Testing Adaptive Local Hyperplane for Multi-class Classification by Double Cross-Validation

Tao Yang, Vojislav Kecman, *Member IEEE*, Longbing Cao, *Senior Member IEEE*
and Chengqi Zhang, *Senior Member IEEE*

***Abstract*— Adaptive Local Hyperplane (ALH) is a recently proposed classifier for the multi-class classification problems and it has shown encouraging performance in many pattern recognition problems. However, ALH's performance over many general classification datasets has only been tested by using a single loop of cross-validation procedure, where the whole datasets are used for both hyper-parameter determination and accuracy estimation. This procedure is appropriate for classifier performance comparison, but the produced results are likely to be optimistic for classifier accuracy estimation on new datasets.**

In this paper, we test the performance of ALH as well as several other benchmark classifiers by using two loops of cross-validation (a.k.a. double resampling) procedure, where the inner loop is used for hyper-parameter determination and the outer loop is used for accuracy estimation. With such a testing scheme, the classification accuracy of a tested classifier can be evaluated in a more strict way. The experimental results indicate the superior performance of the ALH classifier with respect to the traditional classifiers including Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Linear Discriminant Analysis (LDA), Classification Tree (Tree) and K-local Hyperplane distance Nearest Neighbor (HKNN). These results imply that the ALH classifier might become a useful tool for the pattern recognition tasks.

I. Introduction

Recently, the Adaptive Local Hyperplane (ALH) method [6], [5] has been proposed for the general multi-class classification problem, which is a fundamental task in pattern recognition, machine learning and data mining. The ALH classifier is a local margin optimization algorithm in the original, weighted, input space blending a Nearest Neighbors (NN) based approaches and Support Vector Machines (SVMs) ideas about the maximal margin. It has been shown in [6] that the ALH classifier outperforms several traditional classifiers which include the SVM, KNN (K-Nearest Neighbors), LDA (Linear Discriminant Analysis) and HKNN (K-local Hyperplane distance Nearest Neighbor) [4] on eleven benchmark datasets. Encouraged by such a performance, the ALH classifier has been applied in the tasks of face recognition [7], protein fold recognition [8] and small dataset learning [9]. Again, the ALH classifier has obtained superior performance than all the existing classifiers used in the above-mentioned domains. Furthermore, the natural capacity of ALH to the multi-class problems allows ALH to be easily extended to the regression problems, and it has obtained the best performance, on average, among the Linear Regression, SVM, KNN and Regression Tree models over several real datasets [10].

Although the ALH classifier has been tested over a variety of real world problems, we use a more strict classifier performance evaluation procedure to validate ALH's performance. The aim in [6] is primarily to compare ALH with other models on the same datasets and under same conditions, i.e. model comparison, not to assess the true accuracy of ALH, i.e. model assessment. Thus, the classification accuracies for the classifiers considered in [6] are obtained by using a single loop of cross-validation procedure, where the whole datasets are used for both hyper-parameter determination and accuracy estimation. Therefore, the reported results in [6] may be the 'optimistic' estimates of the classification accuracy on new dataset.

In this paper, we test the performance of the ALH classifier as well as the competing classifiers by using two loops of cross-validation procedure. In the outer loop, the dataset is separated into J_1 roughly equal-sized parts. Each part is held out in turn as the test set, and the remaining parts are used as the training set. Then in the inner loop, another J_2-fold cross-validation is performed over the training set only to determine the values of hyper-parameters for the given classifier. Thus, the model with the specified hyper-parameters is applied to the test set and the classification accuracy can be calculated. The whole process is repeated N_r times after the whole dataset has been randomly shuffled. The double cross-validation procedure ensures that the correct labels of the test dataset won't be seen when tuning the hyper-parameters of a given model, which is consistent with the real-world scenario.

By taking the statistical randomness into consideration, the pairwise comparisons between two classifiers can be measured by using both the accuracy values and the associated standard errors. The standard error of the classification accuracies over the $J_1 \cdot N_r$ runs can also be calculated by

$$E = S/\sqrt{J_1 \cdot N_r}, \tag{1}$$

where S represents the standard deviation of the accuracies over the $J_1 \cdot N_r$ runs. Suppose the classification accuracies for classifier 1 and classifier 2 are A_1 and A_2 and the corresponding standard errors are E_1 and E_2, respectively. Thus, we can conclude that classifier 1 has significantly higher accuracy than classifier 2 if we have $A_1 - E_1 > A_2 + E_2$; the difference

Tao Yang (tyan028@gmail.com), Longbing Cao (lbcao@it.uts.edu.au) and Chengqi Zhang (chengqi@it.uts.edu.au) are with the Faculty of Engineering and Information Technology, The University of Technology, Sydney, Australia. Vojislav Kecman (vkecman@vcu.edu) is with the Virginia Commonwealth University (VCU), School of Engineering, Department of Computer Science, Richmond, VA, USA.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

between the accuracies of classifier 1 and classifier 2 is not significant if we have $A_1 - E_1 \leq A_2 + E_2$ and $A_1 \geq A_2$.

II. ALH Method

In multi-class classification, we are given a training dataset of N samples, on which we have D feature measurements: $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})'^1 \in \Re^D$. Suppose there are M classes in both the training and test datasets, then each training samples has a known class label $y_i \in \{1, \ldots, M\}$. The goal of a classification method is to predict the class membership of the query $\mathbf{q} = (q_1, \ldots, q_D)'$.

We give a brief introduction to the ALH classifier for the above problem. The ALH classifier tries to improve both the KNN and HKNN methods by optimizing the local margin for class separation in the weighted feature space. At the training stage, the ALH method computes the feature weights, which refers to the discrimination ability of a feature in a classification task, by using the exponential transformation on the ratio of the between-group to within-group sums of squares for each feature:

$$w_j = \frac{\exp(TR_j)}{\sum_{j=1}^{D} \exp(TR_j)}, \quad (2)$$

$$R_j = r_j / \max(r_j), \quad (3)$$

$$r_j = \frac{\sum_i \sum_c I(y_i = c)(\bar{x}_{cj} - \bar{x}_j)^2}{\sum_i \sum_c I(y_i = c)(x_{ij} - \bar{x}_{cj}^2}, \quad \forall j, \quad (4)$$

where $I(.)$ denotes the indicator function (whose value is 1 if its argument is true and 0 otherwise), $\bar{x}_{cj}$ denotes the jth component of the class centroid for class c and $\bar{x}_j$ denotes the jth component of the grand class centroid. Here, T is a non-negative temperature parameter that controls the influence of R_j on w_j. If $T = 0$, then $w_j = 1/D$, $\forall j$, implying all features have equal weights. On the other hand, when T is large, a change in R_j will be exponentially reflected in w_j. Moreover, the exponential weighting scheme forces $w_j \neq 0$, which prevents neighborhoods from extending infinitely in one direction. The exponential weighting procedure forces $w_j \neq 0$, which prevents neighborhoods from extending infinitely in one direction. We illustrate the dependence of w_j on T in Figure 1. After specifying the feature weights given in Equations (2), (3) and (4), the weighted distance metric for finding the K nearest neighbors of the query $\mathbf{q}$ can then be determined by

$$\text{Dis}(\mathbf{x}_i, \mathbf{q}) = \sqrt{\sum_{j=1}^{D} w_j (x_{ij} - q_j)^2}. \quad (5)$$

If we use the majority voting rule to classify the query, then the algorithm is a weighted KNN (WKNN) rule. A two-class classification problem is shown in Figure 1, where the WKNN algorithms with various values of T are applied on the same data. The data points are uniformly distributed in a square. The dataset consist of two features denoted by x_1 and x_2 and two classes denoted by squares and circles. Here,

[1] $'$ represents the transpose of a vector or matrix throughout this paper.

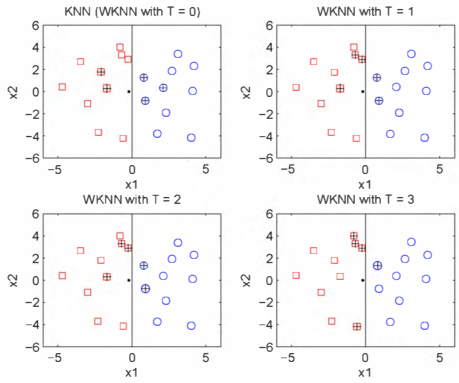

Fig. 1. Two-class example by using weighted KNN (WKNN) classification rule with various T.

the query is represented by a solid dot located in the center of the figures. The vertical line is drawn to separate the two classes. Thus, the correct classification for the query would be the square class. We set $K = 5$ in all four cases. The selected nearest neighbors are labeled by the '+' sign. In this problem, KNN finds three circle nearest neighbors and two square nearest neighbors around the query, and thus it will assign the query to the wrong (circle) class. However, the KNN classifier can be improved here by considering the feature weights. Obviously, the two classes can be perfectly separated by considering x_1 only, whereas x_2 can provide no help for classification. All the WKNN algorithms with various T would make the correct decisions for the query. Moreover, the K neighbors show greater variation in x_2 for a larger T. This example shows that the WKNN algorithm implicitly shrinks the neighborhood in directions in which the class centroids differ. In contrast, the KNN algorithm draws a spherical neighborhood around the query.

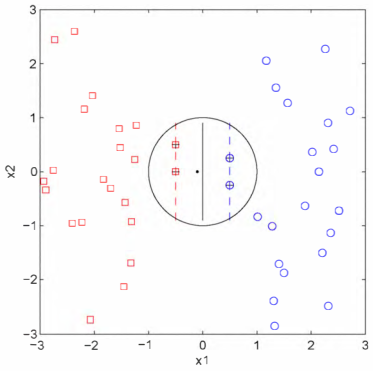

Fig. 2. Local maximum margin found by the ALH classifier.

At the test phase of ALH, the K nearest neighbors of $\mathbf{q}$ are firstly selected as the class prototypes by using the weighted Euclidean distance metric defined in Equation (5), ALH then constructs a local hyperplane for each class to find a virtually enriched training set. This scheme implicitly maximizes the local margin surrounding the query. Figure 2 shows a two-

dimensional example for a two-class classification problem. The feature weights are assumed to be equal for this problem. Consider the spherical neighborhood around the query (labeled by a black dot), two nearest neighbors (labeled by '+') are selected for each class. A local hyperplane is drawn for each class. The local decision boundary produced by ALH is drawn by the black line. The ALH model will classify the query as belonging to the red class, as it will for all points left of the decision boundary. Formally, the local hyperplane of class c is defined as:

$$LH_c(\mathbf{q}) = \{\mathbf{s} \mid \mathbf{s} = \sum_{i=1}^{K_c} \alpha_i \mathbf{V}_{.i} + \mathbf{m}\}, \quad c = 1, \ldots, M, \quad (6)$$

where K_c is the number of prototypes (nearest neighbors) in class c, $\mathbf{m} = \frac{1}{K_c}\sum_{i=1}^{K_c} \mathbf{p}_i$, $\mathbf{p}_i$ is the ith prototype of class c, $\mathbf{V}$ is the $D \times K_c$ matrix whose ith column is defined as: $\mathbf{V}_{.i} = \mathbf{p}_i - \mathbf{m}$, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{K_c})'$ are solved by minimizing the regularized squared distance between $\mathbf{q}$ and $LH_c(\mathbf{q})$ in the *weighted* feature space, which leads to the minimization of

$$J_c(\mathbf{q}) = \min_{\alpha} \sum_{j=1}^{D} w_j \left(\mathbf{V}_{j.}\boldsymbol{\alpha} + m_j - q_j\right)^2 + \lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha}, \quad (7)$$

$$= \min_{\alpha} (\mathbf{s} - \mathbf{q})' \mathbf{W} (\mathbf{s} - \mathbf{q}) + \lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha}, \quad (8)$$

where $\mathbf{V}_{j.}$ is the jth row of $\mathbf{V}$, $\mathbf{s} \in LH_c(\mathbf{q})$, $\mathbf{W}$ is the diagonal matrix with $W(j,j) = w_j$ and λ is the regularization parameter. It has been proved in [6] that the minimization of (8) can be achieved by solving the equation given below for $\boldsymbol{\alpha}$:

$$(\mathbf{U}'\mathbf{V} + \lambda\, \mathbf{I}_{K_c})\boldsymbol{\alpha} = \mathbf{U}'(\mathbf{q} - \mathbf{m}), \quad (9)$$

where $\mathbf{U}' = \mathbf{V}'\mathbf{W}$.

Once the minimal $\boldsymbol{\alpha}^*$ is obtained in (9), the corresponding (squared) weighted distance between $\mathbf{q}$ and $LH_c(\mathbf{q})$ can be found as:

$$J_c^*(\mathbf{q}) = \sum_{j=1}^{D} w_j \left(\mathbf{V}_{j.}\boldsymbol{\alpha}^* + \bar{p}_j - q_j\right)^2 + \lambda (\boldsymbol{\alpha}^*)^T (\boldsymbol{\alpha}^*). \quad (10)$$

Finally, the class label of the query $\mathbf{q}$ is assigned as:

$$f(\mathbf{q}) = \arg\min_{c} \; J_c^*(\mathbf{q}). \quad (11)$$

There are three hyper-parameters in the ALH classifier: the number K of nearest neighbors, the temperature parameter T and the regularization parameter λ. It has been shown that the performance of ALH is not sensitive to the selection of its hyper-parameters [6]. The basic procedure for implementing the ALH model is summarized in Table I.

A graphical illustration of the ALH algorithm is shown in Figure 3, where a classification problem with 4 classes and 2 features is depicted. The training data points are represented by circles, squares, diamonds and triangles with different colors (each color or shape corresponds to a class category), and the query $\mathbf{q}$ is represented by the black dot. In this case, $K = 5, T = 1, \lambda = 0$ are used in ALH. The 5 nearest neighbors of $\mathbf{q}$ are selected by the weighted

TABLE I

THE ALH CLASSIFICATION ALGORITHM

Input: training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, query $\mathbf{q}$, hyper-parameters K, T and λ

1. Weigh the features by using the exponential transformation on the ratio of between-group to within-group sum of squares using Equations (2) - (4).
2. Compute the query - training sample distances in the weighted feature space using Equation (5).
3. Find K nearest neighbors by ordering the distances computed in Step 2.
4. Compute the vector $\boldsymbol{\alpha}$ for each class separately using Equation (9).
5. Compute the query - local hyperplane distances in the weighted feature space using Equation (10).
6. Classify the query using Equation (11).

Output: the class label of the query, $f(\mathbf{q})$

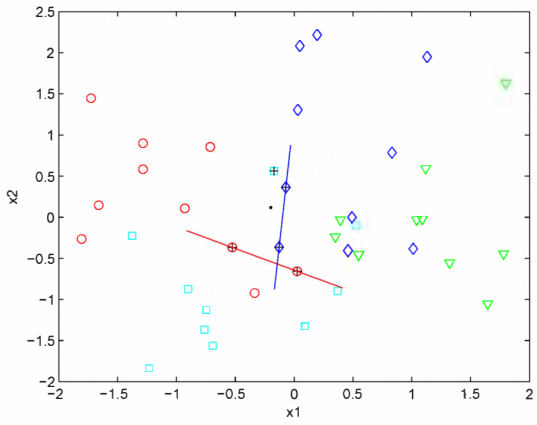

Fig. 3. Graphical illustration of the ALH classifier

Euclidean distances between the query and training data points, and they are labeled by the '+' sign. It can be seen that there are 2 nearest neighbors in the red (circle) class, 2 nearest neighbors in the blue (diamond) class, only 1 nearest neighbor in the cyan (square) class and no nearest neighbor in the green (triangular) class. Hence, the green class will be ignored in later computations. Then, a local manifold is built for each of the red, blue and cyan classes. In our case, straight lines are drawn as the local hyperplane for both the red and blue classes, because there are exactly 2 nearest neighbors in these classes. The last local hyperplane is a point coinciding with a single data point in the cyan class.

Next, the parameter vectors $\boldsymbol{\alpha}^*$ are computed for each class to find the closest projection points $\mathbf{s}$ to $\mathbf{q}$. Then, the weighted Euclidean distances $J_c^*(\mathbf{q})$ between $\mathbf{s}$ and $\mathbf{q}$ can be determined for each class separately. Finally, the query $\mathbf{q}$ is assigned to the class for which $J_c^*(\mathbf{q})$ is the smallest. In this example, we will assign $\mathbf{q}$ to the blue class because $J_r(\mathbf{q}) = 0.1208, J_b(\mathbf{q}) = 0.0079, J_c(\mathbf{q}) = 0.0624$.

The user-friendly, parsed MATLAB code implementing the ALH algorithm can be downloaded from (http://www.people.vcu.edu/∼vkecman/Downloads). This software can be freely used for all non-commercial purposes.

III. Experimental Results

In this section, we compare the ALH classifier to several traditional classifiers including KNN, LDA, Tree (classification tree), SVM and HKNN on the eleven real world datasets used in [6]. The first nine datasets used here are taken from the UCI Machine Learning Repository, and the last two datasets are the benchmarking datasets for protein subcellular localization constructed by Reinhardt and Hubbard [2]. The characteristics of the datasets used in this study are summarized in Table II.

TABLE II
ELEVEN CLASSIFICATION DATASETS USED IN THIS STUDY

Data	# samples	# features	# classes
Iris	150	4	3
Teach	151	5	3
Wine	178	13	3
Cancer	198	32	2
Sonar	208	60	2
Glass	214	9	6
Vote	232	16	2
Heart	297	13	5
Dermatology	366	33	6
Prokaryotic	997	20	3
Eukaryotic	2427	20	4

The double cross-validation scheme is used to test the classification accuracies of the above-mentioned classifiers. We used $J_1 = 5$ and $J_2 = 10$ in all experiments and $N_r = 10$ was used for the first ten datasets and $N_r = 1$ was used for the last dataset (in order to be computationally feasible). By using this cross-validation procedure, the size of the test data for the first ten datasets is actually $J_1 \cdot N_r \cdot N = 50N$ and the size of the test data for the last dataset is N.

Like ALH, several competing classifiers have hyper-parameters in their models. KNN has a single hyper-parameter K (the number of nearest neighbors), Tree has a single hyper-parameter N_v (the minimal number of instances in the impure nodes for further partitioning), HKNN has K and λ and SVM has σ (the Gaussian kernel shape) and C (the penalty parameter). The values of these hyper-parameters were determined empirically by using the inner cross-validation techniques used on the training dataset only.

The suggested candidate values for these hyper-parameters used in our experiments are given as follows:

- KNN: $K \in \{1, 3, \ldots, 31\}$
- Tree: $N_v \in \{5, 10, \ldots, 50\}$
- SVM (with Gaussian kernel): $\sigma \in \{2^{-10}, 2^{-9}, \ldots, 2^5\}$, $C \in \{2^{-5}, 2^{-4}, \ldots, 2^{12}\}$
- HKNN: $K \in \{5, 7, \ldots, 31\}$, $\lambda \in \{0, 0.5\}$
- ALH: $K \in \{5, 7, \ldots, 51\}$, $T \in \{0, 1, 2, 3, 4, 7\}$, $\lambda \in \{0, 0.5\}$

Please note that there are 288 combinations of the candidate hyper-parameter values in both SVM and ALH. Thus, the practical comparisons between SVM and ALH are fair in the hyper-parameters tuning aspect. In all the experiments, the training inputs are first standardized to zero mean and unit variance, and then the test inputs are standardized using the corresponding training mean and variance.

TABLE III
MEAN ACCURACIES (%) AND STANDARD ERRORS FOR SIX CLASSIFIERS OVER ELEVEN DATASETS

Data	KNN	LDA	Tree	SVM	HKNN	ALH
Iris	94.9	**97.9**	94.7	95.7	95.9	95.5
	(0.51)	(0.33)	(0.51)	(0.53)	(0.49)	(0.41)
Teach	58.7	53.4	49.6	54.7	52.5	**63.7**
	(1.37)	(1.14)	(1.34)	(1.21)	(1.46)	(1.19)
Wine	96.1	**98.5**	88.6	97.9	96.9	97.7
	(0.51)	(0.30)	(0.73)	(0.30)	(0.40)	(0.37)
Cancer	76.8	71.7	69.5	**77.8**	67.6	**77.8**
	(0.75)	(1.00)	(1.00)	(0.88)	(1.03)	(0.77)
Sonar	84.7	73.7	67.4	85.9	87.4	**88.3**
	(0.66)	(0.80)	(0.97)	(0.75)	(0.61)	(0.63)
Glass	68.0	58.8	68.8	67.6	**70.1**	70.0
	(1.05)	(1.05)	(0.91)	(1.03)	(0.99)	(0.94)
Vote	91.4	**97.0**	95.8	96.3	93.5	96.2
	(0.59)	(0.36)	(0.42)	(0.42)	(0.49)	(0.41)
Heart	**58.0**	53.7	49.8	56.8	50.6	57.1
	(0.86)	(0.97)	(0.89)	(0.80)	(0.83)	(0.83)
Dermatology	95.6	96.8	94.7	**97.6**	95.9	97.3
	(0.31)	(0.28)	(0.32)	(0.21)	(0.32)	(0.27)
Prokaryotic	87.3	84.5	75.4	**90.0**	89.1	89.5
	(0.28)	(0.34)	(0.45)	(0.27)	(0.30)	(0.29)
Eukaryotic	81.3	62.7	62.3	80.9	81.7	**82.3**
	(0.58)	(0.25)	(0.69)	(1.01)	(0.92)	(1.08)
Average	81.2	77.2	74.2	81.9	80.1	**83.2**

The bold numbers represent the best performance for each dataset.

The cross-validation accuracies and the associated standard errors (shown in brackets) are reported in Table III. The bold numbers represent the highest accuracy values for each dataset. As it can be seen, the overall performance ranking for all the classifiers is ALH > SVM > KNN > HKNN > LDA > Tree. Furthermore, ALH has obtained the highest accuracies in 4 datasets while SVM has obtained the highest accuracies in 3 datasets. While performance improvement of ALH over the competing methods may not be significant on some datasets, the results nonetheless indicate ALHs practical potential over other competing classifiers. SVM is the overall second best classifier, which is consistent with its popularity in literature. It is interesting to note that the performances of LDA and CART are much worse than the other classifiers. LDA is based on the assumption that the underlying distribution for each class is a Gaussian distribution. Since this perfect data assumption is often violated in practice, LDA will not work well for many imperfect but real datasets. The performance of CART is the worst among the six classifiers. It is understandable since CARTs popularity in data mining is due to its interpretability rather than the predictive power [1].

By taking the standard errors into consideration, the number of wins, draws and loses for ALH with respect to the other classifiers are reported in Table IV. For example, it can be seen that ALH shows superior performance with respect to HKNN from the fact that it shows equal performance to, or better one than, HKNN over all datasets used. ALH has also shown equal or better performances with respect to KNN, Tree, and SVM methods over all the datasets too. In

other words, none of these methods has a single win against ALH comparing the performances of the classifiers by one standard error intervals. Only LDA method (in addition to having 7 losses and 1 draw with respect to ALH) was able to outperform ALH over three (Iris, Wine and Vote) datasets in the strict double cross-validation comparisons. This is an interesting result taking into account that LDA, together with Tree method, performed the worst in term of average mean accuracies over 11 datasets. With respect to SVM (which is the second best method in terms of average mean accuracies), ALH has 8 draws and 3 wins. Teach, Sonar and Glass datasets are the ones where ALH outperforms SVM.

TABLE IV
COMPARING ALH WITH FIVE OTHER CLASSIFIERS BY ONE STANDARD ERROR INTERVALS.

	KNN	LDA	Tree	SVM	HKNN
# wins	7	7	8	3	6
# draws	4	1	3	8	5
# loses	0	3	0	0	0

IV. CONCLUSIONS

In this paper, a recently proposed Adaptive Local Hyperplane (ALH) classifier has been tested in a more rigorous evaluation approach compared to the conventional approach. Previously, the ALH method has been tested by using a single loop of cross-validation, where the whole dataset is used for both hyper-parameter determination and accuracy estimation. This study employs a scheme with two loops of cross-validation to assess the classifier performance such that the hyper-parameters of a classifier can be determined by using the training dataset only. The ALH classifier has been compared with several benchmark classifiers, which include the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Linear Discriminant Analysis (LDA), Classification Tree (Tree) and K-local Hyperplane distance Nearest Neighbor (HKNN) methods. The experimental results show that the ALH classifier is the overall best method over the eleven datasets used. In particular, the ALH classifier performs equal or better than the SVM classifier on all eleven datasets, and its accuracy is significantly better than that of SVM in three out of eleven cases.

REFERENCES

[1] T. Hastie, R. Tibshirani, J. Fridman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, New York, 2001.

[2] A. Reinhardt and T. Hubbard, "Using neural networks for prediction of the subcellular location of proteins," *Nucleic Acids Research*, vol. 26, pp.2230–2236, 1998.

[3] V.N. Vapnik, *Statistical learning theory*, The MIT press, New York, 1998.

[4] P. Vincent and Y. Bengio, "K-local hyperplane and convex distance nearest neighbor algorithms," *Advances in Neural Information Processing Systems (NIPS)*, MIT Press, Cambridge, MA, vol.14, pp.985–992, 2002.

[5] T. Yang, *Machine Learning by Adaptive Local Hyperplane Algorithm*, The University of Auckland, PhD Thesis, submitted, 2009.

[6] T. Yang and V. Kecman, "Adaptive local hyperplane classification," *Neurocomputing*, vol. 71, pp.3001–3004, 2008.

[7] T. Yang and V. Kecman, "Face recognition with adaptive local hyperplane algorithm," *Pattern Analysis & Applications*, Theoretical Advances, Springer-Verlag, in press, 2009.

[8] V. Kecman and T. Yang, "Protein fold recognition with adaptive local hyperplane algorithm," In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (IEEE CIBCB 2009)*, pp.75–78, Nashville, TN, USA, 2009.

[9] T. Yang and V. Kecman, "A novel algorithm for learning small medical dataset," *Expert Systems*, vol.26, 355–359, 2009.

[10] V. Kecman and T. Yang, "Adaptive local hyperplane for regression tasks," In *IEEE International Joint Conference on Neural Networks (IJCNN 2009)*, pp.1566–1570, Atlanta, GA, USA, 2009.