K-Local hyperplane distance nearest neighbor algorithm and protein fold recognition

Article *in* Pattern Recognition and Image Analysis · January 2006

DOI: 10.1134/S1054661806010068

1 author:

Oleg Okun
Cognizant Solutions GmbH, Hamburg
71 PUBLICATIONS **1,191** CITATIONS

Protein Fold Recognition with K-Local Hyperplane Distance Nearest Neighbor Algorithm

Oleg Okun
Machine Vision Group
Infotech Oulu & Dept. of Electrical and Information Engineering
P.O. Box 4500, FIN-90014 University of Oulu, FINLAND
e-mail: oleg@ee.oulu.fi

ABSTRACT

This paper deals with protein structure analysis, which is useful for understanding function of proteins and therefore evolutionary relationships, since for proteins, function follows from form (shape). One of the basic approaches to structure analysis is protein fold recognition (protein fold is a 3-D pattern), which is applied when there is no significant sequence similarity between structurally similar proteins. It does not rely on sequence similarity and can be achieved with relevant features extracted from protein sequences. Given (numerical) features, one of the existing machine learning techniques can be then applied to learn and classify proteins represented by these features. In this paper, we experiment with the K-Local Hyperplane Distance Nearest Neighbor algorithm (HKNN) [12] applied to protein fold recognition. The goal is to compare it with other methods tested on a real-world dataset [3]. Two tasks are considered: 1) classification into four structural classes of proteins and 2) classification into 27 most populated protein folds composing these structural classes. Preliminary results demonstrate that HKNN can successfully compete with other methods (by both speed and accuracy) and thus encourage its further exploration.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology - *classifier design and evaluation.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Protein fold recognition, protein secondary structure, feature normalization, nearest neighbor, support vector machine, neural networks.

In *Proceedings of the Second European Workshop on Data Mining and Text Mining for Bioinformatics,* held in conjunction with ECML/PKDD in Pisa, Italy. 24 September 2004.

1. INTRODUCTION

Protein is an amino acid sequence. In bioinformatics, one of the current trends is to understand evolutionary relationships in terms of protein function. Two common approaches to identify protein function are sequence analysis and structure analysis. Sequence analysis is based on a comparison between unknown sequences and those whose function is already known: thus, it may be appealing and in fact, it appeared before structure analysis. However, many sequences resulting from the modern genome projects are not labeled, and some closely related sequences may not share the same function. In addition, proteins may have low sequence identity but their structure and, in many cases, function suggest a common evolutionary origin[1]. In this paper, we therefore turn to structure analysis, though it does not mean that it alone can currently provide a complete solution.

Protein fold recognition is structure discovery without relying on sequence similarity. Proteins are said to have a common fold if they have the same major secondary structure[2] in the same arrangement and with the same topology, whether or not they have a common evolutionary origin. The folding problem is central in molecular biology and it can be formulated as follows: given the primary structure (linear sequence of amino acids in a protein molecule) of a protein, how the final 3-D fold can be deduced from it. To be able to predict the protein structure from the amino-acid sequence would have tremendous impact in all of biotechnology and drug design.

The structural similarities of proteins in the same fold arise from the physicical and chemical properties favoring certain arrangements and topologies. As the gap widens between the number of known sequences and the number of experimentally determined 3-D protein structures (the ratio is more than 100 to 1, and sequence databases are doubling in size every year), the demand for automated fold recognition (or prediction) techniques rapidly grows.

Since it is difficult to learn the rules of protein folding from primary structure, methods of secondary structure prediction are gaining popularity. They employ physicochemical features, such as fold compactness or hydrophobicity.

Our goal here is to test one classifier, which is the modified nearest neighbor algorithm called the K-Local Hyperplane

[1]It is argued that sequence analysis is good at high levels of sequence identity, but below 50% identity it becomes less reliable.

[2]Regions of local regularity within a protein fold, e.g., α-helices, β-strands.

Distance Nearest Neighbor (HKNN) [12] when applied to protein fold recognition based on features derived from secondary structure. This classifier was selected, because it has already showed [12] a very good performance with regard to Support Vector Machine (SVM) [11], which is considered to be the state-of-the-art in pattern recognition. In addition, to our best knowledge, it was never applied to any bioinformatics task. As the pattern-recognition approach in bioinformatics made considerable progress, it is therefore natural to explore new, still untried techniques. We are aware that no single method can solve the task to be tackled. Therefore our intentions are to find out what the method we selected can yield, compared to other techniques when tested on one of the commonly used datasets [3].

Different folds can be grouped into the following structural classes [6]:

1. all-α (those whose structure is essentially formed by α-helices),
2. all-β (those whose structure is essentially formed by β-sheets),
3. α/β (those with α-helices and β-strands),
4. $\alpha+\beta$ (those in which α-helices and β-strands are largely segregated).

Hence, two tasks are considered: 1) classification into four structural classes and 2) classification into 27 most populated folds composing these structural classes.

The paper has the following structure. Section 2 describes the data used in experiments. Previous work involving these data is briefly surveyed in Section 3. Section 4 introduces HKNN, while experiments with it are provided in Section 5. Finally, Section 6 concludes the paper with discussion.

2. DATA

Two datasets derived from the SCOP (Structural Classification of Proteins) database [6] were used. These datasets are available on line[3] and their detailed description can be found in [3]. Each dataset contains the 27 most populated folds represented by seven or more proteins and corresponding to four major structural classes (α, β, α/β, and $\alpha+\beta$). Table 1 shows the protein distribution among these classes. As one can see, both datasets are slightly imbalanced i.e. some classes are better represented than others (the same is true for individual folds), which makes both class and fold recognition tasks somehow challenging.

Table 1: Distribution of the proteins among structural classes for two datasets

Structural class	Dataset 1	Dataset 2
α	55	61
β	109	117
α/β	115	145
$\alpha+\beta$	34	62
Total	313	385

Six features were extracted from protein sequences: amino acids composition (C), predicted secondary structure (S), hydrophobicity (H), normalized van der Waals volume (V), polarity (P), and polarizability (Z). All (except the first) features have dimensionality 21, whereas the composition has dimensionality 20. Thus, in total, a feature vector combining six features has 125 dimensions (or components). In addition, the protein length is reported for each protein fold. When it is included, the dimensionality of the composition rises to 21, while for other features it becomes 22. As a result, in total, a feature vector has the length of 131.

The first set consists of 313 protein folds having (for each two proteins) no more than 35% of the sequence identity for aligned subsequences longer than 80 residues. The second dataset of 385 folds, also known as *independent test dataset*, is composed of protein sequences of less than 40% identity with each other and less than 35% identity with the proteins of the first dataset. In fact, 90% of the proteins of the second dataset have less than 25% sequence identity with the proteins of the first dataset.

3. PREVIOUS WORK

All approaches, briefly analyzed below, use the datasets described in the previous section. Unless otherwise stated, a 125 dimensional feature vector is assumed for each protein fold. Since 27 protein folds form four structural classes, a two-level recognition can be done. At level 1 a protein to be recognized is assigned to one of the four classes, while at level 2 it is classified as one of the 27 folds. Recognition can include either level or both levels. In the latter case, classification can be hierarchical, i.e. classifiers at level 2 employ the outputs of level 1, which means that they are not trained to predict all folds, but only those belonging to a certain structural class.

3.1 Recognition of Structural Classes of Proteins

Shi and Suganthan [10] applied feature selection and scaling, based on the mutual information, prior to K-Nearest Neighbor (KNN) and SVM. Their results indicate that feature selection is beneficial for both methods, while feature scaling (normalization) boosts the performance of KNN, but it is rather useless for SVM.

3.2 Recognition of Protein Folds

Ding and Dubchak [3] employed ensembles of both three-layer feed-forward NNs (trained with the conjugate gradient method) and SVMs (one-vs-all, unique one-vs-all and one-vs-one methods for building multi-class SVMs). Each ensemble consisted of many two-class classifiers.

Bologna and Appel [1] used a 131 dimensional feature vector and an ensemble of four-layer Discretized Interpretable Multi-Layer Perceptrons (DIMLP), where each network learns all protein folds simultaneously, which is in contrast to [3]. Bagging and arcing combined the outputs of DIMLPs.

3.3 Recognition of Both Classes and Folds

Chung et al. [2] selected Neural Networks (NNs) and SVMs as basic building blocks of two-level classification. In contrary to [3], each NN or SVM is a multi-class classifier so that the number of classifiers is greatly reduced (actually, it is equal to five: one classifier for class recognition and the four for fold recognition). The common NN models with a single hidden layer were used: MLP, Radial Basis Function Network (RBFN), and General Regression Neural Network (GRNN).

[3]http://crd.lbl.gov/∼cding/protein

Huang et al. [5] exploited a similar approach by utilizing gated NNs (MLPs and RBFNs). Gating is used for on-line feature selection in order to reduce the number of features fed to a classifier. Gates are open for useful features and they are close for bad ones. First, the original data are used to train the gating network. At the end of the training, the gate function values for each feature indicate whether a particular feature is relevant or not by comparing these values against a threshold. Only the relevant features are then used to train a two-level classifier whose NNs at each level are trained independently of each other.

Pal and Chakraborty [9] performed independent classification at each level, i.e. they did not utilize a hierarchical classifier. They trained the MLPs and RBFNs with new features (400 in number) based on the hydrophobicity of the amino acids. In some cases, the 400 dimensional feature vectors led to a higher accuracy than when using the traditional (125 dimensional) ones.

4. K-LOCAL HYPERPLANE DISTANCE NEAREST NEIGHBOR ALGORITHM

HKNN [12] is a modified KNN algorithm intended to improve the classification performance of the conventional KNN to a level of SVM, which is considered by many as the state-of-the-art in pattern recognition. Unlike the SVM, which builds a (non-linear) decision surface, separating different classes of the data, in a high (often infinite) dimensional feature space, HKNN tries to find this surface directly in input space. If one thinks of each class as a low dimensional manifold embedded in a high dimensional space, it is quite reasonable to assume that this manifold is locally linear. Since the data are usually sparse, they leave "holes" in the manifold. These "holes" introduce artifacts in the decision surface generated by the conventional KNN, thus negatively affecting the generalization ability of this method. To remedy this deficiency, the idea of HKNN is to fantasize the missing points, based on a local linear approximation of the manifold of each class.

To reach its objective, HKNN computes distances of each test point x to L local hyperplanes, where L is the number of different classes. The ℓth hyperplane is composed of K nearest neighbors of x in the training set, belonging to the ℓth class. A test point x is associated with the class whose hyperplane is closest to x.

The ℓth local hyperplane can be expressed as

$$LH_\ell^K(x) = \left\{ p \,\middle|\, p = \overline{N} + \sum_{k=1}^{K} \alpha_k V_k,\ \alpha_{1\ldots K} \in \mathbb{R}^K \right\} \quad (1)$$

where N_k is the kth nearest neighbor of x, $\overline{N} = \frac{1}{K}\sum_{k=1}^{K} N_k$ is the centroid of a K-neighborhood, and $V_k = N_k - \overline{N}$.

$LH_\ell^K(x)$ defines a $K-1$ dimensional hyperplane, unless there are colinearities lowering a dimensionality.

To determine α_k in Eq. 1, one needs to solve the following linear system

$$(\mathbf{V}'\mathbf{V})\alpha = \mathbf{V}'(x - \overline{N}) \quad (2)$$

where x and $\overline{N}$ are n dimensional column vectors, $\alpha = (\alpha_1, \ldots, \alpha_K)'$, $\mathbf{V}$ is $n \times K$ matrix whose columns are the V_k vectors defined earlier.

To penalize large values of α, i.e. to penalize moving away from the centroid when the chosen value of K does not match well the locally linear approximation of the class manifold, a penalty term λ can be introduced. When it is included, the (square) distance of the point x to the hyperplane $LH_\ell^K(x)$ is defined as

$$d(x, LH_\ell^K(x))^2 = \left\| x - \overline{N} - \sum_{k=1}^{K} \alpha_k V_k \right\|^2 + \lambda \sum_{k=1}^{K} \alpha_k^2, \quad (3)$$

where α_k's are found by solving Eq. 4

$$(\mathbf{V}'\mathbf{V} + \lambda\mathbf{I})\alpha = \mathbf{V}'(x - \overline{N}) \quad (4)$$

where $\mathbf{I}$ is the $K \times K$ identity matrix.

Thus, HKNN has two parameters, K and λ, to be pretuned. It has been implemented in MATLAB and can be freely obtained (as well as auxiliary functions performing the normalization, plotting, etc.) from the author.

5. EXPERIMENTS

There are two options how to use the datasets introduced in Section 2. According to the first option, a classifier uses Dataset 1 for training and Dataset 2 (independent test set) for testing. Here we will follow this option. The second option suggests that Dataset 1 is used for K-fold cross-validation, where K is typically 4 to 10. Experiments with this option are reported elsewhere.

We scale the values of each component (dimension) in the feature vectors to have *zero mean* and *unit variance* according to the suggestion about the normalization given in [10][4]. Figures 1 and 2, showing the histograms and normal probability plots for individual components of the *composition* feature vector, confirm that this normalization is appropriate, since many histograms have a Gaussian-like shape and the respective normal probability plots are almost linear.

The Jarque-Bera (JB) test for normality[5] (MATLAB function *jbtest* implements it) at $\alpha = 5\%$ significance level, applied to each component of composition, revealed that the

[4]Unless otherwise stated, while talking about HKNN, we mean the normalized feature vectors.

[5]The JB test evaluates the hypothesis that the input data vector X has a normal distribution with *unspecified* mean and variance against the alternative that X does not follow a normal distribution. The test is based on the sample skewness and kurtosis of X. For a true normal distribution, the sample skewness should be near 0 and the sample kurtosis should be near 3. The JB test determines whether the sample skewness and kurtosis are unusually different than their expected values, as measured by a χ^2 statistic with two degrees of freedom.
The JB statistic is computed as follows:

$$JB = n\left(\frac{S^2}{6} + \frac{(K-3)^2}{24}\right) \sim \chi_2^2,$$

where n is the sample size, S and K stand for skewness and kurtosis, respectively, and they are estimated from the sample data. The JB test statistic has a χ^2 distribution with two degrees of freedom, and in practice, all that one needs to do is to compare the JB test value with the theoretical value of $\chi_2^2 \approx 5.9915$ at a specified level of significance. If the JB statistic is greater than χ_2^2, or alternatively, if the p-value is less than α, we reject the normality assumption. The JB test is more preferable than the Kolmogorov-Smirnov test, for which the parameters of a distribution should be prespecified, and the latter is not accurate if they have to be estimated from the data.

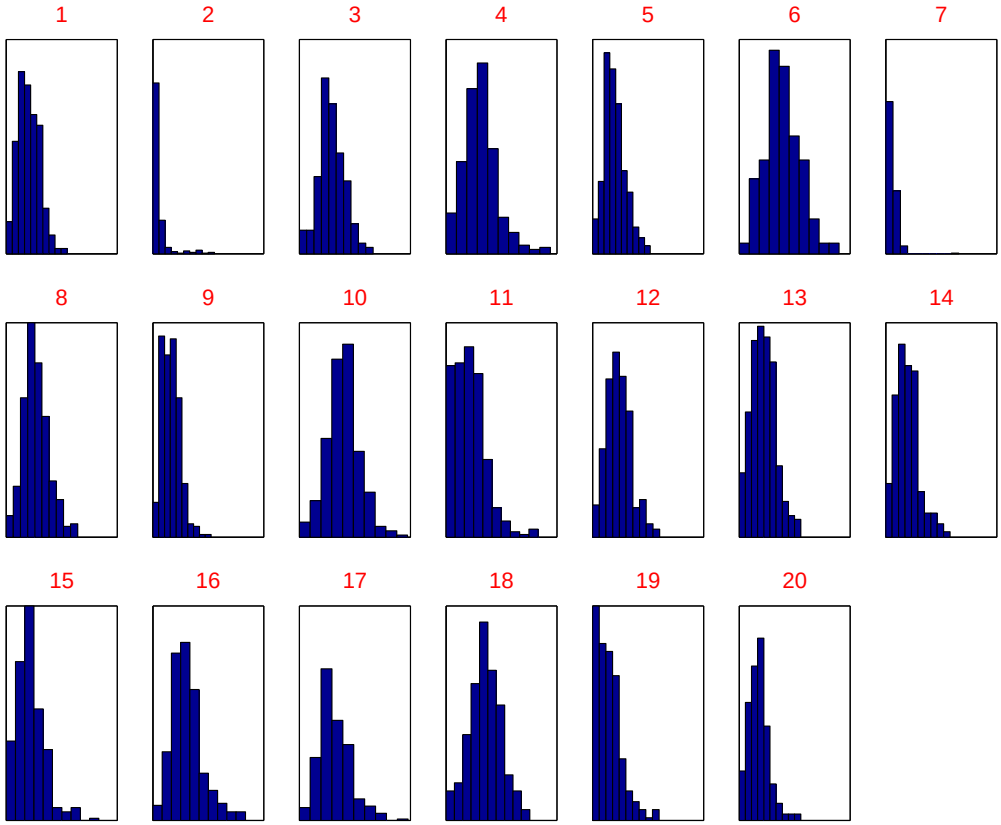

Figure 1: Composition: histograms of individual components

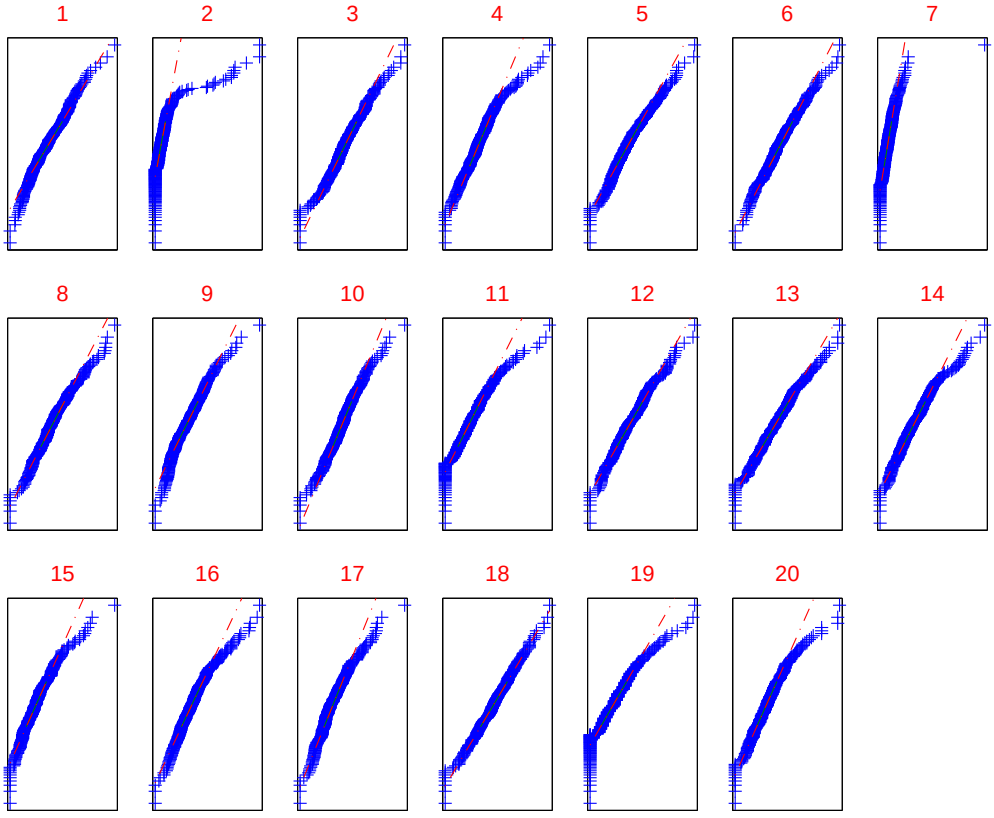

Figure 2: Composition: normal probability plots for individual components

2nd and 7th components are not normally distributed and this fact supports visual observations. Other than composition features have up to six components per feature vector, which fail to satisfy the JB test. However, removing these non-Gaussian components from the feature vectors is not advisable, since the accuracy is reduced.

We start experiments from level 1 of recognition, i.e. recognition of the four structural classes. Table 2 represents the results obtained with different methods on the independent test set (Dataset 2) when Dataset 1 served as the training set. The 125 dimensional feature vector associated with each protein was used as input to classifiers. For HKNN, the optimal values for K and λ were 7 and 8, respectively. It turned out that HKNN outperformed all other methods. At the same time, unlike the NN models, it did not need to tune many parameters.

Table 2: Best accuracy (in percents) for level 1 of protein recognition

[10]		[5]	[9]		our
SVM	KNN	gated RBFN	MLP	RBFN	HKNN
76.9	71.4	81.6	80.5	81.8	82.6

It was remarked in [3] that it is quite easy to achieve the accuracy over 70% at level 1 by using simple features, and all the classifiers reached this objective. However, recognizing the 27 folds is a more formidable task. First, we provide (in Table 3) the results for the case when a single-stage (non-hierarchical) classifier was trained on Dataset 1 and then tested on Dataset 2, i.e. the task was to predict folds without recognizing the structural classes. Again, the 125 dimensional feature vector was associated with each protein fold, except DIMLPs [1] that used the 131 dimensional vector. HKNN used both 125 and 131 dimensional vectors and the accuracy was the same in both cases as given in Table 3.

Table 3: Accuracy (in percents) for level 2 of protein recognition when a single-stage (non-hierarchical) classifier was utilized to label an unknown protein fold as belonging to one of the 27 folds

Paper	Method	Accuracy
[1]	DIMLP-B	61.2
[1]	DIMLP-A	59.1
[2]	MLP	48.8
[2]	GRNN	44.2
[2]	RBFN	49.4
[2]	SVM	51.4
[9]	RBFN	51.2
[3]	SVM	53.9
our	HKNN	57.4

Two rows for SVM/RBFN[6] in Table 3 correspond to results borrowed from different papers. In each case, particular details of SVM (kernel and its parameters) are not reported, except the fact that the ensemble of one-vs-one SVMs was used in [3]. Parameters of HKNN remained the same as above[7]. It is seen that HKNN again outperformed almost all competitors, except DIMLP-B (bagged DIMLP) and DIMLP-A (arced DIMLP). In the latter case, the ensemble of NNs can lead to a higher accuracy. In contrast, the results achieved with HKNN were obtained with a single classifier. When a single DIMLP was used in [1], the accuracy was only 38.2%. In addition, DIMLPs required many parameters to be tuned, while HKNN, in fact, has only two parameters to be set.

It should be noted that higher accuracies for SVM than provided in Table 3 were mentioned in [3] when using three subsets of features: 56.5% (CHPS), 56.0% (CHS) and 55.5% (CHPSV). We also ran HKNN for CHPS, CHS and CHPSV subsets of features and obtained 57.9%, 57.1% and 55.8% of the recognition accuracy, respectively, which confirmed the superiority of HKNN over SVM[8].

Next, we gather together results of hierarchical (two-level) classification in Table 4, where the final accuracy is only reported using the 125 dimensional feature vectors. Dataset 1 (Dataset 2) was used for training (testing).

Table 4: Accuracy (in percents) for hierarchical protein recognition

[2]				[5]
MLP	RBFN	GRNN	SVM	gated RBFN
44.7	56.4	45.2	53.8	56.4

Comparing these results with 57.4% for HKNN, one can observe that the hierarchical scheme was nevertheless unable to outperform a single classifier.

Finally, the accuracy rate of HKNN alone is given in Table 5 for various values of K and λ when 125 dimensional vectors were used. One can see that the accuracy is more rapidly changing along columns than along rows, i.e. it rather depends on K than on λ. Given a fixed K, the maximal difference between the top 1 and top 2 accuracies in the same row does not exceed 1% whereas it is 1.5-5% when varying K while fixing λ. Thus, a choice of proper K is important for reaching a higher accuracy. For the datasets used in experiments it might be good (though this is not a general recommendation) to set K to $\min(n_i)$, $i = 1, \ldots, L$, where min is the minimum function, n_i is the number of representatives of the ith fold in the training set, and L is the total number of folds. This rule can thus provides an upper bound on K. However, in general, it would be better to tune K rather than to set it to $\min(n_i)$, especially if the latter is large, e.g. 200, since the larger K, the more outliers have a chance to be included into a neighborhood and therefore to distort a hyperplane. On the other hand, too small K (few points) may be not sufficient to accurately build a hyperplane, either. Therefore a choice of K should be a compromise of several conditions.

For the original (without normalization) feature vectors, the accuracy is much more weakly affected by changes in either parameter of HKNN. It deviates between 47 and 48%. The smaller K, the smaller a difference in accuracy achieved when using the normalized and original features for a given λ. The maximal (minimal) difference is about 10% (4%).

[6]For the RBFN in [9], 400 features proposed there were fed to a classifier.

[7]In this test, the maximal possible value for K is 7, since many folds have only seven representatives in Dataset 1.

[8]Tests involving HKNN and different subsets including one to five features were also done and CHPZS led to the highest accuracy rate about 59%, while subsets including C, H and S features were among those corresponding to the top accuracies.

Table 5: Accuracy (in percents) for HKNN, depending on K and λ, for normalized features. Top accuracy in each row (K is fixed) is underlined

	$\lambda = 1$	$\lambda = 8$	$\lambda = 20$	$\lambda = 50$	$\lambda = 70$	$\lambda = 100$
$K = 7$	56.88	<u>57.40</u>	56.36	56.10	55.58	54.29
$K = 6$	55.58	<u>55.84</u>	<u>55.84</u>	55.06	54.55	53.51
$K = 5$	<u>54.29</u>	53.77	53.77	53.25	54.03	<u>54.29</u>
$K = 4$	54.03	54.29	<u>55.32</u>	55.06	<u>55.32</u>	55.06
$K = 3$	<u>52.47</u>	52.21	51.69	51.95	51.95	51.95

6. DISCUSSION AND CONCLUSION

In this paper, we investigated the K-Local Hyperplane Distance Nearest Neighbor algorithm, applied for protein fold recognition, and tested it on a real-world dataset from [3]. The obtained results are very encouraging, since like the conventional KNN, HKNN does not need training (though testing may take some time if the training set is large) and unlike the NN models and even SVMs, it has fewer adjustable parameters. Being a single-stage classifier, HKNN demonstrated strong performance, compared to the *ensembles* of SVMs and NNs. Though the accuracy achieved with HKNN is just 57.4%, it can be considered good by taking into account the fact the random accuracy would be 1/27≈3.7% as remarked in [3]. In addition, we are dealing with a relatively small dataset, where four misclassifications comprise 1% of error and there are only few representatives of many folds in the training set! Hence, this issue should be addressed in future research.

We also tried a hierarchical, two-level classification with HKNN employed at both levels, but it did not result in better accuracy. Though protein structural classes were relatively accurately recognized, the final accuracy fell to that achieved with the single-stage HKNN because of failures to correctly identify folds within classes, i.e. even if a class was correctly identified, it did not mean correct fold recognition. This fact can be easily explained, since the folds within each class and between different classes have a low similarity. Another reason for a failure of the hierarchical approach is that in this scheme if a classifier at level 1 makes a mistake, then it cannot be corrected at level 2. Perhaps, a combination of classifiers, one of which is HKNN, can yield better prediction, which will be explored in the future.

Interesting preliminary conclusions[9] can be, anyway, made about combining the classifiers analyzed in this paper, based on Table 6 (results for SVM are borrowed from [3]), comparing HKNN versus the ensemble of one-vs-one SVMs, Table 7 (results for DIMLP are taken from [1]), comparing HKNN versus the ensemble of DIMLPs, and Table 8 with values of Spearman r_S, t-statistics and p-value.

As one can see from Table 8, large r_S's (close to 1) and very small p-values (much less than α) tell us that there are quite strong relationships between results of all classifiers, that is, combinations of these classifiers might not lead to a (significantly) higher accuracy. At least for HKNN and SVM, statistical calculations are well supported by experiments. As advised in [12], we trained a nonlinear SVM, followed by HKNN (it is supposed in [12] that SVM acts as data reduction, since the original training set is replaced with a reduced set of support vectors, and HKNN therefore spends less time for classification of test data). It turned out that the accuracy achieved with this scheme was not better than that of HKNN performing alone. The reason of this behavior can be that almost every pattern became a support vector. Thus, SVM (and, moreover, DIMLP) cannot be a good candidate for coupling with HKNN.

[9]These conclusions are valid for the datasets tested in this paper and must be treated with caution when generalizing to other datasets.

Table 6: Accuracy (in percents) for individual folds and overall accuracy (bottom line) for 27 folds. CHS features are used

Fold index	SVM	HKNN
1	83.3	83.3
3	77.8	77.8
4	35.0	50.0
7	50.0	87.5
9	100.0	88.9
11	66.7	44.4
20	71.6	56.8
23	16.7	25.0
26	50.0	84.6
30	33.3	50.0
31	50.0	50.0
32	26.3	42.1
33	50.0	50.0
35	25.0	50.0
39	57.1	42.9
46	77.1	79.2
47	58.3	58.3
48	48.7	53.9
51	61.1	40.7
54	36.1	33.3
57	50.0	37.5
59	35.7	71.4
62	71.4	71.4
69	25.0	25.0
72	12.5	25.0
87	37.0	25.9
110	83.3	85.2
Avg	56.0	57.1

It turned out that some folds were much easier to classify than others. It can be because it has been observed that during evolution, the 3D structure of a protein is more conserved than its sequence. Thus, seemingly different sequences may have similar folds and this represented a challenge for HKNN.

On the other hand, as remarked in [7], different structure classification systems, e.g., such as SCOP (Structural Classification of Proteins) [6], FSSP (Families of Structurally Similar Proteins) [4], CATH (Class, Architecture, Topology, Homology)[8], have a quite low agreement on assigning folds to classes in uncertain cases. Hence, in the future, it would be fair to identify such cases and eliminate them from benchmarking.

7. ACKNOWLEDGMENTS

I wish to thank the anonymous reviewers for their valuable comments that helped to significantly improve the final paper.

Table 7: Accuracy (in percents) for individual folds and overall accuracy (bottom line) for 27 folds. 131D features are used

Fold index	HKNN	DIMLP-B	DIMLP-A
1	83.3	85.0	90.0
3	88.9	97.8	86.7
4	55.0	66.0	47.0
7	87.5	41.3	72.5
9	88.9	91.1	88.9
11	11.1	22.2	23.3
20	68.2	75.7	65.0
23	33.3	40.0	32.5
26	69.2	80.8	64.6
30	50.0	46.7	58.3
31	50.0	75.0	75.0
32	42.1	22.6	31.6
33	50.0	45.0	47.5
35	25.0	50.0	50.0
39	42.9	74.3	71.4
46	77.1	83.8	76.5
47	66.7	55.0	66.7
48	46.2	52.3	59.2
51	48.2	39.3	36.7
54	50.0	41.7	43.3
57	37.5	46.3	41.3
59	57.1	55.0	55.0
62	71.4	44.3	71.4
69	25.0	25.0	25.0
72	25.0	23.8	25.0
87	33.3	41.1	44.1
110	70.4	100.0	93.3
Avg	57.4	61.1	59.1

Table 8: Spearman r_S, t-statistics and p-values

Pairs of methods	r_S	t	p
DIMLP-B vs DIMLP-A	0.8640	8.5805	6.4003e-9
HKNN vs DIMLP-A	0.8407	7.7614	4.0626e-8
HKNN vs DIMLP-B	0.7248	5.2604	1.9029e-5
HKNN vs SVM	0.6769	4.5979	1.0565e-4

8. REFERENCES

[1] G. Bologna and R.D. Appel. A comparison study on protein fold recognition. In *Proc. of the 9th Int. Conf. on Neural Information Processing, Singapore*, pages 2492–2496, 2002.

[2] I.-F. Chung, C.-D. Huang, Y.-H. Shen, and C.-T. Lin. Recognition of structure classification of protein folding by NN and SVM hierarchical learning architecture. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Lecture Notes in Computer Science (Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Istanbul, Turkey)*, volume 2714, pages 1159–1167. Springer-Verlag, Berlin, 2003.

[3] C.H.Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[4] L. Holm and C. Sander. The FSSP database of structurally aligned protein fold families. *Nucleic Acids Research*, 22:3600–3609, 1994.

[5] C.-D. Huang, I.-F. Chung, N.R. Pal, and C.-T. Lin. Machine learning for multi-class protein fold classification based on neural networks with feature gating. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Lecture Notes in Computer Science (Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Istanbul, Turkey)*, volume 2714, pages 1168–1175. Springer-Verlag, Berlin, 2003.

[6] L. Lo Conte, B. Ailey, T.J.P. Hubbard, S.E. Brenner, A.G. Murzin, and C. Chotia. SCOP: a structural classification of proteins database. *Nucleic Acids Research*, 28(1):257–259, 2000.

[7] L. McGuffin, K. Bryson, and D.T. Jones. What are the baselines for protein fold recognition? *Bioinformatics*, 17(1):63–72, 2001.

[8] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH - a hierarchical classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[9] N.R. Pal and D. Chakraborty. Some new features for protein fold recognition. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Lecture Notes in Computer Science (Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Istanbul, Turkey)*, volume 2714, pages 1176–1183. Springer-Verlag, Berlin, 2003.

[10] S.Y.M. Shi and P.N. Suganthan. Feature analysis and classification of protein secondary structure data. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Lecture Notes in Computer Science (Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Istanbul, Turkey)*, volume 2714, pages 1151–1158. Springer-Verlag, Berlin, 2003.

[11] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, Berlin, 1995.

[12] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 985–992. MIT Press, Cambridge, MA, 2002.