A Feature-Based Approach to Discrimination and Prediction of Protein Folding Groups

Boris Mirkin [*,†]   Otto Ritter [*,‡]

1 Introduction

Proteins are the fundamental molecules of life, they form both the structural and the functional building blocks of cells in all living organisms. Proteins are also the most complex molecules. Their three-dimensional structures are extremely diverse, and so are their biological functions. Elucidation of the relationship between protein sequence, structure, and function is one of the most significant open problems in science and technology. Despite the growing body of observed data and partial theories, there are still no satisfactory methods which would enable us to analyze, characterize, and predict a protein's structure and/or function with regard to its known primary sequence.

There are two fundamentally different approaches to the problem, and a continuum of their combinations. The first approach is based on theory and models from physics and chemistry, the other is based on computational and statistical analysis of observed data. This paper belongs to the second direction.

From the data analysis perspective, we can observe, first, that there has not been much achieved in terms of producing reasonable feature spaces except for those directly related to amino acid sequence similarity data, and, second, the specifics of emerging problems yet have not been addressed in full. One of the specifical issues is related to difference between prediction and description. Typically, prediction is based on a description of the phenomenon in question in the framework of a prediction model. However, such a description not always can be exploited for substantive analyses. For instance, the neural network based predictions of protein folding classes involve neural net models that are useless from the point of view of the biologist: they may give a good prediction, but the net structure and corresponding weights can provide no insights into the biological nature of the genomic processes, at least currently. That means that the problem of description in proteonomics becomes a problem on its own, which must be addressed accordingly.

[*] Department of Molecular Biophysics, DKFZ, Heidelberg, Germany

[†] DIMACS, Rutgers University, Piscataway, NJ, USA

[‡] Biology Department, Brookhaven National Laboratory, Upton, NY, USA

In this paper, we restrict ourselves to considering the protein spatial structure learning problem as a problem in the framework of an existing classification of proteins, SCOP by Hubbard, Murzin, Brenner and Chothia (1997), that is suitable for our purposes because of both its substantive contents and availability in the Internet. The problem of learning SCOP classes is discussed in section 2 along with a review of most popular machine learning approaches. Then we suggest a strategy for solving the problem to involve the following four dimensions: (a) proteins are considered in terms of a feature space rather than in terms of their sequence/structure similarities; (b) the subgroups are supposed to be logically described by the features; (c) resampling is used as a learning tool rather than a testing device; and (d) multiscale representation of a sequence for searching through the feature space is used for learning. Although at least some of these dimensions already have been exploited in literature, their combination and, moreover, application to proteins have never been undertaken, to our knowledge. Discussion of the issues related to the four dimensions is done in four sections 3 through 6. In section 7, the suggested method, APPCOD, is formulated as a computational algorithm, and, in section 8, examples of computations performed with it are presented. Conclusion, section 9, contains a brief discussion of future work.

2 The Problem of Learning SCOP Classes

The problem of learning of a folding group can be put in a more or less statistical way by exploiting a system that has incorporated some knowledge of structural and evolutionary similarities among proteins. Such a system is Structural Classification Of Proteins (SCOP) currently maintained as a website, http://scop.mrc-lmb.cam.ac.uc/scop/. SCOP is a multi-level classification structure involving currently (in its version 1.37, we have been dealing with) over a dozen thousand protein domains from proteins in Protein Data Bank (see Fig. 1).

There have been a number of works reported recently with regard to learning the upper classes of SCOP, the structural α/β classes (first level of the hierarchy) and folding classes (second level of the hierarchy), see Chou et al. (1998), Dubchak et al. (1995, 1999) and references therein.

We consider the problem of learning SCOP classes in the following setting. Let M be an interior node in the SCOP hierarchy (the root, a class, fold, superfamily, or a family) and S another node descending from M (a class, fold, superfamily, family, or a protein domain, respectively). The problem is to find a rule separating protein domains in S from other proteins in M. If, for instance, M is the root (all proteins) and S the Globins family, the problem is to learn what separates Globins from the other proteins. Finding such a rule can be trivial: for instance, the definition of S in SCOP separates it in M, but it is based on the spatial structure information which should not be involved in the learning rule. In this paper, we concentrate on those separating rules that involve only terms related to the primary structure of proteins.

To formalize the problem, we have selected an approach that is based on

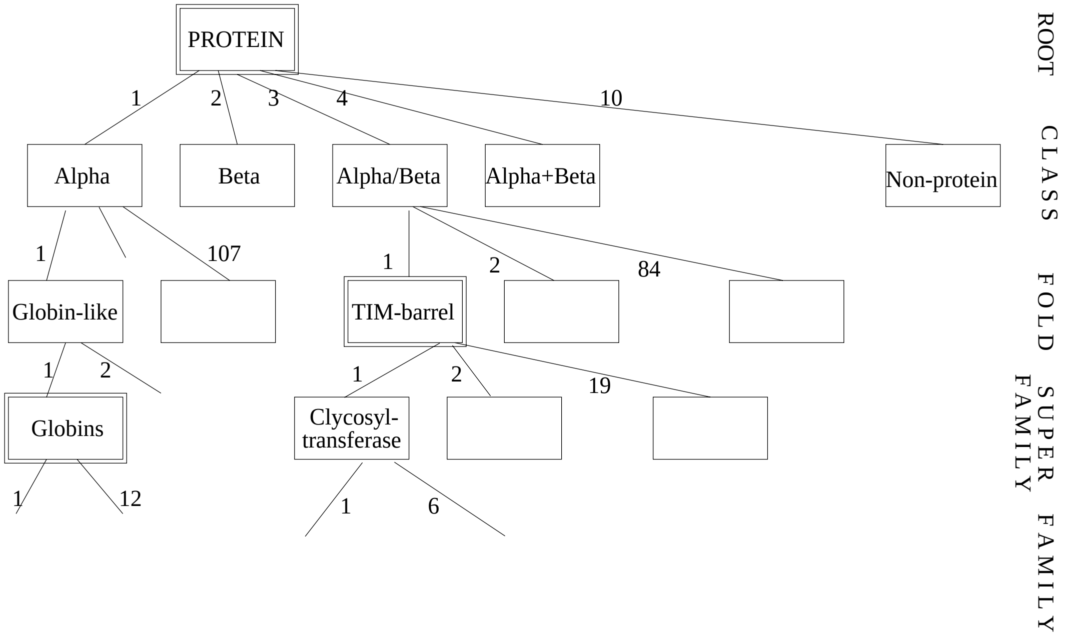

Figure 1: Upper levels of SCOP with some of the classes labelled by their names, some by their numbers (at the links).

features of the amino acids constituting the sequence. This way the entire issue of sequence/structure alignment, that attracts much of attention in the literature, becomes irrelevant. Thus all the gapping and weighting problems in comparing sequences are dropped out with the amino-acid-feature-based approach. Moreover, within this approach we apply a procedure that allows for formulation of an explicit and easy-to-interprete separation rule in a logical format that differs from most work in the field. Yet one more distinctive feature of our approach is that reliability of the rule is achieved by using the standard cross-validation techniques in the process of learning, not just in testing as usually. These four aspects along with related references will be reviewed in the following four sections.

3 Feature Based Descriptions

The problem of prediction of 3D structure of a protein based on its sequence has been met by researchers with a number of developments (for reviews, see Fisher and Eisenberg (1996), Jones (1997), Westhead and Thornton (1998) and paper by W. Taylor in this volume). These developments are mostly based on what is called threading: aligning a target sequence along a known 3D structure (fold) with a scoring function reflecting the energy needed for the sequence to fold into the 3D structure. The best match from a library of known folds then is considered to be the "parent structure" which is then updated according to specifics of the target sequence, to get a predicted fold (comparative modelling). Knowledge of (predicted) secondary structure elements is helpful in threading as well as use of the multialignment approach.

With the development of extensive classifications of the protein 3D structures, such as SCOP by Hubbard et al. (1997) or CATH by Orengo et al. (1997), a possibility emerged of exploring a less challenging task of predicting just a folding class, a node in the classification tree, not a fold. The difference is that a folding class can be represented with just the list of its members, without any explicit reference to their spatial structures. Most efforts so far have been devoted to prediction of the four major structural folding classes (all-α, all-β, $\alpha + \beta$, and α/β) by the amino acid composition. The amino acid composition of a sequence is the vector of relative contents (per cent) of each of the 20 amino acids in the sequence. Most recent results (75% of correct predictions overall in a jackknife experiment) and a review of the subject can be found in Chou et al. (1998). The state of the art in machine learning of the structural folding classes in the space of percentages of the secondary structure elements is described in Zhang and Zhang (1998).

More recently, an effort has been made in predicting other folding classes as, say, of the level of Fold in SCOP (see Fig. 1) by the team led by S.-H. Kim (see Dubchak et al., 1995, 1999). For prediction, they use a neural network based procedure for learning a class in a number of feature spaces with consequent averaging predictions based on a majority rule. Each of the feature spaces exploited in this approach consists of 21 features derived from an amino acid property categorized priorly in three categories. For instance, the amino acids can be categorized in categories of hydrophobic, hydrophilic, or neutral residues (according to their hydrophobicity), or in helix, extended (stranded), or coiled residues (according to the element of the secondary structure they belong to), etc. Having such a categorisation of a feature done, three variables are defined as percentage of each of the categories along the sequence; three more variables are defined as relative frequencies of change from one to another category along the sequence. Each of the remaining fifteen features is defined as the percentage of the length of the sequence when one of the three categories reaches one of the following five points: the beginning, the 25%, 50%, 75% of the number of residues in the category (quantile), and the end position of the category's occurrences. With thus produced feature spaces the method fared relatively well in the CASP2 experiment (Levitt, 1997) .

Some other feature spaces have been exploited in Bachinsky et al. (1997) and Hobohm and Sander (1995), though for less unambigous problems in maintaining of protein family structures. A most comprehensive set of features have been considered so far by Wei, Chang and Altman (1998): five classes of protein features have been specified as those based on atom, residue, chemical group, secondary structure, and other (such as B-factor or solvent accessibility). However, these spaces have been employed in Wei, Chang and Altman (1998) for within protein structure learning problems, such as recognition of protein sites, rather than for fold recognition.

In our study, we selected a residue-based feature space involving the order of residues in the sequence. Of four hundred features of amino acids available in the data base AAindex (see Kawashima et al., 1998), we selected six features related to size or charge of an amino acid in ProtScale library of the ExPASy website (see Appel et al., 1994). More specifically, the variables are molecular weight, MW, bulkiness, BU, (consensus) hydrophobicity, HY by Eisenberg-Schwarz-Komarony-Wall (1984), polarity [two scales], PG by Grantham (1974) and PZ by Zimmerman–Eliezer–Simha (1968), and membrane buried helix parameter, MB by Rao-Argos (1986), all from ProtScale at http://www.expasy.ch/tools/#primary. . These features, averaged over arbitrary intervals of the protein sequences, constitute our feature space. The sequence intervals are measured per cent as, for instance, the intervals [0,20] and [50,75] related to the initial 20% of the sequence length and positions between its midpoint and the three-quarter point, respectively. An advantage of this space, as well as of other spaces mentioned, is that no alignment of proteins is needed to measure the features. On the other hand, the features are relevant to some biochemical and biophysical properties of the sequences and their sites. Of course, these properties cannot express such intuitively defined features as " hydrophilic residues are allowed in the barrel interior but not between the barrel and helices" in Murzin and Bateman, A. (1997), p. 108-109, that involve primary, secondary and tertiary structure elements. However, a somewhat simpler feature like "hydrophilic residues are allowed in the middle of a sequence but not in its end" can be easily formulated in terms of our feature space: just the average hydrophobicity of an interval at the end must be larger than that of an interval in the middle. The comparison can be done by arithmetically combining the features involved. It means, actually, that the ratio of the features must be larger than 1. This way employing more complex logic expressions such as in PROGOL by Finn et al. (1998) can be avoided.

4 Description as a Tool in Machine Learning

4.1 The problem

Having a protein feature space specified, every protein can be represented as a multidimensional point in this space. Thus our learning problem can be reformulated in terms of Fig. 2: given a set of points M (all circles), find a way to separate those representing a subset $S \subset M$ (black colour).

The separation rule should allow to test any sequence to appear and assign it with black or white colour depending on its predicted belongingness to S. The quality of the separation rule can be characterized by the misclassification rate expressed with two quantities: the numbers/proportions of false positives and false negatives. The false negative is an instance of entity from S erroneously diagnosed by the separation rule as being from outside, that is, a black circle recognized as a white one. In contrast, the false positives are entities from outside of S that have been recognized as belonging to S. In Kubar, Bratko, and Michalski (1998) these errors are referred, respectively, as the error of omission and error of commission. The less the misclassification rates, the better the rule.

The rule should be as good as possible in terms of the misclassification rates. Yet one more requirement is that the rule should be interpretable in such terms that

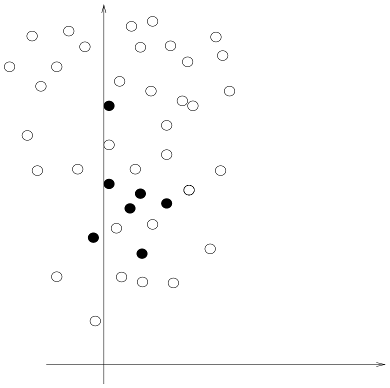

Figure 2: Black circles to be discriminated from the set of all/other circles.

the biologists could use it for further advancements in understanding the nature of proteins.

Of a number of machine learning approaches developed so far, let us discuss in the subsequent three subsections those most popular: (1) discriminant functions, (2) neural networks, and (3) conceptual descriptions.

4.2 Discriminant function

This is an intensional construction in the feature space R^n: a function $G(x)$, $x \in R^n$, is referred to as a discriminant function (separating surface) for a subset $S \subset M$ if $G(y_i) \geq \pi > 0$ for all $i \in S$ while $G(y_i) < \pi$ for all $i \in M - S$, where π is a threshold. Here, y_i are the feature space representations of the proteins $i \in M$.

Usually, the discriminant function is a hyperplane $G(x) = \sum_k c_k x_k$ where x_k are components of x. Linear functions can separate only convex sets, which relates the theory of discriminant hyperplanes to the theory of convex sets and functions.

Such a hyperplane is shown on Fig. 3. Obviously, the solution is too simplistic to be used in practical calculations for general fold recognition.

The theory of discriminant functions, developed by R. Fisher before the World War II, was initially a part of the mathematical multivariate statistics heavily loaded with probabilistic estimates, but currently it is moving into somewhat less rigid area of machine learning to involve more heuristical approaches for transformation of the variables (for a review and references, see Hand, 1997).

There exists an approach related to finding convenient transformation rules, formerly called the *potential function* method, which is quite general and, at the same time, reducible to linearity. The potential function $\psi(x,y)$ (currently, *kernel*) reflects similarity between x and y and, usually, is considered a function of the squared

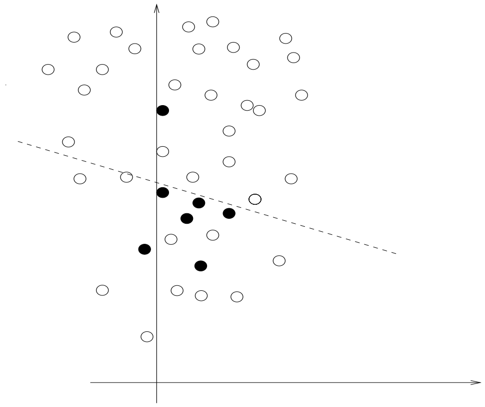

Figure 3: A linear discriminant function admitting one false negative and many false positives.

Euclidean distance between x and y such as $\psi(x,y) = 1/(1+ad^2(x,y))$ or $\psi(x,y) = exp(-ad^2(x,y))$ where a is a positive constant.

The potential discriminant function for a class $S \subseteq M$ is defined then as the average potential with respect to the points of S as the prototype points: $G_S(x) = \sum_{i \in S} \psi(x,y_i)/|S|$. The class of such potential functions is quite rich. It appears, for instance, that using $\psi(x,y) = exp(-ad^2(x,y))$ with sufficiently large a as the potential function in $G_S(x)$, the function $G_S(x)$ can separate any S from $M-S$ Andrews (1972). This shows that the approach is theoretically justified as applicable to any set learning problem.

This approach is related with appropriate transformations of the feature space, as follows. Potential functions depending on x and y through Euclidean distance between them, can be represented as $\psi(x,y) = \sum_p \lambda_p^2 g_p(x) g_p(y)$ where $\{g_p(x)\}$ is a set of the so-called eigen-functions. This allows transforming the classification problem into a so-called "straightening space" based on the transformed variables $z_p = \lambda_p g_p(x)$. In this straightening space, the potential function becomes the scalar product, $\psi(x,y) = (z(x), z(y))$, which makes all the constructions linear. This approach is being currently modified to the format of machine learning in terms of the so-called support vectors that serve as entities "modelling" the discriminant plane Schlkopf et al. (1998), Vapnik (1998). It seems that eventually, when more is known of the relevant feature transformations and computationally effective methods, the approach should be explored in the framework of the general fold learning.

4.3 Neural networks as a learning tool

Artificial neural networks provide for an extremely effective learning framework.

A formal neuron is a model for the neuron, a nerve cell working like an information-transforming unit. The neuron provides a transformation of an input vector $x = (x_k)$ into the output signal, $y = \theta(\sum_k c_k x_k - \pi)$, where $\theta(v) = 1$ if $v > 0$ and $\theta(v) = 0$ if $v \leq 0$. Actually, the neuron discriminates between two half-spaces separated by the hyperplane $\sum_k c_k x_k = \pi$. Frequently, the threshold output function θ is substituted by the so-called *sigmoid* function $\theta(v) = 1/(1+e^{-v})$ which is analogous to the threshold function but is smooth and more suitable for mathematical derivations. An interpretation of the formal neuron: the components k represent synapses excited on the level x_k; the weight c_k shows relative importance of the synapse to the neuron; the neuron fires output if the total charge $\sum_k c_k x_k$ is higher than the neuron threshold π.

Sometimes the neuron is considered to have the identity output function $\theta(v) = v$ thus performing just linear transformation $\sum_k c_k x_k$; this is called *linear neuron.*

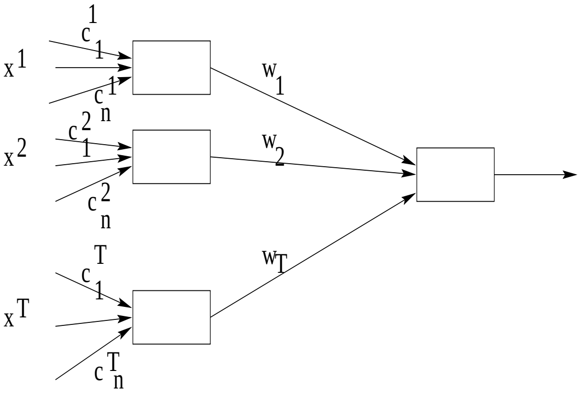

Figure 4: A neural network with a hidden layer.

A single hidden layer neural net (see Fig.4) defines a double transformation, $\theta[\sum_{t=1}^{T} w_t \theta_t]$ where $\theta_t = \theta(\sum_k c^t_k x^t_k - \pi_t)$, of the input vectors x^t, $t = 1, ...T$, into an output through T "hidden" neurons. Such a net has two important properties: 1) it can be used to approximate any continuous function; 2) it can be used to separate any subset S of a given set of normed vectors y_i, $i \in M$. To resolve the latter problem, let us take $T = |M|$ to consider any $t = 1, ..., T$ as a corresponding element of M; then, for any neuron $t \in M$, let its weight vector $c^t = y_t$. Obviously, the maximum of the scalar products (c_t, y_i) with regard to $i \in M$, in this case, is reached for the only $i = t$. Thus, fixing π_t between this maximum value and the second maximum, we have $\phi(t, y_i) = \theta(\sum_k c^t_k y^i_k - \pi_t) = 1$ if and only if $i = t$; $\phi(t, y_i) = 0$ when $i \neq t$. Then, taking $w_t = 1$ for $t \in S$ and $w_t = -1$ for $t \in M - S$, we get the desired output.

The first neuron-like learning algorithm, the *perceptron*, was proposed by F. Rosenblatt (see Nilsson, 1965) to learn a linear separating surface when S can be linearly separated. The perceptron perceives the entity points coming in sequence, starting from an arbitrary coefficient vector c and changing it after every try with the following rule: add to c (respectively, subtract from c) all erroneously classified points from S (respectively, from $M - S$), thus turning c toward a direction between the summary points of S and $M - S$. This guarantees that the method converges.

In a multilayer perceptron, a similar learning idea requires sequential weight changes layer-by-layer starting from the output layer (back-propagating). The *back-*

propagation learning process proposed by Rumelhart, Hilton and Wilson (1986) is, actually, a version of the method of steepest descent (a.k.a. hill-climbing) in the theory of minimization as applied to the square-error criterion $E(c) = \sum_i (x_i - d_i)^2$ where x_i and d_i are actual and ideal (shown by the "teacher") outputs of the neuron network, respectively. Let the output of the p-th neuron in a layer equal $x_{pi}(c) = \theta_p((c_p, y_i))$, where y_i is the input to the neuron from the preceding layer when the input to the network is i-th given point. Change of the weight vector c_p in the neuron is controlled by the equation $\Delta_i c_p = \alpha \delta_{pi} y_i$ where α is the step size factor (usually, constant) and $\delta_{pi} = -\theta'_p((c_p, y_i))(x_i - d_i)$ if this is the output layer, or $\delta_{pi} = -\theta'_p((c_p, y_i)) \sum_q \delta_{qi} c_{qi}$ for a hidden layer where q represents the next (more close to the output) layer's suffix.

Similar ideas, involving though different objective functions and propagation formulas, are put in the so-called Kohonen networks approach (see Kohonen, 1995).

As any local optimization method, in a particular computational environment, the back-propagation method can converge to any local optimum or even not converge at all, which does not hamper its great popularity.

However, in the protein fold recognition problem, this method suffers of a major draw-back: the solution is quite difficult to interpret because the network coefficients are not that easy to express in operational terms related to proteins.

4.4 Conceptual description

The most popular conceptual description tool, the regression or classification or decision tree, divides the entity set M in a tree-like manner, each division done by dividing a feature's range into two or more categories (for a review and references, see Salzberg, 1998). For instance, Fig. 5 presents a decision tree found with initially partitioning the vertical axe (feature y) in two intervals, A and B, and then by partitioning A into $a1$ and $a2$ along the same axe while dividing B into $b1$ and $b2$ along the horisontal axe (feature x). The four regions of the space have simple conceptual descriptions such as " $a2$ of y" or "B of y and $b1$ of x". The separating rule defined by this tree is obvious: black circles are in " $a2$ of y", and white circles are in other regions, as shown in Fig. 5. Each of the regions provides for one false positive.

Deciding on which partition and by which of the variables to do is based on a scoring function that measures the degree of uniformity of the resulting regions with regard to S and $M - S$ as shown in Fig. 5. Such a measure is provided in most popular programs, C4.5 by Quinlan (1993) and CART (see Breiman et al., 1984), by the entropy or index Gini, respectively; the program OC1 in Salzberg (1998) allows for any of seven goodness measues. These measures take into account both, the group to separate, S, and the rest, $M - S$, as equally important so that the tree is designed to maximize overall accuracy of the tree. This may lead to somewhat skewed results since the trees "tend to optimize accuracy on the larger class" (Salzberg, 1998, p. 190), which is especially important in the problem under consideration where S typically is smaller than $M - S$.

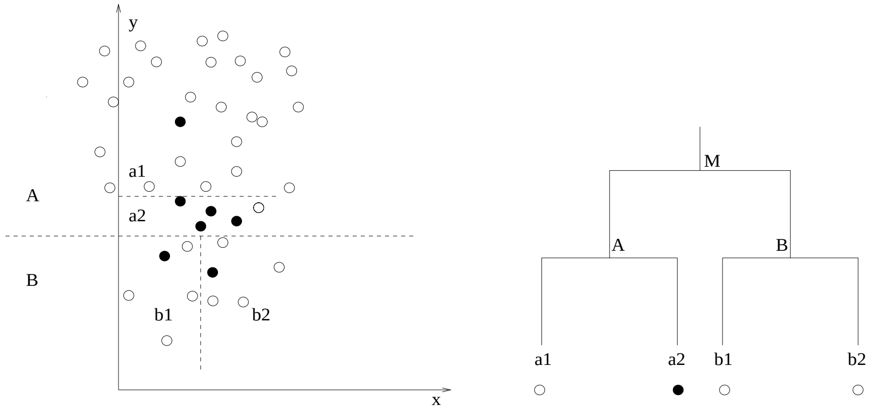

Figure 5: Regression tree shown both ways, in the feature space (on the left) and as a decision tree (on the right).

The conceptual description techniques developed so far do not pay much attention to the cases like ours. We are interested in getting a description for the S only: $M - S$ can be very much nonhomogeneous and its conceptual description may have no meaning at all! For instance, a somewhat better decision tree in Fig. 5 could have been found if the second split of the axe y ($a1$ and the rest) had been the first one to collect as many as possible of the black circles in the same region. Some of the recent data mining techniques do cover our case as those developed in Klösgen (1996), Srikant, Vu and Agraval (1997), Kubar, Bratko and Michalski (1998); however, the algorithms proposed in these publications such as Explora in the latter reference do not come out with a description of S: they are oriented just on finding conceptually described potentially overlapping subsets of S, however many of them and how complex the overlaps are.

In our view, most relevant to our problem would be getting a conceptual description such as that presented in Fig. 6. The black circle set, according to this description, is in the rectangle $a \leq x \leq b$ & $c \leq y \leq d$. The errors, according to this separating rule, are one false negative and two false positives.

4.5 Approximate conjunctive description

An algorithm for finding a conceptual description of S in M without getting a description of $M - S$, as just a conjunction of feature intervals, has been described in Mirkin (1999). This algorithm follows a traditional idea in data mining: the larger the difference between within-S average of a feature and its grand mean, the more important the feature is for separating S (see Klösgen, 1996). Moreover, in Mirkin (1999), it is shown that a proper measure of difference is the difference squared, because it is an additive item of the Pythagorean decomposition of the data scatter into the part explained by the group S and the unexplained part, thus bearing a proportion of the overall data variance.

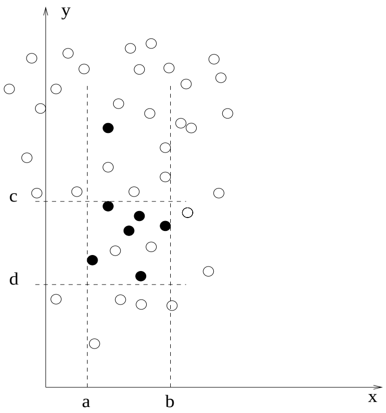

Figure 6: Conjunctive Description: 1 False Negative, 2 False Positives.

However, the difference squared may be not compatible with the criterion of accuracy of the description (minimum of errors), which implies that the simple ordering of the features according to decrease of their within-S and grand mean differences is not enough to get a good description. An algorithm suggested in Mirkin (1999) for finding a conceptual description of S within M uses both of the criteria, the differences and the errors, in the following way.

According to this procedure, a number of within-cluster-range-based terms is to be initially collected into a logical conjunction (first phase), after which redundant terms are excluded one-by-one (second phase). The first phase goes along the ordering of features according to their difference-between-averages squared, starting with the empty set of conjunctive terms. Any particular feature is considered along the ordering to decide whether or not it should be included in the conjunction. It is included only if this decreases the number of false positives. The process stops when there are no features left in the ordering or when the number of false positives becomes zero (or any other prespecified threshold value). The second phase goes in the opposite direction along the terms collected at the first phase to decide whether a single term under consideration can be removed from the collection or not. It is removed if its removal does not change the number of false positives. This method of consecutively applying of what is called forward search and backward search selection strategies in machine learning, builds an APProximate COnjunctive Description and thus will be called APPCOD-1 in the remainder.

APPCOD-1 performs rather well when the classes are located in different zones of the original feature space. The method works poorly in the domains where group S is spread over in the feature space so that it cannot be separated into that box-like cylinder volume which corresponds to an output conjunction.

However, the method's performance can be improved by transforming and combining the variables. To do this, denote by A the set of original features and B a set of features from A participating in the APPCOD-1 generated conjunctive description. Obviously, B is subject to stopping thresholds in APPCOD-1: the number of items in the resulting conjunction and the minimum error level admitted. Denote by S(A,B) the set of all pair-wise products, xy, ratios, x/y and y/x (the latter being denoted as $x\backslash y$), sums, $x+y$, and differences, $x-y$, for all $x \in$ A and $y \in$ B. Then iteratively perform APPCOD-1 on A, S(A,B), S(A, B(S(A,B)), etc. This way of iteratively combining the variables has two properties: first, it exploits the best APPCOD derived features for combining; second, it does not lose the original wealth of features since at each step all original features are involved in the combining process, too.

Such a process, called APPCOD-2, usually leads to drastic reduction of the number of false positives in the APPCOD-1 results.

To the stopping thresholds of APPCOD-1, APPCOD-2 adds one more threshold: a bound on the number of combining operations involved in a compound variable. The more operations, the more complex a variable. Thus, the same question emerges as in all other separation techniques: how complex a separation rule can be accepted by the learner? Potential remedies related to such "simplicity" principles as the minimum description length principle by Rissanen (1989) should be tried.

Still, both APPCOD-1 and APPCOD-2 may produce unsatisfactory descriptions when group S, on average, does not differ from the entire set M; that is, when no feature has its within-S average different from its grand mean. In such cases, other features should be tried or S can be divided into its "extreme parts" to be described separately.

5 Resampling as a Tool for Model Selection

Resampling is a procedure oriented at testing and improving reliability of data based estimators and rules. Generically, it is very simple: take a sample of the data set under consideration, apply your algorithm to the sample (training phase) and test its result on the rest (testing phase). Perform this many times and average the results. Quality of the average test results shows performance of the algorithm in question in changing environments. The better the average test results, the more reliable is the algorithm. The estimates and rules derived from random samples (with replacement) are called sometimes bootstrap estimates/rules Efron and Tibshirani (1993). Two particular schemes of resampling have become most popular: the k-fold cross-validation and leave-v-out cross-validation. In the k-fold cross-validation, the data set is divided into k subsets of (approximately) equal size. The algorithm is trained k times, each time leaving one of the subsets out of training, and using only the omitted subset to test. If k equals the sample size, this is called "leave-one-out" cross-validation. "Leave-v-out" is a more elaborate and expensive version of cross-validation that involves leaving out all possible subsets of v cases.

When the cross-validation is performed over just randomly generated subsets, it is frequently called the "split-sample" or "hold-out" method. Leave-one-out sampling scheme is frequently used for what is called jackknife estimating, when an estimate is calculated not just from the data set as usually, but from each of the subsets involved and then is averaged over the subsets. The difference between the jack-knife estimate and that standard one shows the bias of the latter.

Leave-one-out cross-validation often works well for continuous error functions such as the mean squared error, but it may perform poorly for discontinuous error functions such as the number of misclassified entities. In the latter case, k-fold cross-validation is usually preferred (with a value of 10 for k, typically).

To apply APPCOD to a sample, we need to solve an estimation problem: find left and right bounds of within-S feature intervals. When S is small, its samples are even smaller and they may and do give much biased estimates of the within-S feature ranges: in a sample from S, the left bound increases while the right bound decreases, which leads to a smaller within-S feature interval. As shows Fig. 7, where all circles represent set M, black circles its subgroup S, and the double black circles, a sample from S, the sample-based within-S interval (in this case, (c,d)) is always smaller than the entire-set based within-S interval (in this case, (a,b)).

Changes will also occur in the intervals of white, $M - S$, circles located to the right and to the left of the within-S interval.

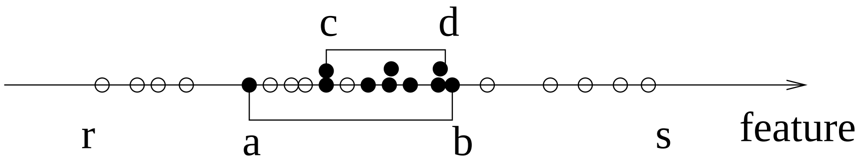

Figure 7: Decreasing the within-group interval under sampling: from (a,b) to (c,d).

This effect can be utilised for correcting the reduced sizes of the feature intervals by exploiting the points located between them or their quantiles. This is a problem that has not been yet properly addressed. In our algorithm, we use a somewhat rough estimate suggested by I. Muchnik (1998): the sample-based within-S interval is scaled with a constant factor (depending on the set and sample sizes) to make it up to the entire set estimate.

Since the number of misclassified entities may be different over different samples, resampling can be used not only for testing, but for learning, too: when a model or algorithm or parameters of an algorithm are selected to minimize the error over a number of computations. This is called model selection. Examples of model selection can be seen in Efron and Tibshirani (1993), p. 243-247 (selection of decision tree sizes), and Salzberg (1998), p. 193-194 (averaging probabilites obtained with split-sample-based decision trees).

Model selection is especially applicable to APPCOD found decision rules because of their simplicity: any rule is just a set of feature intervals. Yet implementation of this idea is not straightforward because of the stage of combining the features, APPCOD-2: different samples may and do lead to diferent compound features, which may effectively prevent comparing and, thus, averaging different APPCOD decision rules.

To overcome this hurdle, the process of learning via resampling can be divided into two phases. The first phase is to collect most effective combined variables over different samples by applying APPCOD-2 to them. These most effective combined variables are included then in the feature space, so that the initial set of features is supplemented with the set of most effective combined variables.

The updated feature set serves as an input to the second phase, which starts with multiple application of APPCOD-1 (no combined variables this time!) to samples from the updated data set. Then all APPCOD-1 found decision rules are comparable subsets of features from the same feature set.

To average these subsets, a majority rule is applied: a feature is put in the averaged decision set if and only if it occurs in more than 50% of the decision subsets. As seems to be well known, this majority rule, actually, follows the optimality principle for a related reconciling problem: for a given set of subsets $A_k \subseteq B$ $(k = 1, ..., n)$, find such an $A \subseteq B$ that minimizes $\sum_{k=1}^{n} d(A, A_k)$. The distance $d(A, A_k)$ here is just the number of features that belong to either A or A_k but not both. As can be easily proven, the majority rule generates a global solution to the reconciling problem.

Still one more step is to be performed: averaging intervals of the majority-selected features by averaging their left and right bounds over all occurrences of the features in the last series of APPCOD-1 found solutions.

This two-phase process involving both APPCOD-2 and APPCOD-1 will be referred to as APPCOD-3.

The two latest steps in APPCOD-3 (averaging feature subsets and averaging interval bounds) can be done over not all of APPCOD-1 generated solutions, but only with regard to those of them that have shown a better performance. For instance, in one of APPCOD-3 versions, the program does twenty APPCOD-1 computations with the updated feature set, and then excludes the worst ten of the results, thus leaving only best ten solutions for further averaging, which greatly improves the error rate in the final conjunction.

One more comment about the presented two-phase implementation of the resampling based model selection is that at each of the phases, the best solutions are selected according to an error rate criterion. These are selecting the best combined features, at the first phase, and selecting the best descriptions, at the second phase. To define what is the best, we need to specify a scoring function as a weighted sum of the number of false positives and the number of false negatives. The weights are compromise factors of the two items. The larger the factor at, say, the number of false negatives, the more important this part of the criterion is in the selected solution. Also, when sizes of S and $M-S$ are much different, the relative numbers, proportions of false positives and false negatives, can be taken to make these two comparable.

6 Feature Refining Approach

With the scale of protein sequence length measured per cent, the number of distinguishable intervals of the scale is about 5050: 1 interval of length 100, 2 intervals of length 99, ..., and 100 intervals of length 1. Thus, each feature of amino acids may be enveloped at the sequence level into 5050 interval features (the averages along the intervals), which leads to about 30,000 sequence features in total (for we have six amino acid features to use as described in section 3.)

Although this size of the feature space is rather challenging at the present time, it is not only the computation time that prevents us from brute force search through the space as is: the feature weighting, the pillar of APPCOD computations, is easy to do as it involves comparing only two averages, the grand mean and within-S mean, per feature. More important an issue, in this case, is that abundance of variables may be misleading when the data in hand are as ambiguous as the data base in SCOP which lacks clear critera of classification. In such cases, typically, redundant features may get not well-grounded importance and lead to false conclusions. This is why we prefer here to search through the feature space by modelling the scientific approach of refining interesting features by exploring them in greater detail. This way, a considerable reduction of computation is achieved, too.

In the beginning, each feature is measured rather roughly along only three intervals covering 0-40, 60-100, and 30-70 of the sequence. The denotation a-b here refers to an interval starting at a% of the sequence and ending at b% of it. The sequences are considered long enough to use the percentage scale without much distortion. Given the six amino acid features, this makes an eighteen-dimensional feature space. If a distinctive description of S in this space is reached, the computation stops. Otherwise, if the APPCOD-3 based feature intervals give a too high misclassification rate, the features occurred in the description should be refined to get a look ot the situation in more detail. Let A be the set of features in the space to which APPCOD is applied and $B(A)$ denotes the set of features that occurred either in APPCOD-3 generated description or in formulas for combined features added at the first stage of APPCOD-3 to the original feature set A. Then each feature from $B(A)$ is to be supplemented in A by three features defined over detailed intervals. If an $f \in B(A)$ is measured over interval a-b, which can be denoted by (f, a, b), the three new features are defined as $(f, a, a + c)$, $(f, b - c, b)$ and $(f, a + c/2, b - c/2)$ where $c = (b - a)/2$, the half length of a-b. This way, new features added to A cover the starting, middle and ending halves of the interval a-b. The process of adding the three refined features to the feature space will be referred to as the *refining*. If, with A thus updated, APPCOD-3 produces nothing better in terms of the misclasssification rate, the process stops and the previous result is considered as the final one. Otherwise, with the APPCOD description of S improved, next iteration of the same feature refining process is executed.

The misclassification rate traditionally is defined as the total number of errors divided by the size of set M of all entities under consideration. This can be misleading when the size of group S is significantly smaller than that of $M - S$. If, for instance, there are 30 false positives and 20 false negatives when $|M| = 1000$, then the misclassification rate is (30+20)/1000=5% which correctly reflects the individual rates of errors when $|S| = 500$: the relative commission error, the number of false positives related to $|M - S|$, is 30/500=6%, and the relative omission error, the number of false negatives related to $|S|$, is 20/500=4%, which is in concordance to the total missclassification rate. However, when $|S| = 40$ so that $|M - S| = 960$, the individual error rates are 30/960=3.1%, for false positives, and 20/40=50%, for false negatives. In this situation, the total error rate, 5%, just hides the fact of unsatisfactory level of recognition of the entities from S. This is why we use the averaged individual rate,

$$a\frac{fn}{|S|} + (1 - a)\frac{fp}{|M - S|},$$

as the scoring (quality) function of the conjunctive descriptions. Here, fp and fn are the numbers of false positives and false negatives, respectively, and a is the coefficient of compromise between the individual rates. In all further examples, $a = 0.5$.

To cope with the increase of the size of the current feature space in the feature refining process, the features that have not been involved in updating the space for two or more consecutive iterations, can be removed from the space at all.

This process of iteratively applying APPCOD-3 to the refined feature spaces will be called APPCOD-4.

The APPCOD-4 resembles methods of processing images on the multiscale basis, when some parts are looked at in greater detail than others are. Recently, this area of research has received a great impetus in mathematical developments to make the image and signal processing algorithms more effective (see Starck, Bijaoui and Murtagh, 1998). Although genomic data bases yet have not as many entities as visual data, some analogues of the techniques described in Starck, Bijaoui and Murtagh (1998) may be applicable to multiscale processing genomic data, too.

7 Four Stages for APPCOD

Let us collect and put the four APPCOD stages described in previous sections, each on top of the preceding one, more formally. The input consists of a set of protein sequences, M, its subset S to be separated with a conjunctive description, and a feature set A represented by amino acid features and intervals of the sequence, over which the features are averaged.

APPCOD-1: Finding a conjunctive description

Step 0. (Initial Setting) Set prior constraints on the maximum number of terms and the number of false positives admitted.

Phase 1. (Forward Feature Space Search) Collecting COD, within-S feature ranges, in a conjunctive description according to the feature weight ordering with skipping those not-decreasing the number of false positives.

Phase 2. (Backward Feature Space Search) Removing from COD those feature ranges that do not or least affect the number of false positives, to get the number of terms (feature ranges) less than or equal to the prespecified in Step 0 quantity.

APPCOD-2: Combining features for conjunctive description

Step 0. (Initial Setting) Set a prior constraint on the maximum number of operations involved in a combined variable.

Phase 1. (Feature Selection) Apply APPCOD-1.

Phase 2. (Combining Features) Combine the APPCOD-1 selected features with the original ones by using arithmetic operations and goto Phase 1.

APPCOD-3: Getting a reliable conjunctive description

Step 0. (Initial Setting) Set a prior proportion of the entities for random sampling, number $n1$ of the samples, compromise coefficients for numbers of false positives and false negatives, and number $n2$ of voting descriptions.

Phase 1. (Producing Descriptions) Apply APPCOD-2 to each of $n1$ samples.

Phase 2. (Updating the Feature Space) Add to A combined features (if any) from the best of $n1$ descriptions at Phase 1.

Phase 3. (Finding Averaged Description) Apply APPCOD-1 to the best $n2$ samples from the updated data set and take the averaged description (majority features with averaged intervals).

APPCOD-4: Multiscale refining search through protein features.

Step 0. (Initial Setting) Set the original feature space A: a number of amino acid features averaged over three rough intervals, 0-40, 30-70, and 60-100, covering all the sequence length. Put counter of the number of iterations, count-iter=1, and the error score, 100%.

Phase 1. (Producing Descriptions) Apply APPCOD-3 with A the space. Compare results with those kept from the preceding iteration. If the current error score is smaller, go to Phase 2. If not, end with the results of iteration number count-iter -1.

Phase 2. (Updating the Feature Space) Add to A features obtained by refining the features occurred in the conjunctive description or in combined features added to A in Phase 1. Remove features that have not been involved neither in resulting conjunctive descriptions nor in added combined variables through t consecutive iterations of the alforithm. Add 1 to count-iter and go to Phase 1.

In further experiments we, typically, set the description complexity (maximum number of the items in a conjunction) equal to 3, the feature complexity (the maximum number of arithmetic operations involved in a compound variable) equal to 1, while maintaining the sample size (in APPCOD-3) on the level of 40% of the entire set and the interval extension factor (also in APPCOD-3) on the level of 55%.

8 Examples of Protein Groups Described

In this section, results of the following three experiments will be presented:

1. Comparison of the results on separating TIM-barrel from the root of SCOP according to a secondary-structure-based feature set;
2. Separating TIM-barrel and α/β-chains in our feature space;
3. Separating subgroups on the lower levels of SCOP.

Though far from a complete assessment of the approach, this may give some preliminary insights into its comparative advantages and shortcomings.

Although currently available protein data bases may contain a significant part of all proteins in living organisms, their contents reflect rather interests of substantive researchers in genomics than the distribution of proteins in life and, in this sense, are biased: some proteins are overrepresented with their much similar multiple versions while the others are underrepresented or not presented at all. This is why most research in bioinformatics is done via so-called non-redundant data bases. A non-redundant data base contains only mutually different proteins: each two have no more than, for instance, 30% (or, 40%) of the sequence identity for the aligned subsequences longer than, for instance, 80 residues. A non-redundant data base, thus, has only one copy of each (sub)family of similar protein sequences, which yeilds one more feature: the cardinality of the non-redundant data base is relatively small amounting to just several hundred, not many thousand as in an original data base, entities.

Such a non-redundant data base was built by Dubchak et al. (1999) to test their approach in recognition of 128 SCOP folding classes. One of the feature spaces designed in Dubchak et al. (1999) to reflect the distribution of the secondary structure elements (helix, strand and coil) along the amino acid primary sequences (as reviewed in section 3), has been used in our study. We restricted ourselves to only one (from 128) folding class considered in Dubchak et al. (1999), TIM-barrel (see Fig. 1), for it is the most numerous in the non-redundant data base of SCOP folding classes (24 entities against the entire set of 516 proteins; in the original data base studied in 1996, the number of proteins was 607 (29, TIM-barrel), but 91 was then removed as corresponding protein structures disappeared from SCOP in 1998). This nonredundant data base is used in most of our experiments.

8.1 TIM-barrel in a secondary-structure-based data set

The neural network based prediction results for TIM-barrel reported in Dubchak et al. (1999): 31% false negatives and 19.8% false positives. These results also involve several feature spaces and a voting rule, in prediction, as well as the bootstrap testing.

In our computations, only the space defined by the categorisation of positions according to the type of the secondary structure element they belong to (helix, strand, or coil) as defined in section 3 is considered.

The results found with the feature complexity equal to 0 (only original features are present) are as follows. The majority averaged description found in a run of APPCOD-3 is this: $8.83 \leq \text{comS} \leq 21.21$ & $59.85 \leq \text{quaH} \leq 87.10$, where comS is the percentage of strand positions in the sequence and quaH is the length of the sequence, per cent, before 25% (quantile) occurrence of an alpha-helix position. The misclassification rates are: 21.2 % of false positives and 8.3 % of false negatives. Another run of APPCOD-3 yeilded the same majority features with slightly tightened bounds: $8.96 \leq \text{comS} \leq 20.82$ & $60.99 \leq \text{quaH} \leq 86.43$. This reduced the rate of false positives to 19.9 % without increasing the rate of false negatives, 8.3 %. One more run of APPCOD-3 has led to increasing the number of items in the majority conjunction to four (though only three items have been permitted in the individual descriptions) by adding features thrH (the length of the sequence, per cent, before 75% (quantile) occurrence of an alpha-helix position) and firC (the length of the sequence, per cent, before the first occurrence of a coiled position) to the two already appeared. The description this time is: $73.84 \leq \text{thrH} \leq 100$ & $36.56 \leq \text{firC} \leq 71.28$ & $7.44 \leq \text{comS} \leq 23.38$ & $59.36 \leq \text{quaH} \leq 89.43$. The accuracy of this description is: 18.2 % of false positives and 4.2% of false negatives. This outperforms the results for TIM-barrel reported in Dubchak et al. (1999). The accuracy can be further enhanced by making the bounds (at least for the latter two features) tighten. By reducing the bounds for comS and quaH to those found at the second run, we could have achieved zero false negatives and less than fifteen per cent of the false positives.

The majority conjunction found when one arithmetic operation in combining features has been admitted is this: 54.57 $\leq$ quaS - firS $\leq$ 82.12 & 0.11 $\leq$ traH/comC $\leq$ 0.36 & 0 $\leq$ traC/firS $\leq$0.012. Here, firS and quaS are, respectively, the lengths of sequences, per cent, before the first and 25% occurences of strand positions; traH and traC are percentages of transitions between helix and coil and between strand and coil, respectively; and comC is the percentage of coiled positions in the sequence. The first combined variable has an obvious meaning of the relative length of the interval between first and 25% occurrences of strand positions. The meaning of other combined features should be examined by the specialists. The accuracy of the description is 15 % of false positives and 8.3 % of false negatives.

What is nice about these descriptions is that they do not require huge numbers of items or operations that are typically needed in neural networks to get good results.

8.2 TIM-barrel and α/β proteins in the feature space

The APPCOD-3 program applied to the nonredundant set of 516 proteins as M and 24 TIM-barrel proteins within as S, with the eighteen interval features (6 amino acid variables averaged over the three rough intervals), produced the following majority conjunction: $0.02 \leq PG(60,100)*HY(60,100) \leq 1.28$ & $14.97 \leq MW(0,40)/PG(60,100) \leq 23.38$ whose misclassification rates are: 169 false positives (error of commission 34.3%) and 1 false negative (error of omission 4.2%).

By refining the four interval features involved, twelve new variables were added, leading to an improved description: $0.04 \leq PG(70,90)/MW(50,70) \leq 0.07$ & $0.92 \leq MW(0,20)/MW(30,50) \leq 1.06$ & $127.22 \leq MW(40,60) + BU(60,100) \leq 143.34$, with the number of false positives equal to 117 (error of commission 23.8%) and the number of false negatives, 2 (error of omission 8.3 %).

By refining the variables involved with simultaneously removing the features that have not been involved in the formulas in neither case, the space dimension has been upgraded to 34 to yield an upgraded majority description:
$822.27 \leq MW(30,40) * PG(75,85) \leq 1097.58$ & $0.94 \leq MW(0,20)/MW(30,50) \leq 1.06$ & $116.43 \leq MW\ (40,50) \leq 130.16$.
This description involves mostly molecular weight on 10% and 20% intervals and admits 73 false positives (error of commission 14.8%) and 4 false negatives (error of omission 16.7%). No further refining improves this description whose quality is comparable with that reached in the space of the secondary-structure-related variables (in the previous subsection).

However, things become more murky when we move on to treating the set of all α/β protein domains. This set consists of 143 specimens in the nonredundant data set and it has been considered in either capacity, as M (with regard to TIM-barrel as S) or as S (with regard to the entire protein domain set).

The best description of α/β class in the set of all 516 protein domains via APPCOD-4 has been produced on just second step (with intervals of 20% length) and accounts for 216 false positives (error of commission 57.9 %) and 3 false negatives (error of omission 2.1%). The enormous misclassification rate suggests that the subset of α/β protein domains is not homogeneous in the feature space. Thus, either the space should be changed or the APPCOD approach upgraded to deal with nonhomogeneous subsets by priorly partitioning them into homogeneous chunks.

Somewhat better results have been found at describing 24 TIM-barrel sequences within 143 α/β sequences. APPCOD-4 has produced the following majority description: $15.48 \leq PZ(30,70)/MB(60,100) \leq 43.83$ & $-6.21 \leq MW(60,80) - MW(50,70) \leq 5.72$ & $116.37 \leq MW(30,50) \leq 129.25$ & $6.63 \leq PG(70,90) \leq 8.37$ with 32 false positives (error of commission 26.9%) and 3 false negatives (error of omission 12.5%). This description involves features that are similar to those in the description of TIM-barrel within the entire protein domain set, above, which indicates again on the idea that the α/β sequences are dispersed over the entire data set in the space.

8.3 Treating lower layers of SCOP

We consider here two problems: description of α-Amylase (36 entities) as a subgroup of the TIM-barrel proteins in SCOP (512 entities) and description of all $\alpha-$ and $\beta-$ haemoglobin chains (178 entities) within the class of all globin-like proteins (359 entities). The haemoglobin chain subgroup combines all subfamilies with codes 1.1.1.1.16 through 1.1.1.1.32 in SCOP. This experiment also can be considered as testing APPCOD against the bias in protein contents of the contemporary databases. We hope that the bias does not much affect the results since APPCOD is based on rather robust functions of samples such as the averages and proportions.

Alpha-Amylase within TIM-barrel

Starting with the rough three intervals, APPCOD-4 has produced the following averaged description: $1.60 \leq HY(0,40) * PZ(0,40) \leq 3.25$ & $0.09 \leq HY(0,40) * MW(60,100) \leq 0.12$. The accuracy of the conjunction is: 23 (4.8%) false positives and 0 false negatives.

After upgrading the features involved in the description by dividing the intervals in three parts (for instance, MW(60,100) has been upgraded into MW(60,80), MW(60,80), and MW(70,90)), the resulting description is: $118.02 \leq MW(60,100) - PZ(0,20) \leq 125.88$ & $1.36 \leq PZ(0,20) * HY(0,40) \leq 2.85$ & $0.18 \leq HY(0,40) \leq 0.25$, which is more balanced and exact. There are 8 false positives (1.7%) and 1 false negative (2.8%).

In the nonredundant data set, there are 24 TIM-barrel proteins of which 4 are α-amylase. Applied to this data set, APPCOD-4 leads to description $0.18 \leq HY(0,40) \leq 0.21$ that admits no false positives and just 1 (25 % though) false negative. This latter description does involve a feature from the above, all-of-SCOP based, description, but it is much more rough than that above.

Haemoglobins in Globin-Likes

The group of 178 hamoeglobin α- and β-chains within the globin-like folding class is characterized by APPCOD-4 with the following description: $7.56 \leq PG(60,100) \leq 8.05$ & $0.116 \leq BU(60,100)/MW(30,70) \leq 0.127$. This seems rather accurate: no false positives and 2 (1.1%) of false negatives. The meaning of the variables is not clear, especially with regard to the fact that they relate to different parts of the sequence.

9 Directions for Work

Traditionally, learning of protein folds is performed with regard to their physical properties expressed in terms of the energy and alignment with homologues. In this paper, an original approach to learning of protein classes has been described as based on different learning strategy, approximate conjunctive description, and different feature spaces, amino acid properties averaged along intervals. This approach appeals to the attractive idea of describing protein groups in a biologically meaningful way. The experimental results show that this strategy is promising and should be further explored.

There are two immediate directions for further developments: (1) enhancement of the APPCOD method and (2) creation and maintaining of a set of descriptions of SCOP classes in such a way that it may become a tool for further spatial and functional analyses.

In the first direction, more restricted analytical tools should be implemented to allow not all potential combinations of the features but only those regarded as appropriate by the user. This can be important when descriptions of other groups are to be taken into account. In particular, the hierarchical structure of SCOP should be incorporated in descriptions so that the APPCOD description of a group S within M could exploit the results of description of a larger group, S' within the same M ($S \subset S'$).

On the more technical side, the approach should involve more attention to compatibility of such criteria as the misclassification rate, simplicity of the description, the debth of scaling, etc. Care should be taken to overcome potential nonhomogeneity of an S versus M by dividing S into separate pieces getting potentially different descriptions.

As to the second direction, the major issue seems to be in creating and maintaining a library of descriptions of all groups in SCOP. Such a library may serve not only as a classification device, but also in explaining the features involved with other feature spaces, for instance, with those related to secondary structure or contact maps. Since the APPCOD descriptions are logical and simple, this may lead to deapening of automatisation in learning structural and functional regularities in proteonomics.

References

Andrews, H.C. (1972) Introduction to Mathematical Techniques in Pattern Recognition, New York, Wiley-Interscience.

Appel R.D., Bairoch A. & Hochstrasser D.F. (1994) A new generation of information retrieval tools for biologists: the example of the ExPASy WWW server. Trends Biochem. Sci. **19**, 258-260 (http://www.expasy.ch/cgi-bin/protscale.pl).

Bachinsky, A., Yargin A., Guseva, E., Kulichkov, V. & Nozolenko, L. (1997) A bank of protein family patterns for rapid identification of possible functions of amino acid sequences, Comput. Appl. Biosciences, **13**, 115-122.

Bairoch, A., Bucher, P. & Hoffmann K. (1996) The PROSITE database, its status in 1995, Nucleic Acids Research, **24**, 189-196. Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984) Classification and Regression Trees. Wadsworth Intl. Group.

Chou, K.-C., Liu, W.-M., Maggiora, G.M. & Zhang, C.-T. (1998) Prediction and classification of domain structural groups, Proteins, **31**, 97-103.

Dubchak, I., Muchnik, I., Holbrook, S.R. & Kim, S.-H. (1995) Prediction of protein folding class using global description of amino acid sequence, Proc. Natl. Acad. Sci. USA, **92**, 8700-8704.

Dubchak, I., Muchnik, I., Mayor, C. & Kim, S.-H. (1999) Recognition of a protein fold in the context of SCOP classification, Proteins, **35**, 401-407.

Efron, B. & Tibshirani, R.J. (1993) An Introduction to the Bootstrap, Chapman and Hall.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (Eds.) (1996) Advances in Knowledge Discovery and Data Mining. Menlo Park, Ca, AAAI Press/The MIT Press.

Finn, P., Muggleton, S., Page, D. & Srinivasan, A. (1998) Pharmacophore discovery using the Inductive Logic Programming system PROGOL, Machine Learning, **30**, 241- 270.

Fisher, D. & Eisenberg, D. (1996) Protein fold recognition using sequence derived predictions, Protein Science, **5**, 947-955.

Gibrat, J.-F., Madej, T. & Bryant, S.H. (1996) Surprising similarities in structure comparison, Current Opinion in Structural Biology, **6**, 377-385. Gracy, J. & Argos, P. (1996) Automated protein sequence database classification, Bioinformatics, **14**, 164-187.

Hand, D.J. (1997) Construction and Assessment of Classification Rules. London, Wiley.

Hobohm, U. & Sander, C. (1995) A sequence property approach to searching protein databases, Journal of Molecular Biology, **251**, 390-399.

Hubbard, T., Murzin, A., Brenner, S. & Chothia, C. (1995) SCOP: a structural classification of proteins database, Nucleic Acids Research, **25**, 236-239.

Jones, D.T. (1997) Progress in protein structure prediction, Current Opinion in Structural Biology, **7**, 377-387.

Kawashima, S., Ogata, H. & Kanehisa, M. (1998) AAindex: amino acid index database, http://www.genome.ad.jp/dbget/aaindex.html.

Klösgen, W. (1996) Explora: A multipattern and multistrategy discovery assistant, in "Advances in Knowledge Discovery and Data Mining" (Eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy), Menlo Park, Ca, AAAI Press/ The MIT Press, pp. 249-271.

Kohonen, T. (1995) Self-Organizing Maps. Berlin, Springer-Verlag.

Kubar, M., Bratko, I. & Michalski, R.S. (1998) A review of machine learning methods, in "Machine Learning and Data Mining: Methods and Applications" (Eds. R.S. Michalski, I. Bratko and M. Kubat), Wiley, pp. 3-70.

Levitt, M. (1997) Competitive assessment of protein fold recognition and alignment accuracy, Proteins (Suppl.), **1**, 92-104.

Mirkin, B. (1999) Concept learning and feature selection based on square-error clustering, Machine Learning, **35**, 25-40.

Muchnik, I. (1998) Sample-based intervals can be corrected by constant factor scaling, Personal communication.

Muggleton, S., King, R. & Sternberg, M. (1992) Protein secondary structure prediction using logic. Protein Engineering, **5**, 647-657.

Murzin, A.G. & Bateman, A. (1997) Distant homology recognition using structural classification of proteins, Proteins (Suppl.) **1** (1997) 105-112.

Nilsson, N.J. (1965) Learning Machines, New York, McGraw-Hill.

Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B. & Thornton, J.M. (1997) CATH – A Hierarchic Classification of Protein Domain Structures, Structure, **5**, 1093-1108.

ProtScale: Amino acid scale representation.
In: http://www.expasy.ch/ tools/#primary.

Quinlan, J.R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA.

Rissanen, J. (1989) Stochastic Complexity in Statistical Inquiry. Singapore, World Scientific Publishing Co.

Rumelhart, D.E., Hilton, G.E. & Wilson, R.J. (1986) Learning internal representation by error propagation. In: D.E. Rumelhart and J.L. McClelland (Eds.) Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Cambridge, MIT Press, pp. 318-362.

Salzberg, S.L. (1998) Decision trees and Markov chains for gene finding, in "Computational Methods in Molecular Biology" (Eds. S.L. Salzberg, D.B. Searls & S. Kasif), Amsterdam, Elsevier Science B.V., pp. 187-203.

Srikant, R., Vu, Q. & Agraval, R. (1997) Mining association rules with the item constraints, in "Proceedings of Third International Conference on Knowledge Discovery and Data Mining" (Eds. D. Heckerman, H. Manilla, D. Pregibon & R. Uthrusamy), Menlo Park, CA, AAAI Press, pp. 67-73.

Schlkopf, B., Burges, C. & Smola, A.J. (Eds.) (1998) Advances in Kernel Methods - Support Vector Learning, MIT Press.

Starck, J.L., Bijaoui, A. & Fionn Murtagh (1998) Image Processing and Data Analysis: The Multiscale Approach, Cambridge University Press.

Vapnik, V.N. (1998) Statistical Learning Theory, Wiley and Son.

Wei, L., Chang, J.T. & Altman, R.B. (1998) Statistical analysis of protein structures, in "Computational Methods in Molecular Biology" (Eds. S.L. Salzberg, D.B. Searls & S. Kasif), Amsterdam, Elsevier Science B.V., pp. pp. 207-225.

Westhead, D.R. & Thornton, J.R. (1998) Protein structure prediction, Current Opinion in Biotechnology, **9**, 383-389.

Zhang, B., Rychlewski, L., Pawlowski, K., Fetrow, J.S., Skolnick, J. & Godzik, A. (1999) From fold predictions to function predictions: Automation of functional site conservation analysis for functional genome predictions, Protein Science, **8**, 1104-1115.

Zhang, C.-T. & Zhang, R. (1998) A new criterion to classify globular proteins based on their secondary structure contents, Bioinformatics, **14**, 857-865.