STUDIA INFORMATICA 2011
Volume 32 Number 4A (100)

Katarzyna STĄPOR
Silesian University of Technology, Institute of Computer Science

USING MACHINE LEARNING APPROACH FOR PROTEIN FOLD RECOGNITION

Summary. Protein fold recognition using machine learning-based methods is crucial in the protein structure discovery, especially when the traditional sequence comparison methods fail because the structurally-similar proteins share little in the way of sequence homology. Based on the selected machine learning classification methods, we explain the methodology for building classifiers which can be used in the protein fold recognition problem.

Keywords: supervised learning algorithm, classifier, features, protein fold recognition

UCZENIE MASZYNOWE W ROZPOZNAWANIU KLASY UFAŁDOWANIA BIAŁKA

Streszczenie. Rozpoznawanie typu ufałdowania białka z wykorzystaniem metod uczenia maszynowego ma kluczowe znaczenie w przewidywaniu struktury białka, szczególnie w przypadkach kiedy tradycyjne podejście oparte na podobieństwie łańcuchów nie znajduje zastosowania ze względu na jego znikomą wartość. Na podstawie wybranych algorytmów uczenia maszynowego klasyfikacji w artykule przedstawiono metodykę automatycznego rozpoznawania typu ufałdowania białka.

Słowa kluczowe: uczenie nadzorowane, klasyfikator, cechy, przewidywanie typu ufałdowania białka

1. Introduction

Proteins are indispensable for the existence and proper functioning of biological organisms [27]. *Proteins* are biochemical **compounds** consisting of one or more **polypeptides** which are single linear **polymer** chain of **amino acids** bonded together by **peptide bonds** between the **carboxyl** and **amino** groups of adjacent amino acid **residues**. The **sequence** of amino acids in a protein known as *primary structure* is defined by the **sequence** of **genes**, which is encoded in the **genetic code** which, in general, specifies 20 standard amino acids. One of the most distinguishing features of polypeptides is their ability to fold typically into a globular or fibrous state, or "*structure*", that means *3D* (*three-dimensional*) or *tertiary structure*. According to Anfinsen [1], the proteins can fold to their native structures spontaneously, therefore he stated that protein fold is coded in the amino acid sequence itself, but it is still not clear as to how structure is encoded in a sequence and, therefore, it is an open problem of much scientific interest in computational biology. The *secondary structure* of proteins is the characterization of a protein with respect to certain local structural conformations like α-helices, β-sheets and other such as loops, turns and coils. *Fold* can be defined as a three-dimensional pattern characterised by a set of major secondary structure elements (e.g., α-helices and β-sheets) with certain arrangement and topological connections.

The structure of a protein serves as a medium through which to regulate either the function of a protein or activity of an enzyme. Understanding of how proteins fold in three-dimensional space can reveal significant information of how they function in biological reactions. Protein's function is strongly influenced by its structure [6, 25, 26, 27].

Currently, sequencing projects rapidly produce protein sequences, but the number of 3D protein structures increases slowly due to the expensive and time-consuming conventional *laboratory methods*, namely X-ray crystallography and nuclear magnetic resonance (NMR). Moreover, not all proteins are amenable to experimental structure determination. The protein sequence data banks such as Universal Protein Resource (UniProt) [2] contains over 5.000.000 protein sequence entries, while the number of stored protein structures in Protein Data Bank (PDB) [4] is less than 50.000.

This leads to the necessary alternative to experimental determination of 3D protein structures, the *computational methods* like ab initio and homology modeling ones. *Ab initio* methods seek to build 3D protein models "from scratch", i.e., based on physical principles [12],[32]. There are many possible procedures that either attempt to mimic protein folding or apply some stochastic method to search the space of possible solutions. The two major problems here are calculation of protein free energy and finding the global minimum of this energy which require vast computational resources, and have thus only been carried out for tiny proteins. These problems can be partially bypassed in the *homology-based* methods [12], [32], when the search space is pruned by the assumption that the protein in question adopts a structure that is close to the experimentally determined structure of another homologous protein. Because a protein's fold is more evolutionarily conserved than its amino acid sequence, a target sequence can be modeled with reasonable accuracy on a very distantly related template, provided that the relationship between target and template can be discerned through sequence alignment.

Among the computational approaches, *protein fold recognition* methods have taken central stage. Many methods have been developed, which are used to assigning folds to protein sequences. They can be broadly classified into three groups: 1) *sequence-structure homology* recogntion methods (for example [29]), 2) *threading* methods (for example Threader [18]), 3) *machine-learning-based* methods (for example [13, 28], see the review in the next section)

Sequence-structure homology and threading methods align target sequence onto known structural templates and calculate their sequence-structure compatibilities (scores) using for example environment-specific substitution tables or pseudo-energy-based functions, and the template with the best score is assumed to be the fold of the target sequence. While these methods each are effective in certain cases, there are drawbacks of both approaches. The first will fail when two proteins are structurally-similar but share little in the way of sequence homology. Threading methods rely on data derived from solved structures, but as we mentioned, the number of proteins whose structure has been solved is much smaller then the number of proteins that have been sequenced. These methods have not been able to achieve accuracies greater than 30%.

Machine learning-based methods for protein fold recognittion assume (according to [10] that the number of protein folds in the universe is limited, and therefore the protein fold recognition can be viewed as a *fold classification problem*: using sequence-derived features (properties) of proteins whose structure (fold) is known, so called the *learning* or *training set* for the construction of a *classifier* that can then be used to assign a structure-based label (class of fold) to an unknown protein (i.e. a protein whose structure has yet not been solved). The procedure for construction of a classifier is called *supervised learning* or *classifier training*. Its role in the fold classification task is to induce a mappings from primary sequences to folding classes.

Supervised learning methods have gained great interest since the work described in (Craven et al. [11]). Craven et al extracted several sequence-derived attributes, i.e., average residue volume, charge and polarity composition, predicted secondary structure composition, isoelectric point, Fourier transform of hydrophobicity function, from a set of 211 proteins belonging to 16 folds and used the sequence attributes to train and test the following popular classifiers: decision trees, *k* nearest neighbor and neural network classifiers in the 16-class fold assignment problem.

The main purpose of this article is to explain the main ideas and principles of machine learning-based methods for protein fold recognition. We illustrate these ideas by re-examining the selected, most popular four classification methods: gaussian classifier, nearest neighbors classifier, support vector machines and multi-layered perceptron. We show how these classifiers work as protein fold predictors using widely known protein dataset and using the calculated set of features, devised previously by other researchers.

To realize the main aim of the article, the second section contains the basic machine learning terminology and the short description of the mentioned four classifiers, which is necessary to understand machine learning paradigm. Then, at the beginning of the third section is a short review of the existing approaches to fold prediction based on the machine learning paradigm. The other subsections of the third section describe the process of building a classifier for protein fold prediction: how we extract features of the amino acid sequence and the dataset used in our experiments, and at last – the results of the conducted experiments. In the last, the forth section there are the conclusions drawn from the current study as well as the directions for future research in the field of machine learning-based fold prediction.

2. Supervised machine learning classification techniques

Machine learning is a branch of artificial intelligence which is concerned with the development of learning algorithms that allow computers to evolve their behavior based on the empirical data (*examples*). Based on the examples a learning algorithm captures characteristics of interest, for example the underlying probability distribution to automatically learn to recognize complex patterns and make intelligent decisions based on data. Because examples compose only a small subset of a whole population, the learning algorithm must *generalize* from the given examples, so as to be able to produce a useful output in future, new cases.

The objective of *statistical classification* is to assign a new observation x_i to one of the pre-specified c classes. The *supervised machine learning* of classification problem [5, 31], [30] can be formally stated as follows. Suppose we have a dataset of n examples (*training dataset*):

$$U_n = \{(x_1, y_1), ..., (x_n, y_n)\}$$

where each $x_i = (x_{i1}, ..., x_{id})$ represents an *observation*, $y_i \in \{1, ..., c\}$ is a categorical variable, a *class label*. We seek for a function $d(x)$ such that the value of $d(x)$ can be evaluated for any new observation x (i.e. not included in a training dataset) and such that label $\hat{y} = d(x)$ predicted for that new observation x is as close as possible to the true class label y of x. The function $d(x)$ known as *classifier* is an element of some space of possible functions, usually called the *hypothesis space*.

In the Bayesian decision framework, in order to measure how well a function fits the training data, a *loss function* $L(y, d(x))$ for penalizing errors in prediction is defined. By far, the most common is 0-1 *loss function*, where all misclassifications are charged a single unit. This leads to a criterion for choosing $d(x)$ as the expected prediction error $E(L(y, d(x)))$, where the expectation is taken with respect to the joint probability distribution $f(x, y)$. Conditioning on x and using 0-1 loss function we obtain a solution, a function $d(x)$ of the form:

$$d(x) = \arg\max_i P(i \mid x) \quad i = 1, \ldots, c$$

or equivalently, using the Bayes rule:

$$d(x) = \arg\max_i f(x \mid i) \cdot P(i)$$

where $f(x \mid i)$ is a class-conditional density, $P(i)$ is a priori probability of class i. The obtained classifier $d(x)$, known as *Bayes classifier*, says that we classify to the most probable class using the conditional distribution.

Classifiers come in a great diversity of techniques and algorithms. Each classifier can be uniformly defined by the set of *c discriminant functions*. Each class *i* has its own discriminant function $d_i(x)$ designed in such a way that for each object from class *i* the value of the corresponding discriminant function is (should be) the largest among the all *c* values:

$$\underset{\substack{j=1,\ldots,c \\ j \neq i}}{\forall} \quad d_i(x) > d_j(x)$$

Discriminant functions are determined based on the training set using different algorithms, depending on the particular classifier. This procedure is known as the classifier *training* or *learning*. The procedure of building a classifier for a particular application typically comprises the following steps [30]: 1) data collection (on appropriate features), 2) data preprocessing (for example normalization, outlier detection), 3) feature selection-/extraction (to avoid curse of dimensionality [5, 30]), 4) classifier training and validation of its internal parameters, 5) classifier testing to estimate its performance. The built classifier can then be used for making predictions on new, unknown observations. The unknown observation *x* is usually assigned to a class whose discriminant function $d_i(x)$ has the largest value [30].

2.1. Gaussian classifiers

The discriminant functions of the above described Bayes classifier are of the form:

$$d_i(x) = f(x \mid i) P(i) \qquad i = 1, \ldots, c$$

Many classification methods are based on the Bayes classifier including *parametric* and *nonparametric* ones according to the estimation method used. The most popular are parametric *gaussian classifiers* which use the gaussian (normal) density:

$$f(x\,|\,i)=\frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}}\exp\left[-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right]$$

where μ and Σ are the parameters: mean vector and covariance matrix of a distribution and d is a space dimensionality.

In the real situations, the parameters μ and Σ as well as a priori class probabilities $P(i)$ are replaced by their maximum likelihood (ML) estimates:

$$\hat{\mu}_i=\frac{1}{n_i}\sum_{j=1}^{n_i}x_j$$

$$\hat{\Sigma}_i=\frac{S_i}{n_i}=\frac{1}{n_i}\sum_{j=1}^{n_i}(x_j-\hat{\mu}_i)(x_j-\hat{\mu}_i)^T \quad i=1,\dots,c$$

$$\hat{P}(i)=\frac{n_i}{n}$$

based on a training set. Plugging the above expressions into discriminant function of a Bayes classifier results in the *quadratic* discriminant function of a gaussian classifier:

$$d_i\,(x)=-\frac{1}{2}(x-\hat{\mu}_i)^T\hat{\Sigma}_i^{-1}(x-\hat{\mu}_i)-\frac{1}{2}\ln|\hat{\Sigma}_i|+\ln\hat{P}(i)$$

In the simplest, special case where covariance matrices in all c classes are identical ($\Sigma_i=\Sigma,\ i=1,\dots,c$), the discriminant function of a gaussian classifier is *linear*:

$$d_i\,(x)=x^T\hat{\Sigma}^{-1}\hat{\mu}_i-\frac{1}{2}\hat{\mu}_i^T\hat{\Sigma}^{-1}\hat{\mu}_i+\ln\hat{P}(i)$$

However, when the number of training examples is small compared to the number of dimensions d there may be a problem in obtaining good ML estimates of class covariance matrices. One solution is to use *regularized estimators* proposed in [15]:

$$\hat{\Sigma}_i(\lambda)=(1-\lambda)\hat{\Sigma}_i+\lambda\hat{\Sigma} \quad 0\le\lambda\le1$$

where the parameter λ determines the amount of „shrinkage" of individual matrices towards the pooled one, and $\hat{\Sigma}$, is the pooled (average) covariance matrix:

$$\hat{\Sigma}=\frac{\sum_{i=1}^{c}S_i}{\sum_{i=1}^{c}n_i} \qquad S_i=\sum_{j=1}^{n_i}(x_j-\hat{\mu}_i)(x_j-\hat{\mu}_i)^T$$

Such a classifier is called *regularized gaussian classifier*.

2.2. Nearest neighbors classifiers

Another possibility is to use nonparametric density estimators, for example kernel or nearest neighbors ones [30], which leads to the different nonparametric Bayes classifiers. Based on the nearest neighbor density estimator and using the discriminant function of the Bayes classifier we can obtain very popular the *k nearest neighbor* (*knn*) classifier. This classifier is based on a distance function for pairs of observations such as Euclidean distance, and proceeds as follows to classify test set observations on the basis of the training set. For each element in the test set: 1) find the *k* closest observations in the training set, and 2) predict the class label by majority vote, i.e., choose the class that is most common among the *k* neighbors. The number of neighbors *k* is usually chosen in a validation step using *n-fold cross-validation* (*n-CV*) procedure (described in the experimental results section).

2.3. Support Vector Machine

Linear support vector machine (SVM) binary classifier is defined by the *optimal separating hyperplane* (OSH), i.e., the one which maximizes the separation margin which is the distance between the hyperplane and the closest training observations (called *support vectors*). In the case when the data are not linearly separable, a non linear transformation is used to map the input data vectors into a higher dimensional space using a kernel function $K(x_i, x)$. The discriminant function of a binary SVM classifier can be written as:

$$d(x) = \operatorname{sgn}\left(\sum_{i=1}^{|SV|} \alpha_i y_i K(x_i, x) + b \right)$$

where $0 \le \alpha_i \le C$ (*i = 1,…,n*) are Lagrange multipliers, *C* is a regularization parameter, *b* is a constant, both obtained through a numerical optimization during learning, |SV| is the number of support vectors. The originally defined SVM is a binary classifier and one way for using it in a multi-class classification problem is to adopt standard techniques for combining the results of binary classifiers. The most popular are one versus all (*1-all*) and *one* versus *one* (*1-1l*) [5]. With *1-all* approach, a binary classifier is constructed to decide between two classes: the class in question and the rest. Given *c* classes, *c* different classifiers are constructed and an unknown observation is assigned the label of whatever classifier returns a yes vote. In the case of multiple ‘yes’ votes, a number of different tie-breaking solutions have been proposed. Using *1-1* strategy and a dataset with *c* classes, a classifier is constructed for every possible pair of classes, resulting in *c(c-1)/2* different binary classifiers. Given an input observation, it is tested with each classifier, and the class returning the largest number of ‘yes’ votes is assigned to the observation.

2.4. Multi-layer perceptron

The multi-layer perceptron (MLP) [5, 30] also termed *feedforward neural network* is a generalization of the single-layer perceptron. In fact, just three layers (including the input layer) are enough to approximate any continuous function. The input nodes form the input layer of the network. The outputs are taken from the output nodes, forming the output layer. The middle layer of nodes visible to neither the inputs nor the outputs, is termed the *hidden* *layer*. The discriminant functions of 3-layered perceptron with M neurons in a hidden layer are of the following form:

$$d_i(x) = f_2\left(\sum_{j=0}^{M} w_{ij}^2 f_1\left(\sum_{r=0}^{d} w_{jr}^1 x_r^0\right)\right) \qquad i = 1,\dots,c$$

where x_r^0 are inputs, w_{ij}^1, w_{jr}^2 are components of two layers of network weights, d is the dimensionality of the input pattern, the univariate functions f_1 and f_2 are typically each set to:

$$f(x) = \frac{1}{1+e^{-x}}$$

The parameters of the network (i.e. weights) are modified during learning to optimize the match between outputs and targets, typically by minimizing the total square error using a variant of gradient descent which is conveniently organized as a *backpropagation of errors* [5].

3. Protein fold classification using machine learning methodology

3.1. Review of the existing methods

The protein fold recognition performance using machine learning paradigm critically depends on two main criteria: features used and employed classifiers. To address the first issue, a variety of features extracted and employed to this task such as: global physicochemical-based features [13, 14] order-based on bi-gram [17], tri-gram [24], and pseudo amino acids composition concepts [7], secondary structure-based features [24], substitution matrices-based [33], and hydrophobicity-based features [28].

However, employing an appropriate classifier has also critical effect to the problem. Researchers used different classification methods for the problem of fold classification. Except the mentioned work by Craven (Craven et al. [11]) good examples are Dubchak et al. [14] and Ding and Dubchak [13] who experimented with one-versus-others unique, one-versus-others and all-versus-all methods using neural networks or support vector machines (SVMs) as classifiers in multiple binary classification tasks on a set of proteins taken from 27 most populated SCOP folds. They were able to recognize the correct fold with the accuracy of approximately 56% using a number of sequence based properties as feature vectors for their classifier. A modified nearest neighbor algorithm called K-local hyperplane (HKNN) was used by Okun [23]. An effective example of methods based on Hidden Markov Model (HMM) is the one using reduced state-space HMM described by Lampros Ch. et al. [19]. Classifying the same dataset as in Dubchak [14] and input features, employing a Bayesian Network-based approach, Chinnasamy et al. improve to 60% on the average fold recognition results reported by Dubchak. The fusion of the different classifiers is widely used to improve the performance (to 63% on the average). For example Nanni [21] proposed ensemble of classifiers: Fisher linear classifier and HKNN, as well as series of SVM classifiers combined with the max rule [22], Shen and Chou [28] proposed ensemble model based on nearest neighbors, Chmielnicki and Stąpor [9] proposed a hybrid regularized gaussian-SVM classifier.

3.2. Re-examination of the selected methods

3.2.1. The datasets

The investigations described in this paper were performed on the dataset developed by Ding and Dubchak [14] (D-B dataset). The D-B dataset contains 311 and 383 proteins for training and testing, respectively.

Table1

The protein folds used in experiments

Fold name	Structural class	Nr of proteins in the training set
Globin-like	α	13
Cytochrome c	α	7
DNA-binding 3-helical bundle	α	12
4-helical up-and-down bundle	α	7
4-helical cytokines	α	9
Alpha; EF-hand	α	6
Immunoglobulin-like β-sandwich	β	30
Cupredoxins	β	9
Viral coat and capsid proteins	β	16
ConA-like lectins/glucanases	β	7
SH-3 like barrel	β	8
OB-fold	β	13
Trefoil	β	8
Trypsin-like serine proteases	β	9
Lipocalins	β	9
(TIM)-barrel	α / β	29
FAD (also NAD)-binding motif	α / β	11
Flavodoxin like	α / β	11
NAD(P)-binding Rossman fold	α / β	13
P-loop containing nucleotide	α / β	10
Thioredoxin-like	α / β	9
Ribonuclease H-like motif	α / β	10
Hydrolases	α / β	11
Periplasmic binding protein-like	α / β	11
β-grasp	α + β	7
Ferredoxin-like	α + β	13
Small inhibitors, toxins, lectins	α + β	13

This dataset has been formed such that, in the training set, no two proteins have more than 35% sequence identity to each other and each fold have seven or more proteins. In the test dataset, proteins have no more than 40% sequence identity to each other and have no more than 35% identity to proteins of the training set. The proteins from training and testing datasets belong to 27 different folds (according to SCOP (*Structural Classification of Proteins*) classification [20]), representing all major structural classes α, β, $\alpha+\beta$ and α/β. The distribution of protein sequences of the training dataset in the 27 folds and four structural classes is presented in Table 1.

3.2.2. Features of protein sequence

A protein sequence is represented by a set of 126 element vectors based on various physico-chemical and structural properties of amino acids along the sequence: 1) *amino acid composition* (20 features collectively denoted by a letter '**C**' plus sequence length), 2) *predicted secondary structure* (21 features denoted by '**S**'), 3) *hydrophobicity* (21 features denoted as '**H**'), 4) *normalized van der Waals volume* (21 features denoted as '*V*'), 5) *polarity* (21 features denoted by '**P**') and 6) *polarisability* (21 features denoted by '**Z**').

We have used predicted secondary structural information based on three-state model: *helix*, *strand* and *coil* as the basis for all the calculations. The predictions were made by the method mentioned in Dubchak et al. [14] using artificial neural networks.

Apart from amino acid composition characteristics ('C' set of features), which is the feature vector that contains the percentage occurrences of amino acids in the primary sequence, all other features were extracted based on the classification of all amino acids into three classes for each of the five mentioned above attributes in the way describe below (for example polar, neutral, and hydrophobic for hydrophobicity attribute, see Table 2). The detailed description can be found in Dubchak et al. [14].

The descriptors *a-composition*, *transition* and *distribution* were calculated for each attribute to describe the global percent composition of each of the three groups in a protein, the percent frequencies with which the attribute changes its index along the entire length of the protein, and the distribution pattern of the attribute along the sequence, respectively.

In the case of hydrophobicity for example, the a-composition descriptor 'aC' consists of the three numbers – the global percent compositions of polar, neutral and hydrophobic residues in the protein (because regarding to hydrophobicity attribute, all amino acids are divided into three groups: polar, neutral and hydrophobic). The transition descriptor 'T' of the following three numbers – the percent frequency with which: a polar residue is followed by a neutral one or a neutral by a polar residue and similarly with the other two types of residues. The distribution descriptor 'D' consists of the five numbers for each of the three groups: the fractions of the entire sequence, where the first residue of a given group is located, and where 25, 50, 75, and 100 percent of those are contained. The complete parameter vector contains 3'aC'+3'T'+5x3'D'=21 components.

Therefore the full feature vector **(C, S, H, V, P, Z)** counts 6 x 21 = 126 features.

Table 2

Amino acid attributes and corresponding groups

Attribute	Group 1	Group 2	Group 3
Secondary structure	Helix	Strand	Coil
Hydrophobicity	Polar R,K,E,D,Q,N	Neutral G,A,S,T,P,H,Y	Hydrophobic C,V,L,I,M,F,W
Polarizability	(0-2.78) G,A,S,C,T,P,D	(2.95-4.0) N,V,E,Q,I,L	(4.43-8.08) M,H,K,F,R,Y,W
Polarity	(4.9-6.2) L,I,F,W,C,M,V,Y	(8.0-9.2) P,A,T,G,S	(10.4-13.0) H,Q,R,K,N,E,D
Van der Waals volume	(0-0.108) G,A,S,D,T	(0.128-0.186) C,P,N,V,E,Q,I,L	(0.219-0.409) K,M,H,F,R,Y,W

3.2.3. Experimental results

Each of the selected for re-evaluation classifiers: regularized gaussian, nearest neighbors, support vector machine, multi-layer perceptron was trained and then tested using the available datasets of 311 and 386 protein sequences respectively, and their sequence-derived feature vectors. As a measure of classifier accuracy we used the *percentage accuracy* (percentage of the examples incorrectly classified) on the test set.

When the number of the training samples is small compared to the number of dimensions of the feature vector, the problem of estimation of classifier parameters may be ill-posed. In our experiment we decided to use the simple selection algorithm to reduce the dimensionality of the feature space. The idea for "simplified" feature selection is based on the fact, that features used in our experiments are based on parameters C, S, H, V, P, Z that create six feature sets containing 21 values each, so all combinations of these sets can be considered. The total number of them is 63, so the brute force algorithm can be used. The best combination obtained using 10-fold cross-validation procedure on a training dataset was 63 feature set based on **C, S, P** sets of features.

The selected classifiers were built for various parameters that define their internal architecture and the best combination in terms of classifier accuracy measured by 10-fold cross-validation procedure on a training dataset for each one was remembered. Testing was performed on a test set.

N-fold cross-validation (N-CV) is the resampling technique that is often used for evaluation of classifier performance. The N-CV method consists of the following steps: 1) divide randomly all the training examples into *N* equal-sized subsets (usually *N*=10 but in general depends on the size of a training dataset), 2) use all but one subset of examples to train the classifier, 3) measure the classification error on the remaining one by means of the percentage accuracy, 4) repeat steps 2 and 3 for each subset, 5) average the results to get an estimate of the classification error of the classifier

The regularized gaussian classifier achieved the maximum accuracy **56%** for parameter $\lambda = 0,7$. The use of gaussian kernel:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2} \| x_i - x_j \|^2\right)$$

with parameters σ=0.5 , C = 100 and *1-1* method of combining binary classifiers leads to the best performing of SVM classifier – enabling to achieve the maximum performance equal to **58%.** Regarding the 2-layered perceptron the maximum average accuracy was obtained for 200 neurons and equals to **49%**. The *knn* classifier was trained and tested for several values of *k*, the optimal *k* value was found equal to 1 giving the performance **50%.**

4. Conclusions and future work

In this study we have explained the principles of using machine learning-based methodology in the complex classification task, namely the protein fold classification. We have investigated the following classifiers: regularized gaussian, nearest neighbors, support vector machine, multi-layer perceptron in the 27-class (fold) classification problem. It is clear from the experiments that none of the described classifiers alone give the accuracy better than 60%. But, the obtained result is very well acceptable accuracy for the 27-class classification problem !. A random classifier would have a **3.7%** (1/27x100) only. Moreover, the described and re-evaluated machine learning-based methods can be used to predict structure of proteins that have no well-understood homologes.

There are several open issues that one can plan to explore in future research. These include: investigating alternative input representations. We believe that the described representation is insufficient and can be improved by incorporating additional features describing amino acid chain. An alternative class structure should also be developed, for example by re-evaluation our current class structure to determine classes which should be aggregated or discarded, or by incorporating larger set of folding classes. At last, better generalization of classifiers can be achieved by developing more accurate learning algorithms. One such possible improvement relies on using better hybrid classification methods (from those describe in the review) which allow to achieve higher accuracy.

BIBLIOGRAPHY

1. Anfinsen C. B.: Principles that govern the folding of protein chains. Science 181, 1973, p. 223÷230.

2. Apweiler R, Bairoch A, Wu CH. et al.: UniProt: the universal protein knowledgebase. Nucleic Acids Res. 32 (Database issue), D115–9, 2004.
3. Baldi P., Brunak S., Chauvin Y., Andersen C., Nielsen H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16, 2000, p. 412÷424.
4. Berman H. M. et al.: The Protein DataBank. Nucleic Acids Res., No. 28, 2000, p. 235÷242.
5. Bishop Ch. M.: Pattern recognition and machine learning. Springer, New York 2006.
6. Chan H. S., Dill K.: The protein folding problem. Physics Today (Feb.), 1993, p. 24÷32.
7. Chen K. C., et al.: Using pseudo amino acid composition and support vector machine to predict protein structural class. Journal of Theoretical Biology, No. 243, 2006, p. 444÷448.
8. Chinnasamy A., Sung W. K., Mittal A.: Protein structure and fold prediction using tree-augmented naïve Bayesian classifier. Proc. PSB, Stanford CA, 2004, World Scientific Press.
9. Chmielnicki W., Stąpor K.: Protein fold recognition with combined RDA-SVM classifier. Lecture Notes on Artificial Intelligence, LNAI 6076, 2010, p. 162÷169.
10. Chothia C.: One thousand families for the molecular biologist. Nature, No. 357, 1992, p. 543÷544.
11. Craven M. W., Mural R. J., Hauser L. J., Uberbacher E. C.: Predicting protein folding classes without overly relying on homology. Proc. of Intelligent Systems In Molecular Biology (ISMB) No. 3, 1995, p. 98÷106.
12. Cymerman I. A. et al.: Computational methods for protein structure prediction and fold-recognition. In: Nucleic Acids and Molecular Biology series, "Practical Bioinformatics", 2004, Editor: Bujnicki J. M. Springer-Verlag.
13. Ding C.H., Dubchak I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics, No. 17, 2001, p. 349÷358.
14. Dubchak I., Muchnik I. Holbrook S. R., Kim S. H.: Prediction of protein folding class using global description of amino acid sequence. Proc. Natl. Acad. Sci. USA, No. 92, 1995, p. 8700÷8704.
15. Friedman J.H.: Regularized Discriminant Analysis. Journal of the American Statistical Association, No. 405, 1989, p. 165÷175.
16. Ghanty P., Pal N. R.: Prediction of protein folds: extraction of new features, dimensionality reduction, and fusion of heterogeneous classifiers. IEEE Trans. on Nano-bioscience 8, 2009, p. 100÷110.

17. Huang C. D., Lin C. T., Pal N. R.: Hierarchical learning architecture with automatic feature selection for multiclass protein fold classification. IEEE Trans. on Nanobioscience, No. 2, 2003, p. 221÷232.
18. Jones D. T. et al.: A new approach to protein fold recognition. Nature, No. 358, 1992, p. 86÷89.
19. Lampros Ch. et al.: Sequence-based protein structure prediction using a reduced state-space hidden Markov model. Computers in Biology and Medicine, No 37, 2007, p. 1211÷1224.
20. Lo Conte L., Ailey B., Hubbard T. J. P., Brenner S. E., Murzin A. G., Chothia C.: SCOP: a structural classification of protein database. Nucleic Acids Res., No. 28, 2000, p. 257÷259.
21. Nanni L.: A novel ensemble of classifiers for protein fold recognition. Neurocomputing, No. 69, 2006, p. 2434÷2437.
22. Nanni L., Lumini A.: Mpps: an ensemble of support vector machine based on multiple physicochemical properties of amino acids. Neurocomputing, No. 69, 2006, p. 1688÷1690.
23. Okun O.: Protein fold recognition with k-local hyperplane distance nearest neighbor algorithm. In: Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics, No. 2, Pisa, Italy, 2004, p. 51÷57.
24. Pal N. R., Chakraborty D.: Some new features for protein fold prediction. Proc. Int. Conf. On Artificial Neural Networks and Neural Information Processing, 2003, p. 1176÷1183.
25. Roterman I., Bryliński M., Konieczny L., Jurkowski W.: Early-stage protein folding - in silico model. In: Recent Advances in Structural Biology. A.G. de Brevern (eds.), Research Signpost, Trivandrum, Kerala, 2007, India.
26. Roterman I., Konieczny L, Bryliński M.: Late-stage folding intermediate in silico model. In: Structure-function relation in proteins. Roterman I. (ed.). Transworld Research Network T.C. 37/661(2), 2009, Fort P.O. Trivandrum, Kerala, India.
27. Konieczny L. Roterman I., Spólnik P.: System Biology. The strategy of functioning of living organism (in Polish). Scientific Publishing House PWN, Warsaw 2010.
28. Shen H. B., Chou K. C.: Ensemble classifier for protein fold pattern recognition. Bioinformatics, No. 22, 2006, p. 1717÷1722.
29. Shi J. et al.: FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. J. Mol. Biol., No. 310, 2001, p. 243÷257.
30. Stąpor K.: Classification methods in computer vision (in Polish). Scientific Publishing House PWN, Warsaw 2011.
31. Vapnik V.: The Nature of Statistical Learning Theory. Springer, New York 1995.

32. Ying X., Dong X., Jie L.: Computational methods for protein structure prediction and modelling. Vol. 2: Structure prediction, Springer, New York 2007.
33. Ying Y., Huang K., Campbell C.: Enhance protein fold recognition through novel data integration approach. BMC Bioinformatics, No.10, 2009, p. 267÷287.

Recenzent: Dr hab. inż. Krzysztof Fujarewicz

Wpłynęło do Redakcji 22 sierpnia 2011 r.

Omówienie

Rozpoznawanie typu ufałdowania białka z wykorzystaniem metod uczenia maszynowego ma kluczowe znaczenie w przewidywaniu struktury białka, szczególnie w przypadkach kiedy tradycyjne podejście oparte na podobieństwie łańcuchów nie znajduje zastosowania ze względu na jego znikomą wartość. Na podstawie wybranych algorytmów uczenia maszynowego klasyfikacji w artykule przedstawiono metodykę automatycznego rozpoznawania typu ufałdowania białka. W pierwszej części zostały przedstawione ogólne zasady uczenia maszynowego klasyfikacji wraz z przykładowymi takimi algorytmami dla klasyfikatora empirycznego Bayesa, wielowarstwowego perceptronu, maszyny wektorów podpierających oraz metody najbliższych sąsiadów. W dalszej części po krótkim przeglądzie istniejących podejść do rozpoznawania typu ufałdowania białka za pomocą automatycznej klasyfikacji dokonano opisu sposobu reprezentowania łańcucha aminokwasów, czyli przestrzeń cech, a także przedstawiono opis 27 klas ufałdowań białek, które standardowo wybiera się dla testowania algorytmów automatycznej klasyfikacji. Ostatnie dwa punkty opisują eksperyment oraz uzyskane wyniki i wnioski. Jak wynika z przeprowadzonych badań, podejście wykorzystujące uczenie maszynowe klasyfikacji stanowi bardzo obiecujący kierunek w badaniach nad przewidywaniem struktury trzeciorzędowej białka.

Adrress

Katarzyna STĄPOR: Silesian University of Technology, Institute of Computer Science,
ul. Akademicka 16, 44-100 Gliwice, Poland, katarzyna.stapor@polsl.pl.