Heterogeneous adaptive systems.

Włodzisław Duch and Krzysztof Grąbczewski
Department of Informatics, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland; www.phys.uni.torun.pl/kmk

Abstract - Most adaptive systems are homogenous, i.e. they are built from processing elements of the same type. MLP neural networks and decision trees use nodes that partition the input space by hyperplanes. Other types of neural networks use nodes that provide spherical or ellipsoidal decision borders. This may not be the best inductive bias for a given data, frequently requiring large number of processing elements even in cases when simple solutions exist. In heterogeneous adaptive systems (HAS) different types of decision borders are used at each stage, enabling discovery of a most appropriate bias for the data. Neural, decision tree and similarity-based systems of this sort are described here. Results from a novel heterogeneous decision tree algorithm are presented as an example of this approach.

I. Introduction

There is no free lunch, no single method that will achieve the best results in all cases [1]. This is due to a different inductive bias that various types of data may require. Humans seem to avoid this problem by focusing on different features for different objects in the categorization process. A single cortical column in the brain provides many types of microcircuits that respond in a qualitatively different way to the incoming signals [2]. Other cortical columns may combine these responses in a perceptron-like fashion to enable complex discriminations. At the level of higher cognition brains do not recognize all objects in the same feature space. Even within the same sensory modality several complex features are selected, allowing to distinguish one class of objects from another.

In contrast to human categorization most pattern recognition systems implicitly assume that classification is done using the same features in all regions of the input space. Memory-based techniques use single distance (or similarity) function to distinguish all objects, statistical or neural methods provide hyperplanes (multilayer perceptrons) or Gaussian functions (Radial Basis Function networks) for discrimination, but rarely both. Decision trees are usually univariate, employing a decision rule for the threshold value of a single feature, partitioning the input space into hyperrectangles. Multivariate decision trees provide several hyperplanes at high computational cost. Support Vector Machines use one kernel globally optimized for a given dataset [3]. All these systems may be called "homogenous" since they search for a solution providing the same type of elements, the same type of decision borders in the whole feature space. Committees of the homogenous systems are frequently used to improve and stabilize results [4]. Combining systems of different types in a committee is a step towards heterogeneous systems that use different types of decision borders, but the model becomes quite complex and is difficult to understand.

Although many adaptive homogenous systems are universal approximators [1] and may model all kinds of data accurately they will not discover simple class structures. For example, if a feature space is divided into two classes by a hyperplane plus a sphere with a center placed on this hyperplane most of the computational intelligence (CI) systems will model densities or classification probabilities using a large number of processing elements. Simple descriptions are desirable for two reasons. First, to understand the data, discover class structure or type of sources that generate signals. Second, because the Ockham razor principle gives simpler systems a better chance for good generalization. In this paper we shall consider heterogeneous adaptive systems (HAS), i.e. systems that use processing elements of qualitatively different types. Such systems include: neural networks that use different types of transfer functions, selecting them from a pool of different functions or optimizing flexible transfer functions to discover the most appropriate bias in different regions of the feature space; similarity-based methods that use different similarity function associated with each reference vector; decision trees that use several types of tests, providing qualitatively different decision borders. Heterogeneous elements may also be created within homogenous systems by various transformations of the input data.

II. Heterogeneous adaptive systems.

Many computational intelligence systems used for classification and approximation are constructed from simple building blocks or processing elements (PE). Three examples of homogenous systems, i.e. systems using one type of PE only, include: feedforward neural networks (NN), with the same type of transfer function realized by each node; decision trees (DT), with the same type of test performed at each node to split the data; similarity-based methods (SBM) with global metric function used with reference vectors. Each processing element contributes a discriminating function with specific decision region. The final decision regions are created by selection or a (non)linear combination of results from individual nodes. In homogenous systems discriminating functions (or basis functions) are always of the same type: hyperplanes in decision trees, perceptron networks and SVMs, hyperellipsoids in SBM systems or in Radial Basis Function (RBF) networks based on Gaussian functions, etc. Combinations of simple decision regions create new, complex decision borders of arbitrary shapes, but it may require large number of parameters to achieve desired approximation accuracy. Complex data models do not allow to understand the structure of the data [5] and discovering inductive bias behind the process that created the data.

Although homogeous systems may have universal approximation properties, they may not be the most appropriate for a given data even if many types of nodes are tried. In [6] we have estimated the minimal number of parameters that a feedforward neural network needs to separate a single hyperspherical data region in N dimensions. For Gaussian functions it is of the order of $O(N)$, while for sigmoidal functions (hyperplanes) it is at least $O(N^2)$. On the other hand separation of the area between the origin of the coordinate system and the $(1,1,...1)$ plane from the area outside requires a single hyperplane, while any classifier based on Gaussian functions will use $O(N^2)$ parameters offering rather poor approximation (a single rotated large Gaussian is sufficient). Both data distributions are rather difficult for univariate decision trees or SBM systems. Combination of these two cases, with the separating hyperplane and the center of Gaussian function placed in its center $1/N(1,1,...1)$, is difficult for all homogenous classifiers. To find the simplest solution a heterogeneous classifier, using processing elements that provide different discriminating functions, is needed.

III. Heterogeneous neural networks

A survey of many functions [6] and a systematic taxonomy of these functions [7] has been presented recently. Networks based on rational functions, various spline functions, tensor products, circular units, conical, ellipsoidal, Lorentzian functions and many others were described in the literature (see references in [6]). Support Vector Machines [3] are used with different kernel functions. Both neural networks and SVMs are wide margin classifiers. At least part of the SVM success is due to the selection of the best kernel for a given data, although for simple benchmark problems the differences between results obtained with different kernels (or different transfer functions) may be small.

All these systems are homogenous, using one type of kernel or transfer function in a given architecture. Three basic ways to create heterogeneous adaptive systems of the neural network type (HAS-NN) exist.

- Construct neural network adding new basis functions selected from a pool of candidate functions .
- Start with large network that uses different transfer functions and prune it.
- Use highly flexible transfer functions containing internal parameters that are optimized, evolving different transfer functions as a result of training.

A constructive method that selects the most promising functions from a pool of candidates in RBF-like architecture and adds it to the growing network has been introduced in [8]–[10]. Each candidate node using different transfer function should be trained and the most useful candidate is added to the network. In effect several networks are created, trained in parallel and evaluated. The Optimal Transfer Function (OTF) network [8] is based on the incremental network (IncNet) network architecture [11]. It uses pruning techniques and statistical criteria to determine which neurons should be selected. For the XOR problem classical two-neuron solutions were discovered using localized, Gaussian-type functions as well as non-local, sigmoidal functions. With more complex conical transfer functions [6] it has also discovered a single neuron solution to the XOR problem; in this case Gaussian output function was combined with inner product activation function. The OTF network has also found optimal solutions to the hyperplane plus sphere problem.

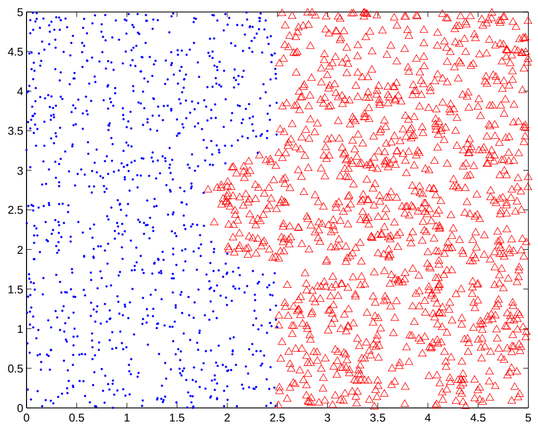

Fig. 1. Hyperplane plus sphere data in two dimensions.

A heterogeneous version of the Feature Space Mapping [12], a neurofuzzy ontogenic network that selects the type of a separable localized transfer function for the next node from a pool of several types of functions, has been applied to the same artificial problems. This network has found an interesting solution to the hyperplane-sphere problem, selecting one rectangular function and one Gaussian function. Variance was rather large and in 5-dimensional cases quite often Gaussian functions were selected, although the simplest solution has also appeared. This network applied to Wisconsin Breast Cancer data [13] created a very simple solution, with 2 rectangular nodes only [9].

In the second approach starting from a network that already contains several types of transfer functions pruning or regularization techniques are used to reduce the final number of functions [8]. Initially the network is too complex but at the end only the functions that are best suited to solve the problem are left. Many combinations of functions may create solutions of similar complexity, therefore large variance of the results, especially for small data, may be expected.

In the third approach neural transfer functions are taken from families of functions parameterized for greatest flexibility. For example, an extension of the conic activation function is defined by:

$$A_C(\mathbf{X};\mathbf{W},\mathbf{R},\mathbf{B},\alpha,\beta) = \alpha I(\mathbf{X}-\mathbf{R};\mathbf{W}) + \beta D(\mathbf{X};\mathbf{R},\mathbf{B}) \qquad (1)$$

where $I(\mathbf{X};\mathbf{W}) = \sum_{j=0}^{N} W_j X_j$ is the weighted activation function and $D(\mathbf{X};\mathbf{R},\mathbf{B})$ is a distance function. Used with the sigmoidal output function this function changes smoothly from a localized Gaussian (with independent biases) to a soft hyperplane type. Several other functions of such type have been presented in [6], [7], but so far they have not been tried in practice. Sebald and Chellapilla [14] have used evolutionary programming techniques to train a network with 3 types of transfer functions, obtaining rapid convergence.

IV. Heterogeneous similarity-based methods

In the similarity-based methods (SBM) [15] a set of reference vectors and a procedure to estimate similarity are defined. In homogenous SBM models similarity or dissimilarity of objects is estimated using a global distance function $D(\mathbf{X},\mathbf{Y})$. The number of reference (prototype) vectors $\mathbf{R}$ that should be included in calculation of $p(C_i|\mathbf{X};M)$ may be fixed (as in the k-nearest neighbor method), or a specific weight function may be imposed around $\mathbf{X}$, for example a Gaussian function.

Heterogeneous SBM models use local functions optimized differently in different regions of space. A set of local similarity functions $D_R(\mathbf{X},\mathbf{R})$ should be defined, associated for example with the reference vectors. For example, using Minkovski's L_α distance with the scaling factors defined as:

$$D_R(\mathbf{X},\mathbf{R};s)^\beta = \sum_{j}^{N} s_R^j |X_j - R_j|^\alpha; \quad s_R^j \geq 0 \qquad (2)$$

and optimizing the scaling factors s_R^j, allows each prototype $\mathbf{R}$ to "pay attention" to different features of the input around different prototypes. In the most common version of the similarity-based methods, such as the k-nearest-neighbor method or LVQ method, decision borders are piecewise linear. Here generalized Minkowski's metric involves two parameters, exponents α and β, and the scaling factors s^j. This enables feature selection and provides very flexible shapes of decision borders. If all contributions $s_i|A_i - B_i|$ for some input feature i are small the feature may be eliminated.

Although all functions in RBF networks are usually of the same type, they are attached to local centers. Optimization of individual dispersions corresponds to the scaling factors s_R^j in the distance functions. In case of RBF networks with Gaussian functions similarity functions are related to the squared Euclidean distance functions $(\mathbf{X}-\mathbf{R})^T\Sigma^{-1}(\mathbf{X}-\mathbf{R})$ via exponential transformation $\exp(-D_R(\mathbf{X},\mathbf{R}))$, while probabilities $p(C_i|\mathbf{X};M)$ are calculated taking linear combination of similarities. The heterogeneous SBM models are more general, allowing different local similarity functions to be attached to different nodes. Various methods may also be used to combine contributions from each node to calculations of probabilities. This is a generalization of the class-dependent distance functions sometimes used in the k-NN method [16]. Except for typical distance functions and correlation based functions data-dependent distances may be used, such as the Modified Value Difference Metric (MVDM) [17].

Adapting the similarity function to minimize in-class distance variance and maximize between-class variance, a nonlinear version of Fishers discrimination analysis is obtained. Optimal parameterization $D_X(\mathbf{X},\mathbf{R})$ may be learned for training an MLP neural network to provide small $D_X(\mathbf{X},\mathbf{R})$ output values for input vectors $(\mathbf{X},\mathbf{R})$ that are of the same class, and larger values otherwise. Another way is to replace the difference $|\mathbf{X}_i - \mathbf{R}_i|$ by a function $d(\mathbf{X}_i,\mathbf{R}_i)$ evaluating contributions to the distance in 2.

Ideas discussed above provide a rich framework for construction of hetereogenous similarity-based systems. Selection of the prototype vectors, creation of the codebook vectors using Learning Vector Quantization techniqes [18] or instance-based learning algorithms [19] are relatively inexpensive procedures that may be combined with optimization of local distance functions.

V. Input transformations

The hidden layer of a neural network maps the inputs into an image space trying to simplify the task of the perceptron output node, for example by creating linearly separable data clusters. Instead of the hidden layer transfer functions that contain some adaptive parameters one could use arbitrary multivariate functions to transform inputs, trying to achieve similar result. In the *functional link* (FL) networks of Pao [20] combination of various functions, such as polynomial, periodic, sigmoidal and Gaussian functions is used. These networks were never popular and little is known about their properties. The use of products of pairs $x_i x_j$ or of higher order products is not very practical for high-dimensional inputs because the number of such products grows rapidly. Functional link networks use such products and other functions, and than pass the results as inputs to an MLP. These pre-processing functions should be regarded rather as filter functions than transfer functions. Adding standard neurons connected to pre-processed inputs is equivalent to using processing elements that are heterogeneous.

Pre-processing may be done independently of the network by basis functions $\Phi(\mathbf{X})$ (acting on the whole input vector $\mathbf{X}$ or on a few features only) if they do not involve adaptive parameters. The network usually performs a weighted combination of enhanced inputs. However, filter functions that have adaptive parameters should be a part of the network. To avoid excessive number of inputs one could form a candidate input node and evaluate its information content using some feature-evaluation techniques before adding new dimension to the input space.

Except for adding filtered inputs to the existing inputs one may renormalize all input vectors, for example using Minkovsky's metric. Such input renormalization has dramatic influence on network decision borders [21]. Adding a new feature based on sum of squares $\Phi(\mathbf{X}) = \sqrt{||\mathbf{X}||^2_{\max} - \sum_i X_i^2}$ creates circular contours of constant value $\Phi(\mathbf{X})$=const and renormalizes all enhanced vectors to $||(\mathbf{X}, \Phi(\mathbf{X}))|| = ||\mathbf{X}||_{\max}$. If renormalization using Minkovsky's metric is desired then $\Phi(\mathbf{X})^\alpha = ||\mathbf{X}||^\alpha_{\max} - \sum_i |X_i|^\alpha$, where $||\mathbf{X}||$ is now the Minkovsky's norm. Adding one extra input normalized in a different way for each hidden node will change the decision borders realized by this node, making the whole network heterogeneous. So far such systems have not yet been implemented in practice.

VI. Heterogeneous decision trees

Heterogeneous decision trees may be created in several ways. A rather obvious, although computationally expensive way, is to place a whole new classifier, such as a neural network, in new node, creating a neural decision tree [22]. Although decision regions may be of different type in fact neural trees are still composed of a combination of simple perceptrons with half-spaces as decision regions and thus are not able to discover the simplest description of the data. The same concerns Fisher and kernel-based decision trees [23].

The simplest approach to create heterogeneous decision trees is to provide new tests at each node. Decision trees select the best feature and its threshold value, differing in functions that are used to evaluate the amount of information gained by splitting the node [1]. The test for continuous or ordered values are usually of the same form:

$$X_i < \theta_k \text{ or } \sum_i W_i X_i < \theta_k \tag{3}$$

for univariate and multivariate trees, respectively. θ_k is the selected threshold value for the node k. Decision regions are in this case half-spaces, either perpendicular to the axis or of arbitrary orientation. Replacing thresholds by intervals $X_i \in [\theta_k, \theta_k]$ does not change the type of decision borders. Adding a distance-based test with the Minkovsky's L_α norm:

$$||\mathbf{X} - \mathbf{R}||_\alpha = \sum_i \left(|X_i - R_i|^\alpha\right)^{1/\alpha} < \theta_R \tag{4}$$

where $\mathbf{R}$ is the reference vector, provides new type of decision regions. In particular for L_2 (Euclidean distance) spherical decision regions are obtained, while for L_1 (Manhattan distance) decision regions are romboidal, and for L_∞ cuboidal. In the last case decision rules performed by the tree nodes are equivalent to standard crisp logic rules.

A general way to provide a new set of features is to introduce $\Phi(\mathbf{X}; \mathbf{R})$ functions, measuring similarity of $\mathbf{X}$ to some reference objects $\mathbf{R}$ using one of the selected distance functions. A decision tree using such features selects the best one, in effect using quite different decision regions for partitioning the input space. In particular adding L_2 norm with $\mathbf{R} = 1/N(1, 1, \ldots 1)$ allows to discover the simplest tree solving the problem with half-plane and Gaussian distributions. There is a tradeoff between the complexity of the tests one can consider in a finite time and the complexity of the final decision tree.

SSV is a general criterion that can be applied to many different problems. The best split value is the one that separates the largest number of pairs of objects from different classes. The *split value* (or *cut-off point*) is defined differently for tests returning continuous and discrete values. In the case of continuous tests the *split value* is a real number, in other cases it is a subset of the set of alternative values of the feature. In all cases we can define *left side* (*LS*) and *right side* (*RS*) of a split value s of feature f for given dataset D:

$$\begin{aligned} \mathrm{LS}(s, T, D) &= \begin{cases} \{\mathbf{X} \in D : T(\mathbf{X}) < s\} & \text{if } T(\mathbf{X}) \text{ is cont.} \\ \{\mathbf{X} \in D : T(\mathbf{X}) \notin s\} & \text{otherwise} \end{cases} \\ \mathrm{RS}(s, T, D) &= D - \mathrm{LS}(s, T, D) \end{aligned} \tag{5}$$

where $T(\mathbf{X})$ is the test applied to the data vector $\mathbf{X}$; in particular the test may select a single feature value that is compared with the threshold s. The *separability of a split value s* is defined as:

$$\begin{aligned} \mathrm{SSV}(s) = 2 * \sum_{c \in C} &|LS(s, T, D) \cap D_c| * |RS(s, T, D) \cap (D - D_c)| \\ - \sum_{c \in C} &\min(|LS(s, T, D) \cap D_c|, |RS(s, T, D) \cap D_c|) \end{aligned} \tag{6}$$

where C is the set of classes and D_c is the set of data vectors from D which belong to class c.

The best split value separates the maximal number of pairs of vectors from different classes, and among all the split values which satisfy this condition, the best one separates the smallest number of pairs of vectors belonging to the same class. For every dataset containing vectors which belong to at least two different classes, for each feature which has at least two different values, there exists a split value of maximal separability.

Some test may include a linear combination of inputs, trying to determine best separating hyperplane; in this case Linear Discriminant Analysis (LDA) or Fisher Discriminant Analysis (FDA) methods [1] may be used to find best combinations, leading to LDA and FDA trees. Simple tests may also be based on distances $||\mathbf{X}-\mathbf{R}|| < \theta_R$ from the data vectors $\mathbf{X}$ to reference points $\mathbf{R}$ in the feature space, providing non-linear decision borders, depending on the type of the distance function. The heterogeneous SSV (h-SSV) algorithm has 3 main steps:

- calculate the value of the test $T(\mathbf{X})$ for all the data vectors in the dataset;
- sort the data vectors according to the values of test function;
- analyze each candidate split value and choose the one with the highest SSV value.

The computational complexity of this algorithm is equal to the complexity of sorting (i.e. $n\log(n)$, because new feature calculation and analysis of candidate splits require only nk and n operations, if the new feature is the distance to a reference point (n is the number of data vectors in the dataset and k is the number of features describing the data). In the case of distance-based tests there are some possibilities of "natural" restrictions on selections of reference vectors for the best test. The training data vectors are good candidates for reference vectors $\mathbf{R}$ that will be used in calculation of the test $T_R(\mathbf{X}) = D(\mathbf{X},\mathbf{R})$. The value of the test $T_R(\mathbf{X})$ may be regarded as a new feature characterizing vector $\mathbf{X}$. To speed up evaluation of such tests one may pre-select a small number of reference vectors covering large regions of the feature space (i.e. vectors relatively far from each other). After finding the best candidate mode detailed search may be done in its neighborhood.

Another simplification is to search for candidates among the training data vectors that belong to the current tree node that is being expanded. This leads to more comprehensive split decisions, selecting a neighborhood of the reference vector as one of the subsets, and the rest of the space as the other subset. The reference vector is than treated as a prototype, giving an understandable logical rule based on similarity to this prototype. The best reference vector does not need to be one of the data vectors. A better reference can be found through a minimization process. To avoid higher computational costs only selection has been used in initial implementation of the h-SSV algorithm.

Because different distance measures give very different decision borders such enhancement in decision tree methods leads to heterogeneous systems that have the chance to discover simple class structures. To avoid complex tests and minimize overall complexity of the decision tree model some penalty of using complex tests may be added to the SSV criterion. The simplest tests are based on cutoff for single features; linear combinations should be used only if the gain in accuracy justifies additional parameters, and novel distance functions are even more complex, requiring determination of the reference vector and the parameters of the distance function. The best model selection approach for the heterogeneous trees is an open question. Below a few preliminary results are presented to show the potential of this approach.

VII. Some results

Artificial data that are difficult for decision trees and other computational intelligence systems have been used first. The plane shown in Fig. 1 has been rotated by 45 degrees and no tests using linear combinations were allowed. 2000 data points have been generated. As a result the SSV decision tree gave:

- 44 rules, 99.7% accuracy without distance-based tests;
- 21 rules, 99.8% accuracy, with distance-based tests selecting only reference vectors from the node vectors;
- 15 rules, 99.85% accuracy with distance-based tests selecting from all vectors.

For the **Iris** dataset [13] searching for distance-based tests has been done with Euclidean distance only, and all training data vectors as candidates for reference vectors. A set of rules with 96.7% accuracy (5 errors overall) has been created:

- if petal length < 2.45 then class 1
- if petal length > 2.45 and $||\mathbf{X}-\mathbf{R}_{15}|| < 4.02$ then class 2
- if petal length > 2.45 and $||\mathbf{X}-\mathbf{R}_{15}|| > 4.02$ then class 3

Here $||\mathbf{X}-\mathbf{R}_{15}||$ is the Euclidean distance to the vector $\mathbf{R}_{15}$. The restriction to search reference points among the data falling into the tree node gives the same set of rules that can be found without distance based premises (96% accuracy, 6 errors):

- if petal length < 2.45 then class 1
- if petal length > 2.45 and petal width < 1.65 then class 2
- if petal length > 2.45 and petal width > 1.65 then class 3

For the **Wisconsin breast cancer** dataset (699 cases, 9 features, 2 classes, [13]) the following decision rule has been found:

if $||\mathbf{X}-\mathbf{R}_{303}|| > 20.27$ then benign else malignant

This rule makes 18 classification errors (97.4% accuracy). It seems to be the simplest rule discovered for this dataset so far. In a cross validation process multiple sets of rules with very similar or equal accuracy can be found. For instance:

$$\text{if } ||\mathbf{X} - \mathbf{R}_{279}|| > 19.57 \text{ then benign else malignant}$$
$$\text{if } ||\mathbf{X} - \mathbf{R}_{612}|| > 20.10 \text{ then benign else malignant}$$

VIII. Discussion

Heterogeneous Adaptive Systems have several advantages over other types of adaptive systems. First, they may discover simplest structures generating the data. Imagine two sources of signals with different distributions imposed on the background signal, sampled by multidimensional measurements. Homogenous classifiers have no chance to discover the structure of such data while HAS should have no problem if appropriate transfer functions, similarity functions or tests functions are provided. This enables simple interpretation of the data structure. The quality of simple HAS solution should be higher than quality provided by more complex systems. HAS methods provide an interesting alternative to kernel based methods and support vector machines. Neurofuzzy systems with nodes based on different membership function and T or S-norms also belong to this category.

There is an obvious tradeoff between the flexibility of representation by a single element and the number of elements needed to accurately model all data. The goal is to optimize overall complexity of the system. On the other hand systems that are very general, based on higher-order logic knowledge representation formalism, are very inefficient. Several training methods have been described here but finding optimal training method for each heterogeneous system requires further investigation and empirical comparison. Our implementation of HAS decision tree, although still containing only very few options, has already found the simplest rule for the Wisconsin Breast Cancer data.

Finding the best HAS model is related to a more general idea of searching in the model space for the best model appropriate for a given data, recently introduced within the framework of the similarity based methods [24]. Both approaches have similar goals, although different biases. HAS finds heterogeneous models of a given type, such as neural networks, while our version of the search in the model space creates different models within a common similarity-based framework. A meta-learning approach could search in a space of all models, including HAS models, for most appropriate model accounting for the data, but it remains to be seen whether effective searching/learning algorithms for such general approach exist.

Acknowledgments

Support by the Polish Committee for Scientific Research, grant 8T11C 006 19, is gratefully acknowledged.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd Ed, John Wiley & Sons, New York 2001

[2] W. Maass, T. Natschläger and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations” (preprint, Technische Universität Gratz, 2001)

[3] N. Christianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel-Based Learning Methods. Cambridge University Press 2000.

[4] L. Breiman, “Bias-Variance, regularization, instability and stabilization”. In: C. Bishop, ed. Neural Networks and Machine Learning. Springer 1998

[5] W. Duch, R. Adamczak and K. Grąbczewski, “Methodology of extraction, optimization and application of crisp and fuzzy logical rules.” IEEE Transactions on Neural Networks, Vol. 12, pp. 277-306, 2001

[6] W. Duch and N. Jankowski, “Survey of neural transfer functions”. Neural Computing Surveys, Vol. 2, pp. 163-213, 1999

[7] W. Duch and N. Jankowski, “Taxonomy of neural transfer functions”, International Joint Conference on Neural Networks (IJCNN), Vol. 3, pp. 477-484, 2000

[8] N. Jankowski and W. Duch, “Optimal transfer function neural networks.” 9th European Symposium on Artificial Neural Networks (ESANN), Brugge 2001. De-facto publications, pp. 101-106

[9] W. Duch, R. Adamczak and G.H.F. Diercksen, “Constructive density estimation network based on several different separable transfer functions.” 9th European Symposium on Artificial Neural Networks (ESANN), Brugge 2001. De-facto publications, pp. 107-112

[10] W. Duch and N. Jankowski, “Transfer functions: hidden possibilities for better neural networks.” 9th European Symposium on Artificial Neural Networks (ESANN), Brugge 2001. De-facto publications, pp. 81-94

[11] N. Jankowski and V. Kadirkamanathan, “Statistical Control of RBF-like Networks for Classification, 7th Int. Conf. on Artificial Neural Networks (ICANN), Springer-Verlag, pp. 385-390, 1997

[12] W. Duch and GHF Diercksen, “Feature Space Mapping as a universal adaptive system”. Computer Physics Communications, Vol. 87, pp. 341-371, 1995

[13] C.J. Mertz, P.M. Murphy, UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-data-bases.

[14] A Sebald and K. Chellapilla, “On Making Problems Evolutionarily Friendly: Evolving the Most Convenient Representations.” 7th Int. Conf. on Evolutionary Programming, EP98, Mar 25-27, 1998, Mission Valley Marriott, San Diego, CA.

[15] W. Duch, “Similarity-Based Methods”. Control and Cybernetics **4**, pp. 937-968, 2000

[16] B.V. Dasarathy, ed. Nearest neighbor norms: NN Pattern Classification Techniques. Los Alamitos, California: IEEE Computer Society Press 1990

[17] D.L. Waltz, “Memory-based reasoning”. In: M. A. Arbib, ed, The Handbook of Brain Theory and Neural Networks. MIT Press, pp. 568-570, 1995

[18] T. Kohonen, Self-organizing maps. Berlin, Springer-Verlag 1995

[19] D.W. Aha, D. Kibler and M.K. Albert, “Instance-Based Learning Algorithms.” Machine Learning, vol. 6, pp. 37-66, (1991)

[20] Y-H. Pao, Adaptive pattern recognition and neural networks. Addison-Wesley, Readnig, MA 1989

[21] Duch W, Adamczak R, Diercksen GHF. Neural Networks in non-Euclidean spaces. Neural Processing Letters 10 (1999) 201-210

[22] J. A. Sirat and J.-P. Nadal, “Neural Trees: a New Tool for Classification”. Network, Vol. 1, pp. 423-438, 1990

[23] S. Mika, G. Rätsch, J. Weston, B. Schölkopf and K.-R. Müller. “Fisher discriminant analysis with kernels”. In: Y.-H. Hu *et al.* , eds, IEEE Neural Networks for Signal Processing IX, pp. 41–48, 1999.

[24] W. Duch and K. Grudziński, “Meta-learning: searching in the model space”. Proc. of the Int. Conf. on Neural Information Processing (ICONIP), Shanghai 2001, Vol. I, pp. 235-240