Building Localized Basis Function Networks Using Context Dependent Clustering

Marcin Blachnik[1,2] and Włodzisław Duch[3]

[1] Silesian University of Technology, Electrotechnology Department, Katowice, Krasińskiego 8, Poland; marcin.blachnik@polsl.pl
[2] Helsinki University of Technology, Adaptive Informatics Research Centre, P.O. Box 5400, 02015 Espoo, Finland
[3] Department of Informatics, Nicolaus Copernicus University, Grudziądzka 5, Toruń, Poland; Google: W. Duch

Abstract. Networks based on basis set function expansions, such as the Radial Basis Function (RBF), or Separable Basis Function (SBF) networks, have non-linear parameters that are not trivial to optimize. Clustering techniques are frequently used to optimize positions for localized functions. Context-dependent fuzzy clustering techniques improve convergence of parameter optimization, leading to better networks and prototype-based logical rules that provide low-complexity models of data.

1 Introduction

Basis set expansions of functions have been studied in approximation theory since the XIX century by such great mathematicians as Weierstrass, Chebyshev, Runge, Volterra, Lebesgue, Markov, Hermite, Fourier and many others. Multidimensional expansions into product of various orthogonal polynomials, splines, or Gaussian functions have been used in quantum mechanics from its beginning. In neural networks radial basis function networks (RBF) [2] have gained great popularity, providing approximations of multidimensional function as a linear combination of radial functions $\Phi(\mathbf{X}) = \sum_i W_i G(||\mathbf{X} - \mathbf{R}_i||)$. Although many radial functions exist (see the survey of neural transfer functions [11]) Gaussian functions are the most popular. One of the reasons is that n-dimensional Gaussian functions belong to the few radial functions that are separable, that is they may be presented as a product of n one-dimensional components. Such products may be identified either with "and" conditions in aggregation of fuzzy membership functions for different features, or with Naive Bayes estimation of multidimensional probability by a product of one-dimensional Gaussian distributions.

Separable basis function (SBF) networks, such as the Feature Space Mapping network [9] use Gaussians or other one-dimensional functions to create approximations to multidimensional probability distributions $\Phi(\mathbf{X}) = \sum_I W_I \phi_{i1}(x_1)\phi_{i2}(x_2)\ldots\phi_{in}(x_n))$, where the multi-index $I = \{i1, i2, \ldots in\}$. In quantum mechanics and quantum chemistry [18] these one-dimensional functions are called orbitals and the total function to be approximated is called the wavefunction. SBF networks have more interesting properties (in particular all similarity-based methods with additive distance functions are equivalent to SBF networks [8]), although they are not so popular as the RBF networks. Local representations are based on cognitive inspirations [21, 9], facilitating "intuitive" representation of knowledge based on similarity (measured by neural output) to prototypes, or reference objects described by probability density distributions in their feature spaces. Classification or approximation is then done using rules based on similarity to reference objects. In this paper only networks that use localized basis function (LBF) expansions (output is larger than some threshold value in a finite region of input space) are considered, and in empirical tests only Gaussian basis functions (GBF) have been used, making these networks equivalent to the popular RBF networks. There is a tendency to call all basis set expansion networks RBF networks, even if the functions used are not radial. To avoid confusion the term LBF shall be used when localization is important, and more precise GBF when Gaussian functions are used.

In multilabel classification the goal is to discriminate between n-dimensional vectors $\mathbf{X} = [x_1, x_2, \ldots x_n]^T$ that carry c symbolic or numeric class labels $\mathbf{Y} = [y_1, y_2, \ldots y_c]$. In contrast to Bayesian formulation [2] class labels do not need to be interpreted as probabilities, and thus one vector may be a member of several classes. The classification system has some parameters that are optimized on a training dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ of N pairs of feature vectors $\mathbf{X}$ and their class labels $\mathbf{Y} = \mathbf{Y}(\mathbf{X})$, with the goal of obtaining good performance on new data sampled from the same probability distributions.

In the LBF network expansion $g(\mathbf{X}; \mathbf{W}, \mathbf{R}) = \sum_i W_i G_i(\mathbf{X}, \mathbf{R}_i)$ linear coefficients $\mathbf{W}$ may be obtained using standard RBF techniques [2], but finding non-linear parameters $\mathbf{R}_i$, such as positions and dispersions of Gaussian functions, is more difficult. Most frequently RBF network training starts from clustering of the whole training set, placing the hidden units in the centers of obtained clusters. The hidden-output linear connections are then trained to minimize some cost function. This process can also be viewed as nonlinear mapping of the input data by localized functions to some image space, and then creating linear model in the image space. Kernel machines, in particular support vector machines (SVM) [23], do not perform this transformation using explicitly defined transfer functions, but the idea is the same.

In the LBF networks used for classification problems clustering approach may lead to non optimal position of the hidden neurons, hence the number of hidden neurons required for sufficient accuracy may become unnecessarily large. Pruning techniques are used to remove less important neurons [2] and even to merge several neurons [14]. Pruning may be followed by re-optimization of parameters, adding more neurons, and repeating the whole cycle [1] to reduce the network and improve its generalization, but additional computational cost are involved.

SVM improves generalization by increasing the classification margin. Support vectors (SV) are selected fom the input vectors that define the shape of decision border. Support vectors are spread around the SVM decision hyperplane. In contrast to many linear discrimination techniques that use all data vectors to define the hyperplane [25] in SVM vectors that are far from the decision border do not influence its final shape. In the Support Vector Neural Training algorithm [7] vectors that are far from the decision border (and therefore cause maximum or minimum activation of neurons) are gradually removed from the training set, leaving only support vectors close to the decision border. In this contribution context dependent clustering is used to restrict the subspace where the decision border is defined. Although SVM may be used for extraction of prototype-based rules (P-rules) [3] more interesting results may be obtained using context dependent clustering.

In the next section the methods of building and optimizing LBF networks are discussed, in section 3 the basic context-dependent fuzzy C-means (CFCM) algorithm and the method of defining the context (section 4) are explained. Empirical results, presented in section 5, compare LBF networks based on the classical clustering and the context dependent clustering. The last section contains a discuss of the results and shows some potential applications of the proposed approach as a tool for creating prototype-based rule systems (P-systems).

2 Building LBF network

Typical LBF network consist of one hidden layer and an output layer. Neurons in the hidden layer use localized transfer functions, for example in the RBF networks a distance-based activation $||\mathbf{X}-\mathbf{R}||$ between input vector $\mathbf{X}$ and the reference vector $\mathbf{R}$ that defines the center of the function. In the most common case neurons of the hidden layer have Gaussian transfer function. However, many other localized non-radial functions may also be used, and many radial functions are non-local [11], therefore we shall use the LBF name rather than RBF.

The output layer implements a linear combination of signals generated by the hidden layer, leading to a decision function:

$$g(\mathbf{X};\mathbf{W},\mathbf{R}) = \sum_{i=1}^{K} W_i G_i(\mathbf{X},\mathbf{R}_i) \tag{1}$$

where K is the number of hidden neurons and $\mathbf{W}$ are weights of the output layer. In case of a generalized Gaussian Basis Function expansion the particular parameterization of localized functions involves:

$$g(\mathbf{X};\mathbf{W},\mathbf{R}) = \sum_{i=1}^{K} W_i \exp\left(-||\mathbf{X}-\mathbf{R}_i||^2_{\Sigma_i}\right) \tag{2}$$

where $\mathbf{R}_i$ is the position of i-th prototype (location of the center of the hidden unit in the input space), and a local Mahalanobis distance function $||\cdot||_{\mathbf{\Sigma}_i}$ with the covariance matrix $\mathbf{\Sigma}_i$ is calculated in some neighborhood around the center:

$$||\mathbf{X}-\mathbf{R}_i||^2_{\Sigma_i} = (\mathbf{X}-\mathbf{R}_i)^T \mathbf{\Sigma}^{-1}(\mathbf{X}-\mathbf{R}_i) \tag{3}$$

Covariance matrix could be replaced with a matrix of adaptive parameters learned from the training data, providing scaling and rotations for the shapes of local decision boarders. Although it is possible to rotate Gaussian function in n-dimensions using only n parameters [11] in practice almost always diagonal matrix $\mathbf{\Sigma}$ is used, providing distance scaling factors s_i. Training of such GBF networks requires adaptation of positions $\mathbf{R}_i$, scaling factors s_i and weights $\mathbf{W}$ of the output layers. As suggested in [24] there are three possible ways to build such networks:

1. One step process, each training data vector used to define one hidden unit, training resticted to the output layer;
2. Two step training, interchanging training of non-linear and linear parameters:
   - adjust location and scaling parameters of hidden units using some form of supervised clustering or decision trees, independently of the network outputs;
   - adjust output layer weights minimizing some cost function, usually mean square error:

     $$E(\mathcal{D};\mathbf{W},\mathbf{R}) = \sum_{i=1}^{N} ||\mathbf{Y}_i - g(\mathbf{X}_i;\mathbf{W},\mathbf{R})||^2 \tag{4}$$

     The output weights may be then determined using the pseudoinverse matrix:

     $$\mathbf{W} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{Y} \tag{5}$$

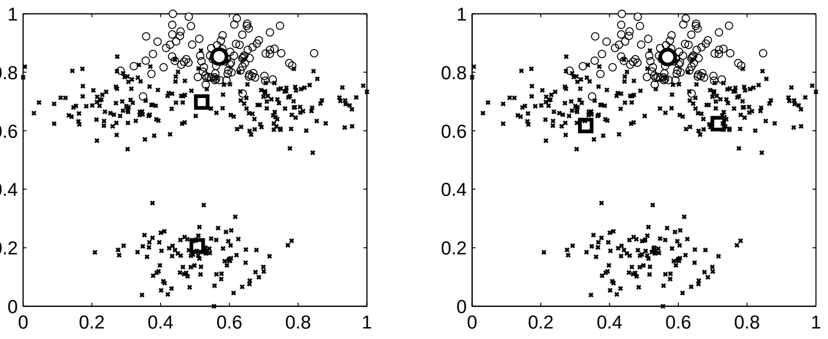
     where $G_{ij} = G(\mathbf{X}_i;\mathbf{R}_j)$ is the rectangular matrix $N \times K$ of similarities between the prototypes $\mathbf{R}$ and the training data $\mathbf{X}$.
3. Three step learning, where the first two steps described above are used for initialization, followed by simultaneous adaptation of all network parameters using gradient descend or other methods.

Unfortunately the error function $E(\mathcal{D};\mathbf{W},\mathbf{R})$ has multiple local minima making the gradient optimization quite difficult. Results may be significantly improved if a proper initialization of the network is found. This shall be the focus of the approach developed below.

Another problem facing the LBF neural networks is to determine the optimal number of hidden units. The simplest solution is to use all vectors to define a large number of hidden unites, but this leads to many problems with data overfitting, convergence difficulties due to ill-conditioned $\mathbf{G}$ matrix and waste of computational resources. Reduction of the number of hidden nodes to $K \ll N$ is a form of regularization that increases generalization of LBF networks. Other advantages of this approach include easier interpretation of the functions implemented by such networks. For example, in [13, 9, 17] relation between basis set expansion networks and fuzzy rule based systems (F-rules) has been analyzed, showing that for separable neural transfer functions (and thus a few types of radial basis functions) they are equivalent. Fuzzy rules are useful if they are comprehensible and generalize well, therefore a small number of neurons is highly desired. This is equally true for the systems based on P-rules [10, 8], that are equivalent to networks with different types of functions implemented by their hidden nodes.

Support Vector Machines provide good inspiration for construction of optimal shapes of decision borders that should be based on prototypes placed in a similar way as SVs around the border area. Classical (unsupervised) clustering algorithms search analyze all data in the whole input space without information about class distribution. Clustering each class independently may not be a good solution, leading to centers of clusters that may be far from optimal, as show in Fig. 1. Optimization methods from the LVQ family [16], or other gradient based methods used in the network training, cannot move such prototypes to areas where they could take an important role in the decision process.

Context-dependent clustering can be used with properly defined context to find much better positions of prototypes. Such approach has been previously applied with very good results for training of the prototype-based rule system with the nearest neighbor rule [5].



(a) Prototypes after FCM clustering. (b) Prototypes after CFCM clustering.

Fig. 1. Comparison of prototype positions obtained from the FCM and CFCM clustering algorithms for the two class problem.

3 Conditional Fuzzy C-Means

Context-dependent clustering methods may be realized in several ways, for example using the Conditional Fuzzy C-Means method (CFCM) [22], an extension of the standard Fuzzy C-Means (FCM) clustering algorithm [12]. CFCM clusters data under some external conditions defined for every training vector. These conditions are specified by an external variable z_k which is associated with each training vector $\mathbf{X}_k$, providing additional information that should be taken into account during clusterization. The strength of the clustering relations between the external variable z_k and the data vector $\mathbf{X}_k$ is defined by $f_k = \mu_A(z_k) \in [0, 1]$, where $\mu_A(z_k)$ may be interpreted as a membership function defined by particular context A. To find clusters that are near the decision border the external variable z is defined as an estimated distance between data vector $\mathbf{X}$ and the decision border.

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

$$J_m(\mathbf{U}, \mathbf{P}) = \sum_{i=1}^{K} \sum_{j=1}^{N} (u_{ji})^m \left\| \mathbf{X}_j - \mathbf{P}_i \right\|_A^2 \tag{6}$$

where K clusters are centered at $\mathbf{P}_i, i = 1..K$, $m > 1$ is a parameter, and the matrix $\mathbf{U} = (u_{ji})$ is a $K \times N$-dimensional membership matrix with elements $u_{ik} \in [0, 1]$

defining the degree of membership of the $\mathbf{X}_j$ vector in the i-th cluster. Conditional clustering is achieved by enforcing 3 conditions on the matrix $\mathbf{U}$:
1^o Each vector $\mathbf{X}_j$ belongs to the i-th cluster to a degree that is between 0 to 1:

$$\underset{1\le i\le K}{\forall}\ \underset{1\le j\le N}{\forall} u_{ji} \in [0,1] \tag{7}$$

2^o Sum of the membership values of j-th vector $\mathbf{X}_j$ in all clusters is equal to f_j

$$\underset{1\le j\le N}{\forall} \sum_{i=1}^{k} u_{ji} = f_j \tag{8}$$

3^o Clusters must not be empty.

$$\underset{1\le i\le K}{\forall} 0 < \sum_{j=1}^{N} u_{ji} < N \tag{9}$$

The cost function (6) is minimized under these conditions by placing centers at:

$$\underset{1\le i\le k}{\forall} \mathbf{P}_i = \sum_{j=1}^{N} (u_{ji})^m x_j \Bigg/ \sum_{j=1}^{N} (u_{ji})^m \tag{10}$$

$$\underset{\substack{1\le i\le K\\ 1\le j\le N}}{\forall} u_{ji} = f_j \Bigg/ \sum_{l=1}^{K} \left(\frac{\|\mathbf{X}_j - \mathbf{P}_i\|}{\|\mathbf{X}_j - \mathbf{P}_l\|}\right)^{2/(m-1)} \tag{11}$$

4 Determining the context

Conditional clustering has been used previously to generate P-rules [5]. Searching for optimal position of prototypes for P-rules is a difficult problem; moreover, a balance between the number of prototypes (simplicity of the rules) and the accuracy of the whole system is very important. Irrelevant prototypes, that is prototypes that do not have any influence on classification, should be removed. Such prototypes are usually far from decision borders. Therefore prototypes that lie close to vectors from one of the opposite classes should be preferred, as they should be close to the possible decision border.

To determine position of prototypes using conditional clustering algorithm an external coefficient z_k is defined, evaluating for a given vector $\mathbf{X}_k$ the ratio of a scatter between this vector and all vectors from the same class, divided by a scatter for all vectors from the remaining classes:

$$z_k = \sum_{j,\omega(\mathbf{X}_j)=\omega(\mathbf{X}_k)} \|\mathbf{X}_k - \mathbf{X}_j\|^2 \Bigg/ \sum_{l,\omega(\mathbf{X}_l)\neq\omega(\mathbf{X}_k)} \|\mathbf{X}_k - \mathbf{X}_l\|^2 \tag{12}$$

Here $\omega(\mathbf{X}_k)$ denotes class label of the vector $\mathbf{X}_k$. For quite inhomogeneous data summation may be restricted to local neighborhoods of the $\mathbf{X}_k$ vector. The z_k coefficients are then normalized to be in the range [0,1]:

$$z_k \Leftarrow \left(z_k - \min_k (z_k) \right) \Big/ \left(\max_k (z_k) - \min_k (z_k) \right) \tag{13}$$

Normalized z_k coefficients reach values close to 0 for vectors inside large homogenous clusters, and close to 1 if the vector z_k 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 effects related to multiple density peaks in non-homogenous data distributions a cut-off point for distance of the order of standard deviation of the whole data could be introduced, although in our tests it was not necessary. These normalized coefficient determine the external variable which then is transformed into appropriate weights used as context or condition for CFCM clustering. Such weighting can be also understood as assigning appropriate linguistic term in the fuzzy sense. In the algorithm used below a Gaussian weighting was used:

$$f_k = \exp\left(\frac{z_k - \mu}{\sigma^2} \right) \tag{14}$$

with the best parameters in the range of μ =0.6–0.8 and σ =0.6–0.9, 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 a larger μ closer to the decision borders. The range in which they are sought is determined by the σ parameter.

5 Experiments and results

5.1 Datasets

To verify usefulness of proposed methodology for creating GBF networks several experiments have been performed, taking four rather different real-world benchmark datasets from the UCI repository [20].

1. Cleveland Heart Disease describes 303 patients, but 7 cases with missing attributes are usually removed, leaving 297 cases that initially have been divided into 5 categories (according to severity of the heart disease), but are usually used in benchmark problems with two classes, sick (139 samples) and healthy (164 samples), with 14 most useful attributes (nominal, binary, ordinal and real) selected from 76.
2. Pima Indians Diabetes - diagnosis of diabetes among Pima Indians using descriptors provided by the World Health Organization. Dataset consist of 768 samples, described by 8 attributes, with 500 samples for healthy and 268 samples for sick people. All attributes are numerical.
3. Ionosphere data describe radar signals reflected from the free electrons in the ionosphere by sending 17 pulses, each described by two components, therefore there are 34 real attributes. The dataset consist of 351 vectors classified into two categories,

depending on the state of the ionosphere. Because the second attribute of the original dataset has all values equal to 0, it has been removed from data together with the first binary attribute.

4. Appendicitis is a small dataset of 8 medical tests performed on 106 patients suspected of appendicitis. Each patient had a biopsy to verify the need for surgery. 85 cases required treatment, and 21 did not. From the original dataset one attributes with missing values has been removed, so the final dataset consist of 106 samples with 7 attributes each.

5.2 Testing methodology and obtained results

The primary goal of these tests is to compare the effects of the context-based clustering with standard clustering algorithms on the initialization and performance of the GBF networks. The best results with the optimal number of neurons determined in the crossvalidation procedure are reported. Dispersions of the Gaussian neurons have been selected using a search procedure, and a search has been made to find appropriate parameters for the context variable in the CFCM clustering case. In all tests the data has been normalized to keep each variable in the $[0, 1]$ range.

Generalization abilities of all methods applied to these datasets have been estimated using the 10-fold crossvalidation tests. Both accuracy of the model and its variance are collected in Table 1. Low variance shows good stability of the model, as some part of variance is due to the partitioning of data by the crossvalidation procedure. Stable models with low variance obtained in cross validation test should be preferred over more accurate but also more fluctuating ones. To select a good model a simple formula that minimizes estimated error minus the standard deviation as suggested in [15], or with a more sophisticate statistical tests.

Table (1) shows the results for selected datasets. For comparison also SVM and P-rules system results are included. SVM was taken from libsvm implementation [19], using Gaussian kernels with C and σ parameters optimized using a grid search procedure to test performance of over 40 models for different values of these parameters. P-rules system (CFCM-LVQ type) based on CFCM clustering followed by LVQ prototype optimization technique [5]. It is also important to noticed that P-rules model has build in feature selection algorithm, while for all other models feature selection haven't been performed, what is also important P-rules model was designed to find the possible simplest data representation what can be seen in the number of selected prototypes.

6 Discussion of results and further perspectives

The context-dependent initialization of the localized basis set expansion networks proposed in this paper is a simple modification of the commonly used fuzzy C-means method. Experiments with GBF networks showed for all datasets and for all number of hidden neurons that the context-dependent clustering is more robust then the standard clustering algorithm used to initialize and train neural networks. The same approach is applicable to all basis set expansion networks that use localized functions with clearly defined centers. Surprisingly good results have been obtained in comparison to the SVM

	GBF FCM		GBF CFCM		SVM		P-rules	
	acc±std	k	acc±std	k	acc±std	# SV	acc±std	# Proto
Ionosphere	3.41±0.56	40	2.29±0.71	60	4.28±0.64	113.70	12.18±1.41	6
Cleveland	16.12±2.50	5	15.13±2.39	5	16.79±1.98	131.80	17.2±2.15	2
Pima Indians	22.14±1.56	55	21.88±1.54	45	21.75±1.52	428.70	23.04±1.17	2
Appendicitis	10.55±3.5	3	10.55±3.75	3	11.46±3.57	30.00	16.23±5.8	2

Table 1. Accuracy of the GBF network with k hidden neurons per class trained by FCM and CFCM algorithms, ooptimized SVM with Gaussian kernels and P-rules system.

model that is also based on Gaussian kernels, and is usually one of the best classifiers. The CFCM GBF networks proved to be not only more accurate, have in most cases lower variance, but also require much lower number of prototypes then support vectors obtain by SVM. Although sparse SVM versions may be used for data understanding [6] results are not so good and models are not so simple as those obtained from properly trained GBF networks. Although the datasets analyzed in the previous section has been rather diverse, all of them are relatively simple; this has been done intentionally to check if simple models with explanatory power may be generated. More complex data may not have simple models and thus black box predictors may be acceptable.

Performance of all method that calculate distances is strongly affected by noise in high-dimensional problems, and that includes also clustering algorithms. In this case feature selection or feature aggregation methods may be a necessary preliminary step. In contrast to SVM and most other kernel methods, basis expansion networks with full optimization of individual node parameters are capable of multi-resolution, providing complex shapes of decision border in selected areas of the input space where it is really needed, and simple description in other areas. Small number of neurons has been sufficient for Diabetes and Appendicitis datasets, but for the Ionosphere large number of neurons is needed. Tests are being conducted on more complex datasets to see how well CFCM training of GBF and other LBF networks will compare with other methods.

As suggested in Sec. 2 LBF networks may be used not only as a black box, but also to understand data structures, helping to understand the reasons for decision made by the model. Separable Basis Function networks may be interpreted as equivalent fuzzy system if the full covariance matrix mixing different features is not used. Such restriction does not occur in systems that extract prototype-based rules (P-rules), where the explanatory power of the model comes from the analysis of a set of characteristic examples taken as prototypes, and optimization of individual similarity measures [5, 4, 8, 10]. The goal is to find minimal set of reference cases that allow for good generalization. P-rules systems may take decisions in several ways, but most often the single nearest prototype rule is used. In RBF, LBF or SBF networks the model is more complex, using instead of the $\max(\cdot)$ aggregation function a weighted sum of contributions from all hidden nodes. Such model is more flexible, but in effect more appropriate for approximation problems than for data understanding. Using appropriate normalization SBF networks may still be viewed as P-rule systems of the Takagi-Sugeno fuzzy model type.

In the CFCM approach to LBF networks initialization and training, all prototypes lie in the vicinity of decision borders. Comparing with the standard approach to the training of such networks this leads to a reduction of unnecessary prototypes, which is highly desirable from the point of view of comprehensibility. However, model selection for LBF has to be done carefully, to avoid placing many prototypes close to the decision borders, when one prototype far from the border will be sufficient. This is not a big problem in P-systems with the nearest neighbor decision rule, but in LBF the size of the support of localized functions has to be explicitly optimized. One drawbacks of the CFCM algorithm is the need to determine correct mean and sigma values in Eq. 14. Although default parameters work well in most cases for some data a grid search for optimal values may be important. This leads to an increase of computational complexity that may become a problem for very large datasets, although for most datasets learning takes very little time.

References

1. R. Adamczak, W. Duch, and N. Jankowski. New developments in the feature space mapping model. In *Third Conference on Neural Networks and Their Applications*, pages 65–70, Kule, Poland, Oct 1997.
2. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.
3. M. Blachnik and W. Duch. *Prototype rules from SVM*, volume 80 of *Springer Studies in Computational Intelligence*, pages 163–184. Springer, 2008.
4. M. Blachnik, W. Duch, and T. Wieczorek. Probabilistic distance measures for prototype-based rules. In *Proc. of the 12th Int. Conference on Neural Information Processing (ICONIP'2005)*, pages 445–450, Taiwan, 2005. Taipei University Press.
5. M. Blachnik, W. Duch, and T. Wieczorek. Selection of prototypes rules – context searching via clustering. *Lecture Notes in Artificial Intelligence*, 4029:573–582, 2006.
6. J. Diederich, editor. *Rule Extraction from Support Vector Machines*, volume 80 of *Springer Studies in Computational Intelligence*. Springer, 2008.
7. W. Duch. Support vector neural training. *Lecture Notes in Computer Science*, 3697:67–72, 2005.
8. W. Duch and M. Blachnik. Fuzzy rule-based systems derived from similarity to prototypes. In N.R. Pal, N. Kasabov, R.K. Mudi, S. Pal, and S.K. Parui, editors, *Lecture Notes in Computer Science*, volume 3316, pages 912–917. Physica Verlag, Springer, New York, 2004.
9. W. Duch and G. H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.
10. W. Duch and K. Grudziński. Prototype based rules - new way to understand the data. In *IEEE International Joint Conference on Neural Networks*, pages 1858–1863, Washington D.C, 2001. IEEE Press.
11. W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2:163–213, 1999.
12. F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, 1999.
13. J-S. R. Jang and C.T. Sun. Functional equivalence between radial basis function neural networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4:156–158, 1993.
14. N. Jankowski. Approximation and classification in medicine with incnet neural networks. In *Machine Learning and Applications. Workshop on Machine Learning in Medical Applications*, pages 53–58, Greece, Jul 1999.

15. N. Jankowski and K. Grąbczewski. Handwritten digit recognition – road to contest victory. In *IEEE Symposium on Computational Intelligence in Data Mining*, pages 491–498. IEEE Press, 2007.
16. T. Kohonen. *Self-organizing maps*. Springer-Verlag, Heidelberg Berlin, 1995.
17. L. Kuncheva. On the equivalence between fuzzy and statistical classifiers. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 15:245–253, 1996.
18. I.N. Levine. *Quantum Chemistry*. Prentice Hall; 5th edition, 1999.
19. K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.
20. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998-2004. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
21. E. M.Pothos. The rules versus similarity distinction. *Behavioral and Brain Sciences*, 28:1–49, 2005.
22. W. Pedrycz. Conditional fuzzy c-means. *Pattern Recognition Letters*, 17:625–632, 1996.
23. B. Schölkopf and A.J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.
24. F. Schwenker, H.A. Kestler, and G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14:439–458, 2001.
25. A.R. Webb. *Statistical Pattern Recognition*. J. Wiley & Sons, 2002.