Internal representations of multi-layered perceptrons

Włodzisław Duch

School of Computer Engineering, Nanyang Technological University, Singapore
& Dept. of Informatics, Nicholaus Copernicus University, Toruń, Poland
Google: Duch

Abstract. Feedforward neural networks make incomprehensible decisions resulting from mappings learned from training examples defined in high dimensional feature spaces. What kind of internal representations are developed by multi-layered perceptrons (MLPs) and how do they change during training? Scatterograms of the training data transformed to the hidden and the output space reveal the dynamics of learning and variety of internal representations that may be formed. Approximately linearly separable problem and non-linearly separable problem (XOR) have been used for illustration. Observation of the activity of neural nodes leads to a number of interesting conclusions about the quality of MLP mappings and their potential for generalization, suggesting various improvements of the training process.

1 Introduction

Feedforward neural networks have been very successful in real life applications, but there is one serious obstacle to a wider acceptance of this technology. Neural networks are black boxes that may show surprising behavior in novel situations and therefore should not be used for safety-critical applications. Multidimensional mappings learned by the hidden layers and mappings from inputs to outputs provided by neural networks are incomprehensible, possibly containing singularities and various kinks.

An alternative to the opaque character of neural networks, advocated by the machine-learning community, is to use logical rules [8]. A rule-based solutions are transparent, expose potential problems and they are easy to explain and understand. Neural network decisions may be converted to logical rules [10, 7]. Unfortunately all attempts to use logical rules always have their price. They imply partitioning of the feature space into hypercuboids (for crisp logical rules) or ellipsoids (for typical triangular or Gaussian fuzzy membership functions), each corresponding to a rule. Such decision borders may be too simple to solve the problem accurately with small number of rules. Moreover, frequently the number of logical rules needed for accurate description is large and they are neither comprehensible nor reliable. Each data analysis method introduces an inductive bias, and rule based methods work well only for data with inherent logical structure that are rare in pattern recognition.

Visualization is the simplest way to understand neural mappings [5, 6, 4] without distorting them by conversion to logical rules, and without loosing important information about the types of errors that the network makes. Error functions optimized by neural networks force the mappings to take a few discrete values. Frequently point-like images in the output space are generated, with a single output equal 1 and all others 0. Even for separable data clusters such mapping may lose important information about data clusters (homogeneity of classes, typicality) and their relations (for example, are B hand-written characters easier to confuse with 8 characters than with E). For non-separable data clusters distinct output values of a network correspond to partitioning of the input space into regions where vectors from a single class dominate. Not only important information is lost in this way, but neural mapping may be quite hard to learn, be unstable against small perturbations of the training set, show poor generalization, and its values (network decisions) may change rapidly for very similar inputs [4]. Regularization [2] and early stopping [11, 9] partially cure these problems, avoiding the point-like final mappings, but an optimal image of the feature space remains to be defined.

Several conclusions concerning the training process may be drawn from careful observation of the network node activity during the learning dynamics. This article is focused on better understanding of the learning process, creation of internal representations, stability and generalization potential of neural mappings. To simplify visualizations only two-diemensional problems are considered, allowing for faithful creation of images of the activity of the hidden and output neurons, and illustrating the creation of internal representation during the learning process. In the next section approximately linearly separable case is analyzed, while section 3 contain analysis of the fuzzy XOR problem.

2 Approximately linearly separable case

For neural classifiers, committees and several other classification systems, inputs are mapped first to at least one internal space $\mathcal{X} \to \mathcal{H}$, and from there to the output space, $\mathcal{H} \to \mathcal{Y}$. In case of feedforward neural networks each hidden layer k defines such internal space $\mathcal{H}_k$. For committees of classification models the outputs of individual classifiers form an internal representation $\mathcal{H}$ of the input data, with the final output obtained from some kind of voting procedure. Images of the training data and of the query vectors in relation to the training data both in the internal and the output spaces should be visualized.

The training data set $\mathcal{T} = \{\mathbf{X}^{(i)}\}, i = 1 \ldots n$ is defined in N-dimensional input space, $\mathbf{X}_i \in \mathcal{X}, i = 1..N$. In some dimensions feature values $\mathbf{X}_i$ are nominal (in this case binary or several distinct values are assumed), and in some real-valued. The training data is divided into subsets $\mathcal{T}_k, k = 1 \ldots K$, and in most cases each subset of data corresponds to a single class and forms a small number of clusters. Neural networks and other classification systems usually try to map each of the training subsets $\mathcal{T}_k$ to one of the K vertices of the hypercube. If the desired outputs are all zero, except for the one that flags the class of the query vector, for every $\mathbf{X} \in \mathcal{T}_k, M(\mathbf{X}; \mathbf{W}) = \mathbf{Y} = [0, 0, .., 1, 0, ..0]$, these vertices are on the coordinate axes.

The images of the training data set vectors in the output space $\mathbf{Y}(\mathbf{X})$ and in the internal (hidden) space $\mathbf{H}(\mathbf{X})$ carry important information. Neural networks may achieve the same result in many different ways, as will be evident from visualization of the structure of internal representations of the training data subsets $\mathcal{T}_k, k = 1 \ldots K$. Two vectors $\mathbf{X}, \mathbf{X}'$ that are very similar should be mapped to similar vectors $\mathbf{H}, \mathbf{H}'$ in the hidden space and in the output space $\mathbf{Y}, \mathbf{Y}'$ (the converse is obviously not true). The stability of the mapping may be identified by small perturbation of the input vectors [4].

Mappings provided by networks of different type will significantly differ. This is illustrated by comparing the two most popular network types, MLP with sigmoidal functions, and RBF with Gaussian functions, trained on two slightly overlapping Gaussian clusters, each forming its own class. Both networks have two inputs, two hidden and two output nodes, but in case of RBF output nodes are linear and in case of MLPs outputs are sigmoidal. The two network outputs are normalized, but independent (i.e. they do not have to sum to 1), therefore their joint activity (Y_1, Y_2) is a point somewhere in a unit square. The training is done using Scaled Conjugated Gradient (SCG) procedure for the MLP and Maximum Likelihood for RBF [2].

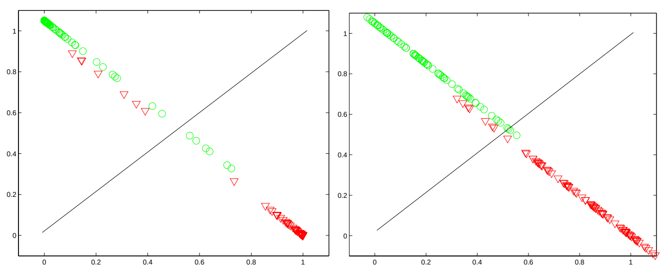

Fig. 1. Left: training data (two Gaussian clusters) mapped to the 2D output space by a typical MLP network; right: same data mapped with RBF network; in both cases two hidden nodes were used and the number of errors is similar, but MLP shows over-confidence, mapping most points close to the two corners of the square.

If the values of two outputs of the mapping are forced to sum to 1 to estimate posterior probabilities $p(C_i|\mathbf{X}; M) = \mathbf{Y}_i(\mathbf{X})$, images of all vectors will fall on a single line $Y_1 + Y_2 = 1$ connecting the $(0, 1)$ and $(1, 0)$ corners. MLP networks with two outputs trained on the $(1, 0)$ and $(0, 1)$ targets develop symmetric solutions $\mathbf{W}^{(1)} = -\mathbf{W}^{(2)}$ for the output perceptrons in the first few iterations; the output perceptrons find the same line (or in general, hyperplane), mirroring each other. This is evident in Fig. 1, showing mappings obtained with standard MLP (left) and RBF network (right) with Gaussian functions and linear outputs. To avoid the overlap of points from different classes a small shift has been applied to the vectors represented by circles. Both output neurons receives the same activations from the hidden layer and because of the symmetry of the data distribution develop mirror hyperplanes. This symmetry will be broken if each output is calculated by two independent networks with a single output each, because the stochastic nature of training algorithm may produce different hyperplanes.

The decision border $Y_1 = Y_2$ shown in Fig. 1 divides the output space into regions where $Y_1 > Y_2$. However, well trained MLP shows errors that are far from decision border, close to the $(0, 1)$ corner; in fact some training methods will make the separating plane almost infinitely steep, creating the network that is overconfident in its decisions. This is a result of rapid growth of the output weights that may reduce error function to zero (in a separable case) or to a minimum even if the input hyperplanes are not optimal. On the other hand RBF networks are less confident as long as dispersions of Gaussians functions are not very small (Fig. 1). Vectors that correspond to images near the decision border are easily identified and may be traced back in the input space, but are they outliers, far from the main data clusters, or do they come from the overlapping region, where data clusters are quite dense? In the first case their images should be closer to the $(0, 0)$ corner, indicating low confidence in classification, or "don't know" answer, in the second case they should be closer to the $(1, 1)$ corner. Classifiers that estimate probability density may provide such information.

More insight may be gained by looking at the image of the training set in the hidden space $\mathcal{H}$. Comparing such images for different mappings will immediately show significant differences despite similar classification accuracy. The position of the image $\mathbf{Y}$ of a new vector $\mathbf{X}$ in relation to the images of training vectors $\mathbf{Y}^{(k)}$ shown in the scatterogram may allow for evaluation of the reliability of its classification. For two-dimensional images all information is contained in scatterograms. In Fig. 2 three types of internal representations are shown for the MLP mapping of two overlapping Gaussian clusters. The first two were obtained without regularization. Although initialization may place perceptron planes (shown as lines in Fig. 2, left column) provided by hidden neurons close to the optimal positions (as in the bottom row), backpropagation training may move them quite far away. Without regularization weights are growing quickly and there is no time for optimal internal representations to form. Unstable representations are created, with two basic types being: 1) a single neuron representation, with one neuron decision surface removed far from the data and thus contributing a constant value to the network activity, leading to decision lines parallel to one of the axis in the hidden space (Fig. 2, middle row); or 2) neurons that cut through the center of clusters, leading to the scattered internal representations (Fig. 2, top row). Due to the symmetry of the problem any reflection and rotation by multiple of 90 degrees of the hidden layer scatterogram is possible.

Adding regularization at the 0.5 level leads to a perfect solution (Fig. 2, bottom row), with the weights of two hidden layer perceptrons becoming essentially identical but of the opposite sign (positive side of the sigmoidal function is indicated by the dashed line), cutting between the clusters in an optimal way. Each hidden layer perceptron solves the problem, mapping the training data on a diagonal line; the output layer is redundant and simply cuts the line into two. In contrast to the representation presented in the middle row both neurons have relatively high variance. In the training procedure regularization term keeps the weights low and the error function contains information about distribution of data in a larger region; perfect internal representation is slowly reached even though the total cost function does not decrease (the regularization and the error term change, but their sum is almost constant). Thus it may be worthwhile to train the network for a longer time, improving internal representation. Weights of the output perceptrons grow too quickly, preventing formation of good internal representations, therefore different regularization parameters for each layer should be beneficial.

The variance of training with regularization is very small, while the variance of standard training is quite high, reflecting the variability of internal representations. In crossvalidation essentially all variance is due to the statistical sampling. The quality

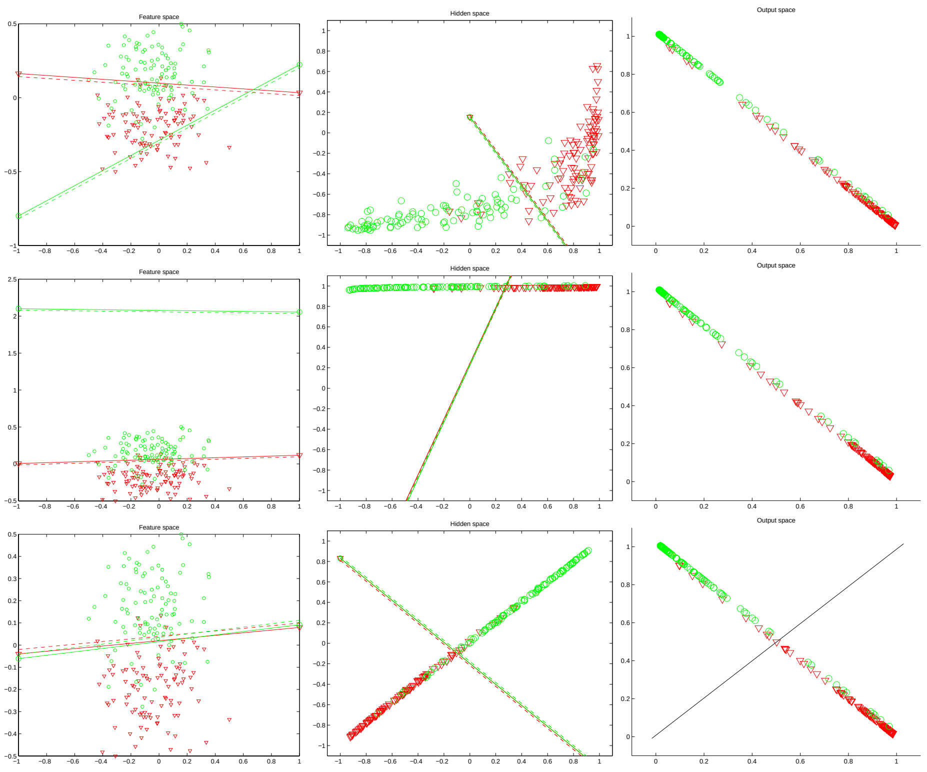

Fig. 2. Overlapping Gaussian clusters in the feature, hidden and output space mapped by an MLP with logistic output neurons. Left column: feature space, showing data and hidden perceptron lines; middle column: hidden space, transformed data and output perceptron lines; right column: output space. First row: most common solution without regularization; second row: only one neuron utilized by the network; third row: perfect solution with regularization coefficient $\alpha = 0.5$.

of the internal representation is seen in the errors that networks make on the test data, especially if probabilities, not the number of errors, are counted. One way to measure that is to sum probabilities $p(C|X)$ for all vectors that have been misclassified and divide it by the sum of all probabilities. The difference for the two Gaussian clusters is as high as 12%. If sharp decision borders are desired (they may reduce the number of errors, especially if the problem has inherent logical structure) regularization coefficient should be reduced near the end of the training to allow for weight growth, as it is done in the MLP2LN algorithm for logical rule extraction [7].

3 Non-linearly separable case

For each of the four XOR points a small Gaussian cluster with 100 vectors is generated. For two well separated clusters both hidden neurons will implement the same hyper-

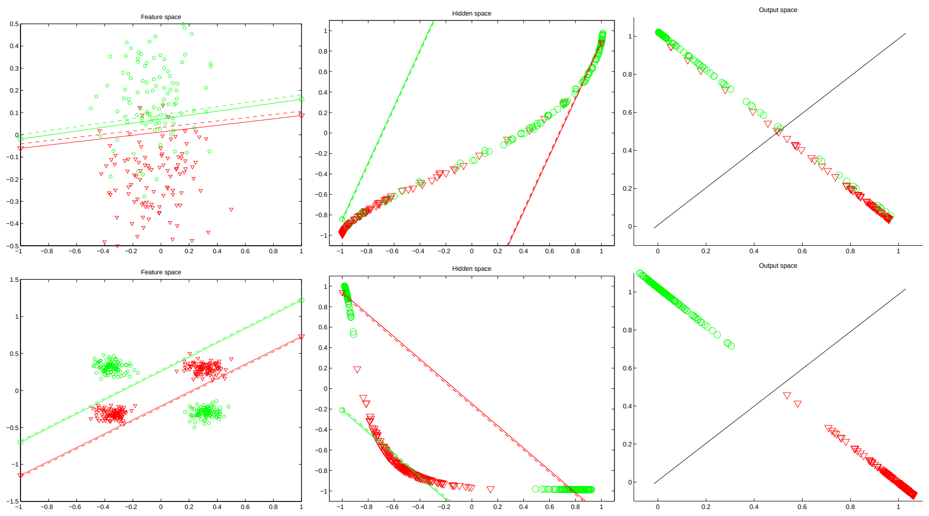

Fig. 3. Overlapping Gaussian clusters (top row) and fuzzy XOR clusters (bottom row) in the feature, hidden and output space mapped by an MLP with linear outputs. Note that interenal representations are quite different than in the previous case and the quality of output separation is not so good.

planes $W^{(1)} = \pm W^{(2)}$; if the orientation of these planes is identical the hidden space image shows the data along $h_1 = h_2$ line, with images of vectors from one cluster around $(0, 0)$ and from the second around $(1, 1)$; otherwise the planes point in two different directions, with data along $h_1 = 1 - h_2$ line, clustering around $(1, 0)$ and $(0, 1)$ corners.

The first row of Fig. 4 shows one of the well converged solutions, with the final values of weights exceeding 10. Perfect division of the circles and triangles is observed, with all 200 vectors from each class mapped to the $(1, 0)$ or $(0, 1)$ corner. The internal representation shows both clusters of circles in the $(0, 0)$ corner because the two sigmoidal functions are zero in the region between the lines (left column) and one outside. SCG training finds this solution most of the time. Although it seems perfect after a small perturbation of the input data many errors will be generated, mapping vectors to the wrong corner, far from decision border in the output space.

The second row shows one of the failed solutions, with one clusters from each class mapped into the same $(0, 1)$ corner by the hidden perceptrons, and the remaining two clusters well separated. In the output space vectors from the two mixed clusters are mapped exactly on the decision border, activating both output neurons in the same way (test accuracy in most cases is 66%). This solution is found only with logistic output functions and no regularization or small regularization coefficients $\alpha < 0.2$.

With regularization coefficient $\alpha = 0.5$ perfect solutions (bottom row, Fig. 4) are generated in all runs. The network is not overconfident, and vectors close to the decision borders are not mapped to the corners but fall on the diagonal closer to the middle of the plot. Small perturbations may lead to small errors but not to a sudden jump to a wrong

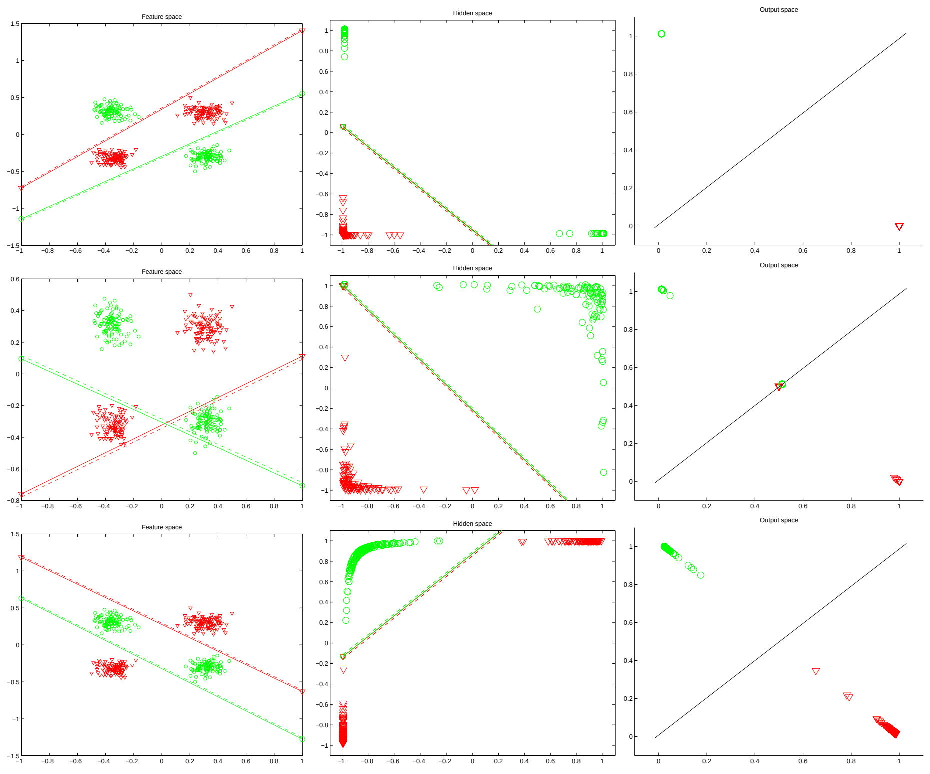

Fig. 4. Activity of two hidden neurons for overlapping XOR clusters. Left column: feature space, showing data and hidden perceptron lines; middle: hidden space, transformed data and output perceptron lines; right: output space. First row: most common solution without regularization; second row: only one neuron used; third row: perfect solution with regularization coefficient $\alpha = 0.5$.

corner, equivalent to high confidence prediction of a wrong class. Thus networks with regularization are rather safe, performing smoother mappings and providing optimal internal representations.

4 Conclusions: some lessons learned from visualizations

Concentration on numbers makes it rather difficult to evaluate many aspects of neural network mappings. It may seem that regularization should solve most problems, clarifying internal representations. Because regularization encourages small weights decision borders become softer and the sum of several sigmoids has rounded, soft corners. Unfortunately problems with inherent crisp logical structure require sharp decision borders and cannot be solved accurately unless the regularization term will encourage both small and large weights and is decreased at the end of the training. Such regularization terms are used in our MLP2LN networks for extraction of logical rules [7] and to some degree with Weigend's regularization term [2].

A surprising observation is that linear outputs, sometimes used in the MLP networks, lead to internal representation with hyperplanes that cut through the data clusters rather than between them, measuring distance to the transformed training vectors (Fig. 3). These internal representations show band-like structure, needed to keep approximate 0 and 1 distance from the transformed data vectors and are quite different from internal representations obtained with the logistic outputs. Although MLPs with linear outputs always converged to the XOR solution while sigmoidal outputs led frequently to wrong solutions the quality of solutions with linear outputs were not so good as with logistic outputs. This clearly shows the weakness of the quadratic error function and suggests its modifications (Duch, in preparation).

Scaled conjugated gradient gives better internal representation than conjugated gradient (CG); this is also reflected in the CG output space images, with single vectors placed very close to the decision border, and feature space images, with perceptron lines cutting through the clusters instead of being optimally placed between them. Mapping training data to vertices on the coordinate axes of the output hypercube is too restrictive. All that should be required in classification tasks is that the mapping of $\mathcal{T}_k$ subsets preserves information about distribution of data in the input space. The definition of appropriate targets that would be the easiest to learn is an open question.

If $\mathcal{T}_k$ subsets do not form a homogenous cluster several alternative targets could be defined for each input vector. The Error-Correcting Output Codes approach [3] to multiclass problems is a step in similar direction. Instead of K targets identified with vertices on the coordinate axes, L-dimensional cube is defined (where L may be smaller or larger than K), with targets coded as binary strings (vertices of this cube), in such a way that the Hamming distance of outputs from different classes is large and individual bits act as hints [1], coding the presence of some high-level features. A single classifier (or L separate classifiers) mapping the query data to binary outputs makes a final decision by selecting the target vertex with the shortest Hamming distance to its output.

The complexity of internal representations grows quickly and some loss of information in the visualization process for more than 3 dimensions is inevitable. Visualizations presented here were done only for two simple problems in two dimensions, but even that enabled interesting observations. Higher-dimensional visualization techniques, such as various linear projections, non-linear mappings and parallel coordinates plots, allow for creation of scatterograms of the hidden and output layer activity on the training data, evaluation of the quality of internal representations, discovering potential problems and predicting the generalization potential of neural mappings. In practical applications, instead of just getting a number (probability or predicted class) neural network outputs show the new case in relation to the known cases, as seen by the trained network. 90% accuracy does not mean that there is always 10% chance of error. If it falls close to the border region it should be carefully inspected and alternative predictions should be taken into account. Visualization should be a part of all neural network programs.

References

1. Y.S. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6:192–198, 1989.
2. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
3. T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal Of Artificial Intelligence Research*, 2:263–286, 1995.
4. W. Duch. Coloring black boxes: visualization of neural network decisions. In *International Joint Conference on Neural Networks, Portland, Oregon*, volume I, pages 1735–1740. IEEE Press, 2003.
5. W. Duch. Visualization of hidden node activity in neural networks: I. visualization methods. In R. Tadeusiewicz L. Rutkowski, J. Siekemann and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 38–43. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.
6. W. Duch. Visualization of hidden node activity in neural networks: Ii. application to rbf networks. In R. Tadeusiewicz L. Rutkowski, J. Siekemann and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 44–49. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.
7. W. Duch, R. Adamczak, and K. Grąbczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.
8. R. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.
9. L. Prechelt. Early stopping – but when? In G.B. Orr and K.-R. Mueller, editors, *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, Berlin, 1998.
10. R. Setiono W. Duch and J. Zurada. Computational intelligence methods for understanding of data. *Proceedings of the IEEE*, 92(5):771–805, 2004.
11. F. Hergert W. Finnof and H.G. Zimmermann. Improving model selection by nonconvergent methods. *Neural Networks*, 6:771–783, 1993.