Protein Fold Class Prediction using Neural Networks with Tailored Early-Stopping

Thomas Wiebringhaus
Bioinformatics and Mathematics
Department of Applied Science
University of Applied Sciences of Gelsenkirchen
45665 Recklinghausen, Germany

Christian Igel
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
E-mail: c.igel@ieee.org

Jutta Gebert
Center for Applied Informatics Cologne
University of Cologne
50931 Cologne, Germany

Cologne University Bioinformatics Center
University of Cologne
50674 Cologne, Germany

***Abstract*— Predicting the three-dimensional structure of a protein from its amino acid sequence is an important problem in bioinformatics and a challenging task for machine learning algorithms. We describe an application of feed-forward neural networks to the classification of the protein fold class given the primary sequence of a protein. Different feature spaces for primary sequences are investigated, a tailored early-stopping heuristic for sparse data is introduced, and the achieved prediction results are compared to those of various other machine learning methods.**

I. Introduction

Resolving the biological function of proteins is an important task in bioinformatics. All proteins are made up of linear chains of amino acids. The sequence of amino acids—the primary sequence / structure—dictates the way in which the linear chain folds into a complex three-dimensional spatial structure, which determines the function of the molecule. Knowing the spatial structure of a protein is therefore necessary for the understanding of cellular function, evolutionary studies, drug design, artificial synthesis of proteins with desired properties in general, etc. While genome sequencing provides data about primary protein sequences, there is no practical way to compute the tertiary (spatial) structure from these sequences. However, some protein structures have been experimentally resolved and can function as templates for other sequence-structure examinations, e.g., by using them as training samples for machine learning algorithms. Starting from the work of [1], neural networks (NNs) have frequently been applied to protein structure prediction. Early studies concentrated on classifying the secondary protein structure using NNs [1], [2], [3], [4], [5], [6], [7]. Then NNs and other machine learning methods were also applied to determine the fold class of proteins [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. A fold class contains proteins that have tertiary structures with similar backbones. Recent publications propose to use Support Vector Machines (SVMs) for fold class prediction as they showed superior performance compared to NNs [11], [15]. However, in most of the experiments using NNs for protein structure classification little care was taken to build well generalizing models, e.g., by using an appropriate regularization method.

In the following, we apply standard feed-forward NNs to assign primary sequences of proteins to one out of 42 fold classes. No additional information about associated physiochemical properties are considered, although this would lead to an increase of classification accuracy (e.g., see [11]). We use early-stopping for implicit regularization to improve the generalization properties of the NNs [18]. In section II, we describe the representation of amino acid sequences by dipeptide-frequencies and present a new, lower dimensional encoding. Then we propose an early-stopping variant in the case of sparse training data. In section IV, we empirically evaluate our NN approach and the new encoding. The results are compared to those reported in the literature.

II. Embedding of Sequence Data

We consider the problem of assigning a primary sequence of a protein to one out of 42 fold classes [12], [15], i.e., classes of tertiary structures with similar backbones. Let P_k with

$$P_k = (s_1^k, \ldots, s_{q_k}^k) \text{ with } s_i^k \in \{A_1, \ldots, A_{20}\} \cup \{*\} \ , \quad (1)$$

be the primary sequence of a protein k with length q_k, where $A_1, \ldots, A_{20}$ denote the 20 amino acids and "$*$" stands for an incorrect or misclassified amino acid.

The primary sequences of the proteins have different lengths, but for the processing with most machine learning algorithms an embedding in a fixed-dimensional input space, the feature space, is necessary. Common, length-invariant—but also (almost) position-invariant—encodings are based on n-gram methods [13], where the occurrences of patterns of n consecutive residues in a sequence string are counted. Typically, each sequence k is represented by its relative dipeptide-frequency [8], [19], a 400-dimensional real-valued vector with components

$$h_{ij}^k = \frac{|\{l \,|\, s_l^k = A_i \wedge s_{l+1}^k = A_j\}|}{q_k - 1} \quad (2)$$

for $i, j \in \{1, \ldots, 20\}$. That is, h_{ij}^k corresponds to the relative frequency of how often A_i is followed by A_j in P_k. In order to reduce the dimensionality of the feature space, in [12]

a principal component analysis is applied and the input is restricted to the first 74 principal components.

We propose an alternative, low-dimensional representation by a 210-dimensional real-valued vector with components

$$\tilde{h}_{ij}^k = \frac{|\{l \,|\, (s_l^k = A_i \wedge s_{l+1}^k = A_j) \vee (s_l^k = A_j \wedge s_{l+1}^k = A_i)\}|}{q_k - 1} \tag{3}$$

for $i, j \in \{1, \ldots, 20\}$ and $i \leq j$. In this embedding, the order of the amino acids within the dipeptides is neglected, i.e., $\tilde{h}_{ij}^k$ corresponds to the relative frequency of how often A_i is followed by A_j or A_j is followed by A_i in P_k.

III. Early-Stopping and Sparse Data

The goal of NN training is not to learn the training patterns by heart, but to find a statistical model for the underlying relationship between input and output data. Such a model will generalize well, that is, will make good predictions for cases other than the training patterns (e.g., will classify unknown primary sequences correctly). A critical issue is therefore to avoid overfitting during the learning process: the NN should just fit the signal and not the noise [20]. This is usually done by restricting the effective complexity of the network, for example by regularization of the learning process. A common implicit regularization method is *early-stopping* [21], [18]. Early-stopping prevents overfitting in the following way: The data D_{train} available for training is split into two sets D_{learn} and D_{validate}. Training of the NN is done by minimizing an appropriate error measure on D_{learn}. During optimization, the error on the independent data set D_{validate} is monitored. For the final model those parameters (i.e., weights) are selected that have yielded minimum error on D_{validate} (e.g., see [21]).

However, this method is problematic when D_{train} is too small and one cannot afford to leave out samples for training because subsets of D_{train} would not accurately enough characterize the underlying data distribution. In this case, we propose a two stage training process: First, the network is trained on D_{train} until the classification error on D_{learn} vanishes (or falls below some predefined threshold). Then training is continued on D_{learn} until generalization loss (i.e., an error increase) is observed on D_{validate}. The weight configuration yielding the smallest error on D_{validate} is taken as the final outcome of the training process.

IV. Experimental Evaluation

A. Performance Comparison

For comparison of our NN approach to results in the literature, we used the same data and basically the same performance measures as in [12], [15]. The 268 sample patterns of primary sequences with corresponding fold classes are taken from the database for expected fold classes (DEF, [8], [19]).

Two measures for assessing the performance of different machine learning methods for fold class prediction have been used in [12], [15]. First, the available data are split as described in [12] into a set D_{train} of patterns available for training and a disjoint test set D_{test}, consisting of 143 and 125 patterns, respectively. Only D_{train} is used for building the statistical model (e.g., adapt the weights of a NN), whose prediction performance is measured on D_{test} afterwards. In our experiments, training and test set are exactly the same as in [15].

Second, the 10-fold cross-validation error, 10-CV, is computed. That is, the available data are split into 10 disjoint subsets. For each subset i, the other 9 subsets are used for building a statistical model which is then tested on subset i. The final 10-CV error is the average classification error over these 10 test errors. When computing the 10-CV error for NNs that are trained using early-stopping for regularization as described above we have two nested validation criteria. Note that our partitioning of the data into ten subsets need not be the same as in [12], [15]—this weakens the comparison.

B. Neural Network Architectures and Training

Standard NNs with a single hidden layer and shortcut connections (i.e., connections from the inputs directly to the outputs) were employed for classification [21]. We used a 1-of-42 output encoding and set ad hoc the number of hidden neurons to 10. Depending on the feature space, this yields a 400-10-42 or a 210-10-42 architecture. This means, the network has by far more weights than training patterns are available. However, the generalization performance of NNs is surprisingly insensitive to excess capacity, see [22] and [23] for theoretical and empirical studies, when combined with an appropriate regularization method. For example, early stopping in conjunction with a suitable learning algorithm starting from small initial weights can "be used to stop training large nets when they have learned models similar to those learned by smaller nets of optimal size" [22]. Here, an improved version of the Rprop learning algorithm was used for training [24], [25]. The weights are optimized with respect to the mean squared error (MSE), however, in the following all results refer to classification errors when not indicated otherwise.

For early-stopping as described in section III, we split D_{train} randomly (but ensuring an as equal distribution of the classes as possible) into D_{learn} and D_{validate} with 97 and 50 patterns, respectively. As we have only 143 training patterns for discriminating 42 classes, it is problematic to restrict the complete training process to a subset of D_{train}, i.e., early stopping can not be applied in the standard way. Thus, we used the two-stage training process as described above. The weight configuration with the smallest MSE on D_{validate} among those with the smallest classification error on D_{train} is selected as the outcome of the training process.

For the 10-fold cross-validation, where more data points are available for each training process, there was no need to apply the two stage training procedure. In addition, the classification error could not be reduced to zero for every training sample and hence an additional threshold parameter for switching between the two stages would have become necessary. Hence, standard early-stopping was used for regularization when determining the 10-CV results.

C. Results

In [12], 11 different classification methods were applied to the described data including linear and quadratic discriminant analysis, and k nearest neighbor classification. Support Vector Machines (SVMs, see [26], [27]) with different kernels were used in [15]. An SVM with radial basis function (RBF) kernel gave the best classification results in the two data sets scenario, an SVM with polynomial kernel yielded the lowest 10-CV error.

TABLE I

RESULTS GIVEN IN [12], [15] FOR SUPPORT VECTOR MACHINES (SVMS) WITH A RADIAL BASIS FUNCTION (RBF) KERNEL AND POLYNOMIAL KERNELS OF DEGREES 1, 2, AND 3 (POLY1, POLY2, POLY3), k NEAREST NEIGHBOR (kNN) CLASSIFICATION, AND LINEAR AND QUADRATIC DISCRIMINANT ANALYSIS (LDA, QDA). GIVEN ARE THE APPARENT PREDICTION ERROR (APE) ON D_{TRAIN}, THE TEST PREDICTION ERROR (TPE) ON D_{TEST}, AND THE 10-FOLD CROSS-VALIDATION CLASSIFICATION ERROR (10-CV).

error (%)	kNN	LDA	QDA	SVM			
				RBF	Poly1	Poly2	Poly3
APE (D_{train})	0.0	0.0	0.0	0.0	0.0	4.2	1.4
TPE (D_{test})	33.6	34.4	83.2	**23.2**	28.8	32.0	32.8
10-CV	34.0	30.2	38.4	30.2	**29.1**	35.0	34.2

TABLE II

RESULTS FOR THE TWO FEATURE SPACES ACHIEVED BY THE NEURAL NETWORKS IN OUR STUDY. GIVEN ARE THE MEAN $\pm$ STANDARD DEVIATION AND THE MEDIAN OF THE APPARENT PREDICTION ERROR (APE) ON D_{TRAIN}, THE TEST PREDICTION ERROR (TPE) ON D_{TEST}, AND THE 10-FOLD CROSS-VALIDATION CLASSIFICATION ERROR (10-CV). ALL RESULTS ARE BASED ON 20 TRIALS.

error (%)	NN			
	dimension 400		dimension 210	
	average $\pm$ sd.	median	average $\pm$ sd.	median
APE (D_{train})	0.0	0.0	5.1 ± 15.9	0.0
TPE (D_{test})	23.2 ± 1.7	**22.4**	26.8 ± 12.6	22.8
10-CV	27.2 ± 0.7	27.3	26.8 ± 1.0	**26.7**

For each feature space (400 and 210 inputs) and each performance measure (single test and training data set as well as 10-fold cross-validation) NNs starting from 20 independent, small random weight initializations were trained for 80 iterations. The outcomes of our experiments are shown in table II and figure 1, selected results from [12], [15] in table I. As obvious from figure 1 and confirmed by statistical tests, the results cannot be assumed to be normally distributed in most of the scenarios; many ties make the application of standard rank-sum-tests problematic. Hence, we decided not to present results from common statistical tests (although t-tests supported our conclusions).

Let us first consider the classification errors on a single test set D_{test} (the test prediction errors, TPEs). For both input dimensions most of the NN trials (including the one with the lowest MSE, i.e., the statistical model one would pick from

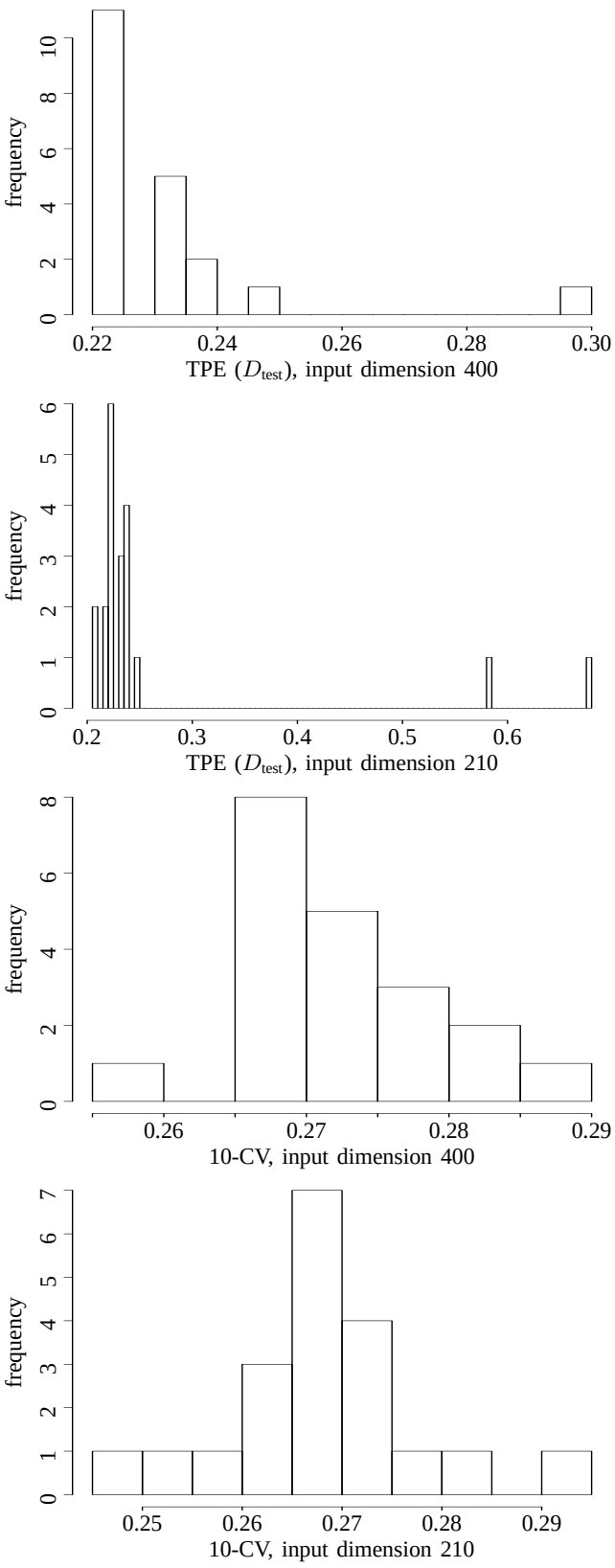

Fig. 1. Histograms of the results achieved by the neural networks in our study, see caption of table II.

the 20 possible) yielded a better classification error on the test set than the SVMs. The best TPEs of the NNs are 20.8 % and 22.4 % for input dimension 210 and 400, respectively, compared to 23.2 % reported for the best SVM in [15].

For input dimension 400, all trials classified the complete training set correctly (i.e., the apparent prediction errors, APEs, vanished). When the input dimension was reduced, two trials did not reach zero classification error. These two trials led to models with extremely bad classification rate on the test set (58.4 % and 68.0 %, respectively). They are responsible for the bad average result and large difference between median and average for input dimension 210. We conclude that one does not get worse NN classifiers with the reduced dimension, but the training process becomes less robust. This is also reflected by the standard deviations of the results, see table II.

The 10-CV results of the NNs are superior to those reported for other machine learning methods so far. The performance of the networks appears to be slightly better when neglecting the order of the amino acids. The 10-CV errors of the best NNs are 25.8 % and 24.7 % for input dimension 400 and 210, respectively, the medians are 27.3 % and 26.7 %. The best SVM in [15] achieved a 10-CV error of 29.1 %.

V. Discussion & Conclusions

Our experiments showed that standard feed-forward neural networks combined with an appropriate regularization scheme can classify the fold class of a protein given solely its primary sequence at least as good as other machine learning methods that have been applied to this problem [12], [15]. They clearly outperformed standard statistical approaches (like the nearest neighbor method etc. [12]) and did not perform worse than Support Vector Machines (SVMs). In particular, our 10-fold cross-validation results are better than those reported for SVMs in [15]. Note that it is sufficient to train a single NN for a few generations to get these results, whereas 42 SVMs have to be built for the one-against-all multiple-class discrimination approach in [15].

We proposed a two stage learning process when using early-stopping in the case of sparse training data, where first the training patterns are used completely for gradient calculation and are then split into a training and validation data set. This general heuristic led to good generalizing neural networks and is promising for other bioinformatics applications where the available training data are sparse.

We compared two embeddings of the primary sequences of the proteins, namely the relative dipeptide-frequency with and without considering the order of the amino acids within the dipeptides. Surprisingly, the reduction of the input dimension (i.e., throwing away all information about positions) did not lead to worse prediction results. Thus it appears that the order of the amino acids when computing relative dipeptide-frequencies does not contain information about the fold class that can be exploited by neural networks. This may carry over to other machine learning approaches and therefore the new embedding could be an appropriate way to reduce the input dimension for other methods applied to this task.

References

[1] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," *Journal of Molecular Biology*, vol. 202, pp. 865–884, 1988.

[2] H. Bohr, J. Bohr, S. Brunak, R. J. Cotterill, B. Lautrup, L. Nørskov, O. H. Olsen, and S. B. Petersen, "Protein secondary structure and homology by neural networks. The α-helices in rhodopsin," *FEBS Letters*, vol. 241, no. 1-2, pp. 223–228, 1988.

[3] L. H. Holley and M. Karplus, "Protein secondary structure prediction with a neural network," *Proceedings of the National Academy of Science USA*, vol. 86, pp. 152–156, 1989.

[4] D. G. Kneller, F. E. Cohen, and R. Langridge, "Improvements in protein secondary structure prediction by an enhanced neural network," *Journal of Molecular Biology*, vol. 214, pp. 171–182, 1990.

[5] P. Stolorz, A. Lapedes, and Y. Xia, "Predicting protein secondary structure using neural net and statistical methods," *Journal of Molecular Biology*, vol. 225, p. 363, 1992.

[6] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70 % accuracy," *Journal of Molecular Biology*, vol. 232, pp. 584–599, 1993.

[7] X. Zhang, J. P. Mesirov, and D. L. Waltz, "Hybrid system for protein secondary structure prediction," *Journal of Molecular Biology*, vol. 225, pp. 1049–1063, 1992.

[8] M. Reczko and H. Bohr, "The DEF data base of sequence based protein fold class predictions," *Nucleic Acids Research*, vol. 22, no. 17, pp. 3616–3619, 1994.

[9] J.-M. Chandonia and M. Karplus, "Neural networks for secondary structure and structural class prediction," *Protein Science*, vol. 4, pp. 275–285, 1995.

[10] J.-M. Chandonia and M. Karplus, "The importance of larger data sets for protein secondary structure prediction with neural networks," *Protein Science*, vol. 5, pp. 768–774, 1996.

[11] C. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[12] L. Edler, J. Grassmann, and S. Suhai, "Role and results of statistical methods in protein fold class prediction," *Mathematical and Computer Modelling*, vol. 33, pp. 1401–1417, 2001.

[13] C. H. Wu, "Gene classification artificial neural system," in *Methods in Enzymology: Computer Methods for Macromolecular Sequence Analysis*, R. F. Doolittle, J. N. Abelson, and M. I. Simon, Eds. Academic Press, 1996, vol. 266, pp. 71–88.

[14] C. H. Wu and J. W. McLarty, *Neural Networks and Genome Informatics*, ser. Methods in Computational Biology and Biochemistry. Elsevier Science, 2000, vol. 1.

[15] F. Markowetz, L. Edler, and M. Vingron, "Support vector machines for protein fold class prediction," *Biometrical Journal*, vol. 45, no. 3, pp. 377–389, 2003.

[16] D. Wang, N. K. Lee, and T. S. Dillon, "Extraction and optimization of fuzzy protein sequences classification rules using GRBF neural networks," *Neural Information Processing – Letters and Reviews*, vol. 1, no. 1, pp. 53–59, 2003.

[17] T. Wiebringhaus, U. Faigle, D. Schomburg, J. Gebert, C. Igel, and G.-W. Weber, "Protein fold class prediction using neural networks reconsidered," in *Currents in Computational Molecular Biology, The Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 2003)*, 2003, pp. 225–226.

[18] C. M. Bishop, "Regularization and complexity control in feed-forward networks," in *Proceedings International Conference on Artificial Neural Networks (ICANN'95)*, F. Fogelman-Soulié and P. Gallinari, Eds., vol. 1. EC2 & Cie, 1995, pp. 141–148.

[19] M. Reczko, D. Karras, and H. Bohr, "An update of the DEF database of protein fold class predictions," *Nucleic Acids Research*, vol. 25, no. 1, p. 235, 1997.

[20] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation,*, vol. 4, no. 1, pp. 1–58, 1992.

[21] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[22] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.
[23] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 402–408.
[24] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms," *Computer Standards and Interfaces*, vol. 16, no. 5, pp. 265–278, 1994.
[25] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithm," *Neurocomputing*, vol. 50, no. C, pp. 105–123, 2003.
[26] V. N. Vapnik, *Statistical Learning Theory*. New-York: Wiley, 1998.
[27] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.