models using the standard dropout have slightly better performance than standout dropout ones.

After the cross-validation on S388, top 5 models (model indices: 1, 2, 3, 4, and 8) are selected to train 5 models (DBN-

1, DBN-2, DBN-3, DBN-4, DBN8) on the whole S388 dataset. The performance of these five models on the independent test dataset S65 is reported in **Table 2**. In order to assess the results, a confusion matrix is calculated for each experiment, which contains numbers of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Based on the confusion matrix, the overall accuracy (the percent of correctly classified mutations), precision on positive examples, precision on negative examples, specificity, sensitivity, and correlation coefficient are calculated as follows to gauge the prediction performance of the methods:

$$Precision(N) = \frac{TN}{FN + TN}$$

$$Specificity = \frac{TN}{FP + TN}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Corr.Coef. = \frac{TP \times TN - FP \times FN}{\sqrt{[(TP + FN) \times (TP + FP) \times (TN + FN) \times (TN + FP)]}} \quad (8)$$

All of the 5 models have similar accuracy (i.e. percent of correctly classified mutations) on the test dataset that is comparable to the cross-validation accuracy on S388 dataset. This demonstrates that the models do not overfit the training set S388. The specificities are usually higher than the sensitivities because the dataset has many more negatives than positives. The DBN-4 model has the highest sensitivity among the 5 models. DBN-1 and DB-3 have the best specificity as well as the high precision on positive examples. So we choose the parameters of DBN-1, DBN-3 and DBN-4, respectively, to test their performance on the entire S1165 dataset through cross-validation.

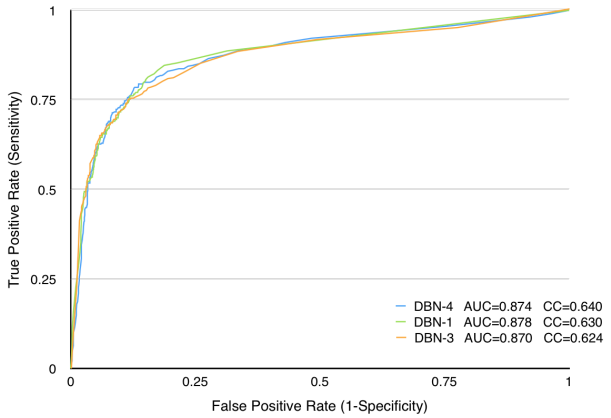

Figure 4. The ROC curves of three DBN models cross-validated on S1615. The AUC values and correlation coefficients are reported in the legend.

Results on the S1615 Dataset

To further investigate the performance of DBN models, the parameters of DBN-1, DBN-3, and DBN-4 are tested on the S1615 data using 20-fold cross-validation. The ROC curves of the three models are plotted and the Area Under the Curve (AUC) is calculated (**Figure 4**). The ROC curve is the receiver operating characteristic curve, which shows the relationship between true positive rate (sensitivity) and false positive rate (1 - specificity). This graphical plot can effectively illustrate a binary classifier's performance when the threshold on the false positive rate is varied.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Precision(P) = \frac{TP}{TP + FP}$$

As shown in **Figure 4**, the three ROC curves are very close and their AUCs are almost the same. DBN-1 has the best ROC curve with AUC value of 0.878, while DBN-4 has the highest correlation coefficient of 0.640. The results show that the DBN models are stable and robust.

Comparison with Other Mutation Stability Prediction Methods

Table 3 reports the performance of our three DBN models (DBN-1, DBN-3, and DBN-4) on the S1615 dataset in comparison with other four popular mutation stability predictors: AdaBoost/C4.5, iPTREE, NeuralNet and MUpro. Among these methods, AdaBoost/4.5, iPTREE, and NeuralNet use both sequence and structural information as input, while the three DBN models and MUpro only use sequence information as input.

Among all of the methods, AdaBoost/C4.5 has the highest overall accuracy (~87%). It applies AdaBoost algorithm on C4.5 decision tree and uses both sequence and structure information as input. The overall accuracy (~86%) of each DBN method that uses only sequence information as input is only slightly (~1%) lower than AdaBoost/4.5, and it is about 2% higher than MUpro that uses sequence information only. In particular, DBN's precision on the positive examples is around 80%, which is higher than all other methods. This makes the DBN methods very useful for identifying rare mutations that increase the stability of proteins in the real-world situation. Therefore, the DBN's good performance makes it generally applicable to the vast majority of proteins without known experimental structures.

Table 3. Results of three deep earning models (DBN-1, DBN-3 and DBN-4) on S1615 dataset in comparison with AdaBoost/C4.5, iPTREE, NeuralNet and MUpro.

Method	Input Information	Accuracy	Precision(P)	Precision(N)	Specificity	Sensitivity	Correlation Coefficient
AdaBoost/C4.5	Sequence & Structure	0.872	0.796	0.898	0.929	0.733	0.670
Random Forest	Sequence & Structure	0.862	0.771	0.894	0.919	0.713	0.650
iPTREE	Sequence & Structure	0.871	0.836	0.881	0.949	0.668	0.670
NeuralNet	Sequence &	0.810	0.710	0.830	0.910	0.520	0.490

	Structure						
MUpro	Sequence only	0.841	0.693	0.888	0.897	0.711	0.590
DBN-3	Sequence only	0.862	0.824	0.871	0.953	0.608	0.624
DBN-1	Sequence only	0.856	0.782	0.880	0.927	0.673	0.630
DBN-4	Sequence only	0.861	0.808	0.877	0.940	0.660	0.640

Protein Stability Prediction Tool and Web Server

We construct a user-friendly web server (DNpro) available at [38] for users to use our deep learning methods for protein stability prediction. Users can enter a protein sequence, mutated position, and mutated residue at the web page and then click "submit" button to get prediction results. The standalone DNpro package implemented in Java can also be downloaded at the same web site. The DNpro tool has a pre-trained model that can be used to predict protein stability change upon mutation. It also has training and cross-validation function that allows users to train and test a deep learning model on any input dataset.

CONCLUSION AND FUTURE WORK

In this work, we present a deep learning method to predict the protein stability change induced by single amino acid mutations. On a standard benchmark dataset, the deep learning method that uses protein sequence as only input performs better than another popular method that uses sequence information only and similarly to the most accurate methods that use both sequence and structural information. The experiment demonstrates that deep learning is a promising approach to the protein mutation stability prediction problem and the sequence-based deep learning method can be widely useful for studying the mutations of the vast majority of proteins without known tertiary structures. In the future, we plan to test different classifiers (e.g. support vector machines) other than logistic regression on top of the deep learning networks to see if the prediction accuracy can be further improved. A larger training dataset and additional input features may be used with deep learning to improve the prediction accuracy too.

REFERENCES

[1] J Cheng, A Randall, and P Baldi. Prediction of Protein Stability Changes for Single-Site Mutations Using Support Vector Machine. PROTEINS: Structure, Function, and Bioinformatics 2006;62:1125-1132.

[2] E Capriotti, P Fariselli, and R Casadio. A neural-network-based method for prediction protein stability changes upon singe point mutations. Vol. 20 Suppl. 1 2004, p i63-i68, DOI: 10.1093/bioinformatics/bth928.

[3] N Srivastava, G Hinton, and A Krizhevsky. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Reasearch 15 2014;1929-1958.

[4] L.J. Ba, and B Frey. Adaptive dropout for training deep neural networks. Advances in Neural Information Processing System 26, 2013.

[5] G Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Momentum, 2010.

[6] PA Bash, UC Singh, R Langridge, PA Kollman. Free-energy calculations by computer simulations. Science 1987;256:564-568.

[7] M Prevost, SJ Wodak, B Tidor, M Karplus. Contribution of the hydrophobic effect to protein stability: analysis based on simulations of the ile-96-ala mutation in barnsase. Proceeding of the National Academy of Sciences, USA 1991;88:10880-10884.

[8] B Tidor, M Karplus. Simulation analysis of the stability mutant r96h of t4 lysozyme. Biochemistry 1991;30:3217–3228.

[9] C Lee, M Levitt. Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. Nature 1991;352:448–451.

[10] C Lee. Testing homology modeling on mutant proteins: predicting structural and thermodynamic effects in the ala98-val mutants of t4 lysozyme. Folding and Design 1995;1:1–12.

[11] D Gillis, M Rooman. Predicting protein stability changes upon mutation using database-derived potentials: solvent accessibility determines the importance of local versus non-local interactions along the sequence. Journal of Molecular Biology 1997;272:276–290.

[12] MJ Sippl. Knowledge-based potentials for proteins. Current Opinion in Structural Biology 1995;5:229–235.

[13] D Gillis, M Rooman. Prediction of stability changes upon single-site mutations using database-derived potentials. Theoretical Chemistry Accounts 1999;101:46–50.

[14] H Zhou, Y Zhou. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. Protein Science 2002;11:2714–2726.

[15] H Zhou, Y Zhou. Quantifying the effect of burial of amino acid residues on protein stability. Proteins 2004;54:315–322.

[16] R Guerois, JE Nielsen, L Serrano. Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. Journal of Molecular Biology 2002;320:369–387.

[17] V Villegas, AR Viguera, FX Aviles, L Serrano. Stabilization of proteins by rational design of alpha-helix stability using helix/coil transition theory. Folding and Design 1996;1:29–34.28.

[18] V Munoz, L Serrano. Development of the multiple sequence approximation within the AGADIR model of alpha-helix formation: comparison with zimm-bragg and lifson-roig formalisms. Biopolymers 1997;41:495–509.

[19] H Domingues, J Peters, KH Schneider, H Apeler, W Sebald, H Oschkinat, L Serrano. Improving the refolding yield of interleukin-4 through the optimization of local interactions. Journal of Biotechnology 2000;84:217–230.

[20] AJ Bordner, RA Abagyan. Large-scale prediction of protein geometry and stability changes for arbitrary single point mutations. Proteins 2004;57:400–413.

[21] E Capriotti, P Fariselli, and R Casadio. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. Nucleic Acids Research, 2005

[22] G Hinton, S Osindero, Y Teh. A fast learning algorithm for deep belief nets. Neural computation 2006.

[23] Y Bengio, P Lamblin, and D Popovici. Greedy Layer-Wise Training of Deep Networks. Advances in neural information processing systems, 2007;19:153.

[24] G Hinton, and R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. Science, 2006;313(5786):504-507.

[25] R Casadio, M Compiani, P Fariselli, F Viarelli. Predicting free energy contributions to the conformational stability of folded proteins from the residue sequence with radial basis function networks. Proceedings of International Conference on Intelligent Systems for Molecular Biology, volume 3, p 81–88.1995.

[26] C Frenz. Neural network-based prediction of mutation-induced protein stability changes in staphylococcal nuclease at 20 residue positions. Proteins 2005;59:147–151.

[27] N Srivastava. Improving Neural Networks with Dropout. 2013.

[28] H Lee, R Grosse, and R Ranganath. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. Proceedings of the 26th Annual International Conference on Machine Learning, p609-616, 2009

[29] DW Hosmer Jr, S Lemeshow. Applied logistic regression. 2004.

[30] M Gromiha, J An, H Kono, M Oobatake, H Uedaira, P Prabakaran, A Sarai. Protherm, version 2.0: thermodynamic database for proteins and mutants. Nucleic Acids Res 2000;28:283–285.

[31] M Masso, L Vaisman. Accurate prediction of stability changes in protein mutants by combining machine learning with structure based computational mutagenesis. Bioinformatics(2008) 24(18): 2002-2009.

[32] C Worth, R Preissner, T Blundell. SDM- a server for predicting effects of mutations on protein stability and malfunction. Nucleic Acids Research, 2011.

[33] Y Dehouck, J Kwasigroch, D Gilis, M Rooman. PoPMusic 2.1: a web server for the estimation of protein stability changes upon mutation and sequence optimality. BMC Bioinformatics 2011, 12:151.
[34] Y Dehouck, A Grosfils, B Folch. Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. Bioinformatics (2009) 25 (19): 2537-2543.
[35] L Huang, M Gromiha, S Hwang, S Ho. Knowledge acquisition and development of accurate rules for predicting protein stability changes. Computational Biology and Chemistry 30 (2006) 408-415.
[36] V Parthiban, MM Gromiha, D Schomburg. CUPSAT: prediction of protein stability upon point mutations. Nucleic Acids Res 2006.
[37] C Deutsch, B Krishnamoorthy. Four-body scoring function for mutagenesis. Bioinformatics 2007.
[38] http://sysbio.rnet.missouri.edu/dnpros/