Improving accuracy of LVQ algorithm by instance weighting.

Marcin Blachnik[1], Włodzisław Duch[2]

[1] Department of Management and Informatics, Silesian University of Technology
Katowice, Krasinskiego 8, Poland; marcin.blachnik@polsl.pl
[2] Department of Informatics, Nicolaus Copernicus University,
Grudziądzka 5, Toru, Poland; Google: W Duch

Abstract. Similarity-based methods belong to the most accurate data mining approaches. A large group of such methods is based on instance selection and optimization, with Learning Vector Quantization (LVQ) algorithm being a prominent example. Accuracy of LVQ highly depends on proper initialization of prototypes and the optimization mechanism. Prototype initialization based on context dependent clustering is introduced, and modification of the LVQ cost function that utilizes additional information about class-dependent distribution of training vectors. The new method is illustrated on 6 benchmark datasets, finding simple and accurate models of data in form of prototype-based rules.

1 Introduction

Lazy learning has many advantages [1] and can be extended in many ways [2, 3]. Nearest neighbor algorithms belong to the simplest models of the similarity-based learning (SBL) framework. They need very little training (parameter selection), are frequently quite accurate, may use any type of attributes, and may be applied directly to similarity matrices in cases when vector-based representation of information is not possible. They may use specialized kernels [4] for similarity evaluation, including kernels for time series, biological sequences, spectral kernels, graphs and other complex structures, mutual information and other measures of dependency etc. [5, 6]. Although these kernels are commonly used in Support Vector Machines (SVMs) to create extended feature space in which linear discrimination is performed they may also be used in any SBL algorithm. This is still a relatively unexplored area in machine learning.

Unfortunately similarity-based algorithms, such as the k-nearest neighbor (kNN) method, also have some drawbacks. First, although there is little learning prediction may be time-consuming, because it requires distance calculations between the test $\mathbf{t}_i$ and all training set instances $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n]$. Several algorithms for exact and approximate fast computation of distance exist. For example, BallTree or KDTree [7] algorithms reduce complexity of distance calculations by using tree-structured search methods (see recent paper [8] for review of approximate ϵ-NN algorithms). Another approach that can reduce prediction time, improve prediction accuracy and comprehensibility of the kNN algorithm is based on selection and optimization of the training set reference vectors $\mathbf{D}$.

Lecture Notes in Computer Science Vol. 6353, pp. 256–265, 2010.

Selection removes those instances from the training data that do not bring significant gains in the prediction process. Some examples fo algorithms that belong to this group include top-down algorithms: **Edited Nearest Neighbor (ENN)** [9] removes from the training set instances that are incorrectly classified by kNN for $k \geq 3$; **DROP3** [10] removes instance $\mathbf{x}$ from the training set if it does not change classification of instances for which $\mathbf{x}$ is one of the k nearest neighbors; **Iterative Case Filtering (ICF)** [11] starts from DROP3 and creates hyperspheres with instances from a single-class. Bottom-up algorithms: **CNN (Condensed Nearest Neighbor rule)** starts with one vector per class and adds training instances that are wrongly classified by the current reference set. **Graph-based methods**: **Gabriel Editing (GE)** [12] uses Gabriel graph to remove from the training dataset all instances from the same class as all their neighbors, leaving only those instances that define decision borders between different classes. Probability-density estimating algorithms: **Edited NRBF** [13] uses normalized RBF to remove instances $\mathbf{x}$ inconsistent with probability density estimation around point $\mathbf{x}$.

Optimization methods may start from the selected vectors but shift reference instances to optimal positions. Typical examples that belong to optimization group are numerous clustering algorithms, including the LVQ algorithm [14, 15], which is used in this paper.

Empirical comparison of these methods [13, 16] shows that the best results are usually achieved by combination of selection and optimization algorithms. Such conclusion follows from the sensitivity of LVQ to initialization of codebook vector positions ("codebook" is used in vector quantization as a synonym of reference vector [14]). Selection algorithms usually provide a very good initialization for LVQ optimization algorithm. Although the accuracy is high selection algorithms frequently overestimate the number of codebook required to achieve maximum accuracy. Minimization of the number of codebook vectors increases comprehensibility of the model, allowing to define prototype-based rules (P-rules) based on evaluation of similarity to a small number of prototypes [17]. In [18] context-dependent clustering algorithm has been applied to find appropriate initial position of codebooks vectors. The context of the clustering algorithm was defined to ensure that clusters centroids are placed close to the decision borders. Focus on regions close to the decision border allows for improvement of classification accuracy and reduction of the number of codebook vectors at the same time.

In this paper further improvements of the LVQ optimization procedure along these lines are considered. This is achieved by using training vector weights determined by the clustering context. In the next section a short update on the context-dependent clustering is given, followed by a definition of context and reformulation of LVQ algorithm. Empirical tests on 6 benchmark datasets are presented in section 3, and results are compared to the original LVQ algorithm.

2 Context dependent clustering

Context dependent clustering uses context information to guide clustering process. In the simplest case context is described by an external variable that estimates importance of each vector. Conditional Fuzzy C-Means (CFCM) [19] is an extension of the FCM clustering algorithm, with additional variable $\mathbf{y}$, defined for every vector $\mathbf{x}$, that helps

to cluster related data. For every vector $\mathbf{x}_i$ the strength of its relation with the external variable y_i is defined by a function $f_i = \mu_A(\mathbf{y}) \in [0,1]$, where $\mu_A(\mathbf{y})$ is interpreted as a membership function for some fuzzy set A, or simply as a weight that creates clustering condition in context A.

FCM and CFCM are based on minimization of a cost function defined as:

$$J_m(\mathbf{U},\mathbf{V}) = \sum_{k=1}^{N_c}\sum_{i=1}^{n}(u_{ki})^m \left\|\mathbf{x}_i - \mathbf{v}_k\right\|_A^2 \tag{1}$$

where N_c is the number of clusters centered at $\mathbf{v}_k$ (such centers will be used as reference vectors or the LVQ classifier), n is the number of vectors, $m > 1$ determines clustering fuzziness, and $\mathbf{U} = (u_{ki})$ is a $N_c \times n$ dimensional membership matrix with elements $u_{ki} \in [0,1]$ defining the membership degree of vector $\mathbf{x}_i$ in cluster $\mathbf{v}_k$. The matrix $\mathbf{U}$ has to fulfill three conditions:

1^o each vector x_i belongs to the k-th cluster to a degree between 0 to 1:

$$u_{ki} \in [0,1]; k = 1..N_c; i = 1..n \tag{2}$$

2^o sum of the membership values of i-th vector x_i in all clusters is equal to f_i

$$\sum_{k=1}^{N_c} u_{ki} = f_i; i = 1..n \tag{3}$$

3^o clusters are not empty, and a single cluster does not cover whole space:

$$\sum_{i=1}^{n} u_{ki} \in (0,n); k = 1..N_c \tag{4}$$

Centers of the clusters are calculated as:

$$\mathbf{v}_k = \sum_{i=1}^{n}(u_{ki})^m x_i \Bigg/ \sum_{i=1}^{n}(u_{ki})^m; k = 1..N_c \tag{5}$$

The partition matrix is then calculated using cluster centers, and the cost function (1) is minimized iteratively:

$$u_{ki} = f_i \Bigg/ \sum_{k=1}^{C}\left(\frac{\|x_i - \mathbf{v}_k\|}{\|x_i - \mathbf{v}_k\|}\right)^{2/(m-1)}; i = 1..n; k = 1..N_c \tag{6}$$

3 Determining the context

Defining appropriate context should help to find minimum number of LVQ prototypes that provide accurate model of the data. This is an alternative to various editing methods. Support vector machines work quite well because they focus only at the border area,

and thus may provide also useful prototypes for LVQ [20]. To encourage selection of prototypes that lie close to decision border context for CFCM clustering is defined with the help of a coefficient w_k describing distribution of vector positions around $\mathbf{x}_k$ within the same and relatively to other classes. This is defined by the ratio of a within-class scatter to out-of-class scatter:

$$w_i = w(\mathbf{x}_i) = \frac{(n - n_c)}{d \cdot n_c} \frac{\sum_{j, c(\mathbf{x}_i)=c(\mathbf{x}_j)} \|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sum_{j, c(\mathbf{x}_i)\neq c(\mathbf{x}_j)} \|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{7}$$

Here $c(\mathbf{x}_i)$ denotes class label c_i of the vector $\mathbf{x}_i$, with n_c vectors in this class, and d is the number of features. For normalized attributes w_i coefficients have values that are close to 0 for vectors inside large homogenous clusters, and close to 1 if the vector $\mathbf{x}_i$ is near the vectors of the opposite classes and far from other vectors from the same class (for example, if it is an outlier). To avoid such situations these weights should be centered around $\mu \in [0.6, 1]$ with a Gaussian membership function, defining the context factor:

$$f_i = f(\mathbf{x}_i) = \exp\left(-\gamma \left(w(\mathbf{x}_i) - \mu\right)^2\right) \tag{8}$$

with $\gamma \in [0.4, 0.8]$, determined empirically for a wide range of datasets. The μ parameter controls where the prototypes will be placed; for small μ they are closer to the center of the cluster and for larger μ closer to the decision borders. The range in which they are sought is determined by the γ parameter. This function is used to determine the clustering context for CFCM clustering in the weighted LVQ algorithm (WLVQ).

4 Introducing context in LVQ algorithm

Classical LVQ1 algorithm is based on the cost function defined as:

$$\begin{aligned} E(\mathbf{P}) = &\frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^{n} \mathbf{1}\left(\mathbf{x}_i \in R_k\right) \mathbf{1}\left(c(\mathbf{x}_i) = c(\mathbf{p}_k)\right) ||\mathbf{x}_i - \mathbf{p}_k||^2 \\ &-\frac{1}{2} \sum_{k=1}^{N_c} \sum_{i=1}^{n} \mathbf{1}\left(\mathbf{x}_i \in R_k\right) \mathbf{1}\left(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)\right) ||\mathbf{x}_i - \mathbf{p}_k||^2 \end{aligned} \tag{9}$$

where $\mathbf{1}(L)$ identity function returns 1 when condition L is *true*, and 0 otherwise, $c(\mathbf{x})$ returns class label of vector $\mathbf{x}$, and R_k is the Voronoi area defined for prototype $\mathbf{p}_k$. This relation can also be interpreted as $\mathbf{x}_i \in R_k$ if $\mathbf{p}_k = \arg \min_{l=1\ldots N_c} \left(D\left(\mathbf{x}_i, \mathbf{p}_l\right)\right)$. This cost function is minimized in respect to position of all prototypes $\mathbf{P}$ using the two update formulas that iteratively change the position of k-th prototype $\mathbf{p}_k$ according to:

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_k + \alpha(j)\mathbf{1}\left(c(\mathbf{x}_i) = c(\mathbf{p}_k)\right)\left(\mathbf{x}_i - \mathbf{p}_k\right) \\ \mathbf{p}_k &= \mathbf{p}_k - \alpha(j)\mathbf{1}\left(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)\right)\left(\mathbf{x}_i - \mathbf{p}_k\right) \end{aligned} \tag{10}$$

where $\alpha(j) \in [0, 1]$ is monotonically decreasing step size function and j is the iteration number.

In the original LVQ algorithm only local information is taken into account. Applying the context dependent clustering to LVQ training the $f(\mathbf{x}_i)$ factor is introduced to incorporate some information about overall distribution of vectors from different classes, such that outliers or vectors that appear far from the decision border should have less influence on the value of the cost function. It requires reformulating the cost function, which takes the form:

$$\begin{aligned} E(\mathbf{P}) = &\frac{1}{2}\sum_{k=1}^{N_c}\sum_{i=1}^{n}\mathbf{1}\left(\mathbf{x}_i \in R_k\right)\mathbf{1}\left(c(\mathbf{x}_i) = c(\mathbf{p}_k)\right) f(\mathbf{x}_i)\left|\left|\mathbf{x}_i - \mathbf{p}_k\right|\right|^2 \\ &-\frac{1}{2}\sum_{k=1}^{N_c}\sum_{i=1}^{n}\mathbf{1}\left(\mathbf{x}_i \in R_k\right)\mathbf{1}\left(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)\right) f(\mathbf{x}_i)\left|\left|\mathbf{x}_i - \mathbf{p}_k\right|\right|^2 \end{aligned} \tag{11}$$

This cost function can be minimized modifying formula (10) such that it takes form:

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_k + \alpha(j) f(\mathbf{x}_i)\mathbf{1}\left(c(\mathbf{x}_i) = c(\mathbf{p}_k)\right)\left(\mathbf{x}_i - \mathbf{p}_k\right) \\ \mathbf{p}_k &= \mathbf{p}_k - \alpha(j) f(\mathbf{x}_i)\mathbf{1}\left(c(\mathbf{x}_i) \neq c(\mathbf{p}_k)\right)\left(\mathbf{x}_i - \mathbf{p}_k\right) \end{aligned} \tag{12}$$

where the context factor $f(\mathbf{x}_i)$ describes class-dependent distribution of vectors around $\mathbf{x}_i$. The $f(\mathbf{x}_i)$ value can also be understood as an instance weight describing the importance of that instance during the training process.

5 Experimental results

Experiments have been performed on 6 benchmark datasets with real valued attributes obtained from the UCI repository [21]. They include Cleveland Heart Disease, Ionosphere, Pima Indian Diabetes, Sonar, Spambase, and Wisconsin Breast Cancer (WBC).

Title	#Features	#Samples	#Samples per class	Source
Heart	13	297	160 absence / 137 presence	[21]
Ionosphere	34	351	224 / 126	[21]
Diabetes	8	768	500 / 268	[21]
Sonar	60	208	111 metal / 97 rock	[21]
Spambase	57	4601	1813 / 2788	[21]
WBC	9	699	458 / 241	[21]

Table 1. Summary of datasets used in experiments

5.1 Experiment description

Diagram presenting data flow in computational experiments is presented in Fig (1). In these experiments accuracy of the nearest-prototype classifier with prototypes obtained from classical LVQ algorithm have been compared with those based on prototypes optimized using the WLVQ algorithm. In both cases prototypes/codebooks were initialized

using CFCM clustering algorithm. All processing steps have been embedded in the 10 fold cross-validation procedure. In case of WLVQ algorithm optimal values γ and μ were selected using greedy search algorithm from the 4 values of $\gamma = [0.2, 0.4, 0.6, 0.8]$ and 4 values of $\mu = [0.4, 0.6, 0.8, 1]$.

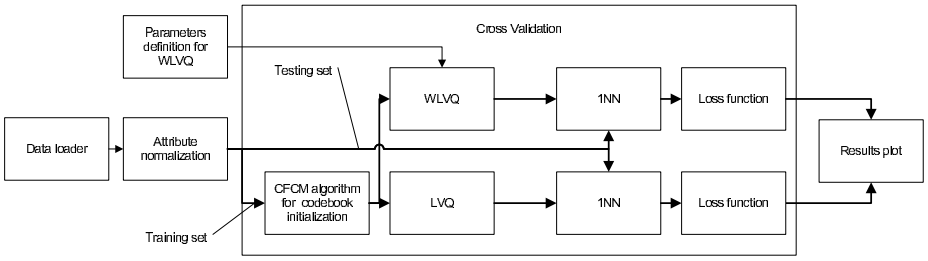


Fig. 1. Data flow diagram

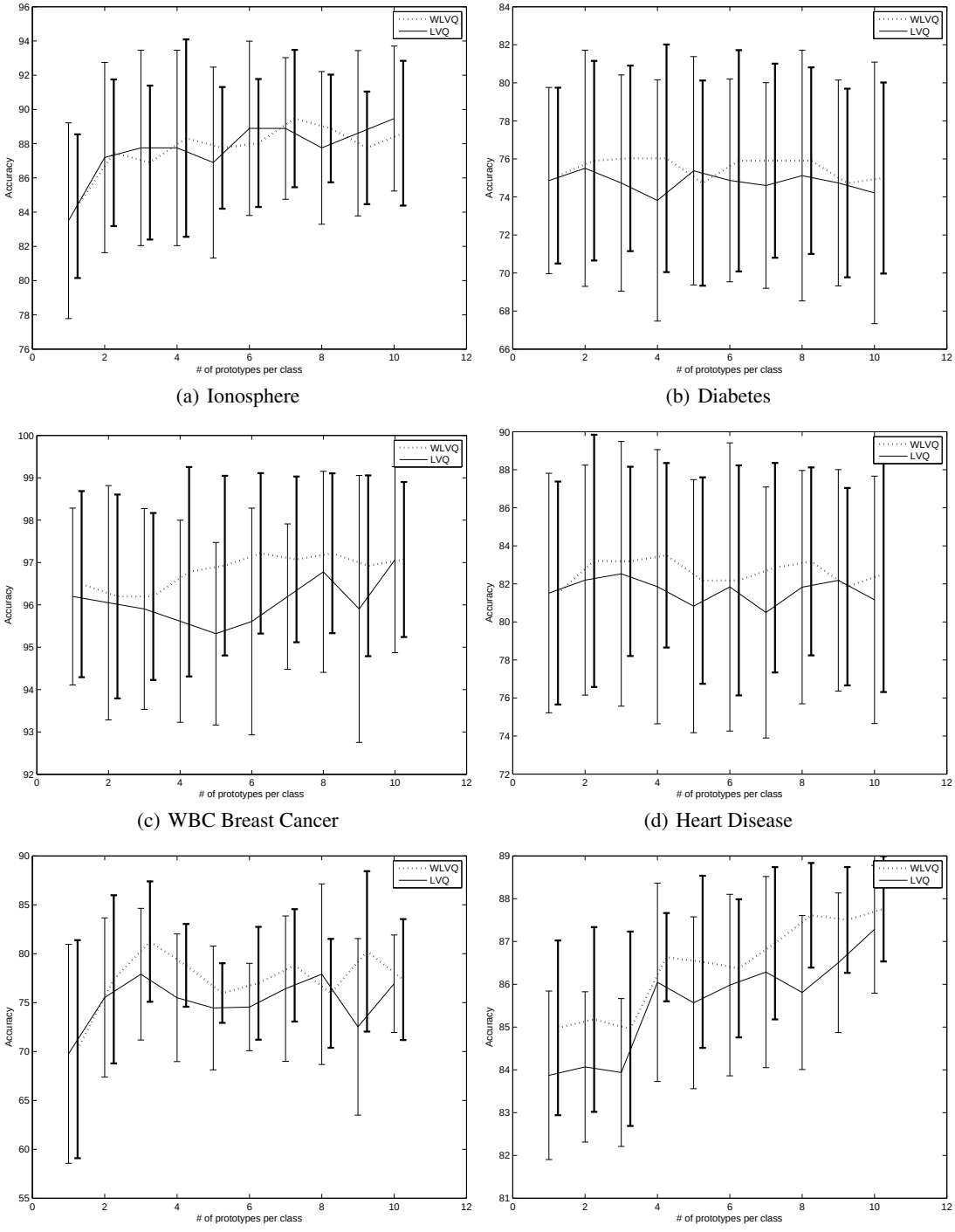
5.2 Results

Results are presented in Fig. 2. Each graph shows the average accuracy and variance as a function of the number of selected prototypes per class. In all plots solid thin line represents accuracy of the reference LVQ algorithm and the bold dotted line represent mean accuracy obtained with WLVQ algorithm.

Results obtained for the *ionosphere* dataset do not show significant improvement in accuracy, but WLVQ has somewhat reduced variance. For *pima diabetes* and *heart disease* datasets average results for the WLVQ algorithm are a bit better then obtained for LVQ, but high variance makes these differences insignificant. The difference can be seen in *sonar* and *spambase* datasets, where especially in the second dataset improvement is significant.

It is worth noting that for *heart disease*, *pima diabetes* and *breast cancer* LVQ finds one prototype per class that is optimal and adding more prototypes is not justified. This leads to a conclusion that linear methods should work well on these datasets, because they are equivalent to prototype classifier with just two prototypes, one per class. This is indeed the case, in all these cases linear solutions do not significantly differ from the best results that have been obtained for these datasets, and the same is true for the LVQ prototypes that may serve as logical rules based on similarity. Such good results may be at least partially attributed to the CFCM codebooks initialization.

For *sonar* 3 prototypes are needed, increasing accuracy by 10%. In this case linear methods are significantly worse than methods based on similarity. In case of *ionosphere* and the *spam* data linear methods completely fail and accuracy is significantly improved by adding more prototypes.



(a) Ionosphere (b) Diabetes

(c) WBC Breast Cancer (d) Heart Disease

(e) Sonar (f) Spambase Database

Fig. 2. Comparison of results obtained for LVQ classifier trained with and without instance weighting (both LVQs use CFCM codebooks initialization). Accuracy and standard deviation is presented as a function of the number of codebook vectors per class. Variance of LVQ is shown shfifted to the left of the variance of WLVQ.

6 Conclusions

In this paper context dependent clustering has been applied to the LVQ method. We have already shown [22] that the context-dependent clustering may be successfully used to initialize and train neural networks based on localized functions with clearly defined centers, such as the radial basis function networks with Gaussian functions [23, 24]. In this case all prototypes generated for network initialization have always been in the vicinity of decision borders. This led to a reduction of the number of prototypes and creation of low-complexity networks. Also application of conditional clustering to represent knowledge contained in the SVM model in a comprehensible way as prototype-based rules (P-rules) led to prototypes that served as support vectors and were placed near the decision border [20]. The SVM hyperplane was used to fit appropriate weights to selected prototypes. Definition of context (Eq. 8) used here is quite simple, and also focused on area around the decision border. Although this definition may be improved in many ways we have now been able to demonstrate that such approach is beneficial for convergence of the LVQ algorithm, leading to improvement of accuracy with reduced variance.

Similarity-based methods provide a powerful framework for knowledge discovery and data mining [2, 3]. They allow to build accurate prediction models in cases where decision borders are strongly non-linear, and to discover knowledge structures in form of similarity to prototypes, or P-rules. These rules are rarely used, although they agree with human intuition based on memory of similar cases seen earlier [25]. Explanatory power of such models may result from optimization of similarity measures optimized for each prototype and finding minimal set of such prototypes that allow for good generalization. P-rule systems make decisions in several ways, but most often the single nearest prototype rule is used. For some datasets a single prototype per class is sufficient, allowing for formulation of a simple classification rule: if the new case $\mathbf{x}$ is more similar to prototype $\mathbf{v}_1$ rather than to $\mathbf{v}_2$ it should be assigned to the same class as $\mathbf{v}_1$. Such rules may also be converted to fuzzy rules [26].

Selection and optimization of prototypes is an important part of methods that belong to the similarity-based framework. Without such selection the nearest neighbor methods are too slow for real-time applications, they may easily overfit the data, will be hard to use for very large datasets, and lead to data models that are hard to understand. An alternative to good initialization of a few LVQ prototypes is based on editing techniques. Using many prototypes that are pruned at a later stage may be more costly, but direct comparison of such methods with our approach remains to be done. It is clear that training on appropriately pruned data will be especially useful for very large datasets, giving hope to find solutions that are compact, accurate and easy to understand.

Performance of all similarity-based methods, or more generally all methods that calculate distances, including clustering algorithms, is strongly affected by noise that cannot be avoided in high-dimensional problems. To avoid it feature selection or feature aggregation methods should be used as the preliminary step. Simultaneous feature selection, weighting and prototype optimization is of particular importance, although it is still a great challenge. We hope that progress in this direction may be done along similar lines as used in this paper.

Acknowledgment

Authors wants thank to the Polish Ministry of Science and Higher Education to for the financial support of the project no. N516 442138.

References

1. Aha, D.W.: Lazy learning editorial. Artificial Intelligence Review **11** (1997) 1–6
2. Duch, W.: Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics **29** (2000) 937–968
3. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. Applied Mathematics and Computer Science **10** (2000) 101–120
4. Pękalska, E., Paclik, P., Duin, R.: A generalized kernel approach to dissimilarity-based classification. Journal of Machine Learning Research **2** (2001) 175–211
5. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
6. Sonnenburg, S., G.Raetsch, C.Schaefer, B.Schoelkopf: Large scale multiple kernel learning. Journal of Machine Learning Research **7** (2006) 1531–1565
7. Shakhnarovish, G., Darrell, T., Eds., P.I.: Nearest-Neighbor Methods in Learning and Vision. MIT Press (2005)
8. Arya, S., Malamatos, T., Mount, D.: Space-time tradeoffs for approximate nearest neighbor searching. Journal of the ACM **57** (2010) 1–54
9. Wilson, D.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Systems, Man and Cybernetics **2** (1972) 408–421
10. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning **38** (2000) 257–286
11. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery **6** (2002) 153–172
12. Bhattacharya, B., Mukherjee, K., Toussaint, G.: Geometric decision rules for instance-based learning problems. LNCS **3776** (2005) 60–69
13. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms. i. algorithms survey. Lecture Notes in Computer Science **3070** (2004) 598–603
14. Kohonen, T.: Self-organizing maps. Springer-Verlag, Heidelberg Berlin (1995)
15. Biehl, M., Ghosh, A., Hammer, B.: Dynamics and generalization ability of lvq algorithms. J. Mach. Learn. Res. **8** (2007) 323–360
16. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms. ii. results and comments. Lecture Notes in Computer Science **3070** (2004) 580–585
17. Duch, W., Grudziński, K.: Prototype based rules - new way to understand the data. In: IEEE International Joint Conference on Neural Networks, Washington D.C, IEEE Press (2001) 1858–1863
18. Blachnik, M., Duch, W., Wieczorek, T.: Selection of prototypes rules – context searching via clustering. Lecture Notes in Artificial Intelligence **4029** (2006) 573–582
19. Pedrycz, W.: Knowledge-Based Clustering: From Data to Information Granules. Wiley-Interscience (2005)
20. Blachnik, M., Duch, W.: Prototype rules from SVM. In: Rule Extraction from Support Vector Machines. Volume 80 of Springer Studies in Computational Intelligence. Springer (2008) 163–184

21. Asuncion, A., Newman, D.: UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html (2007)
22. Blachnik, M., Duch, W.: Building Localized Basis Function Networks Using Context Dependent Clustering. Lecture Notes in Computer Science **5163** (2008) 953–962
23. Haykin, S.: Neural Networks - A Comprehensive Foundation. Maxwell MacMillian Int., New York (1994)
24. Wilson, D.R., Martinez, T.R.: Heterogeneous radial basis function networks. In: Proceedings of the International Conference on Neural Networks. Volume 2. (1996) 1263–1276
25. Spivey, M.: The continuity of mind. New York: Oxford University Press. (2007)
26. Duch, W., Blachnik, M.: Fuzzy rule-based systems derived from similarity to prototypes. Lecture Notes in Computer Science **3316** (2004) 912–917