Learning data structures with inherent complex logic: neurocognitive perspective

WŁODZISŁAW DUCH
Department of Informatics, Nicolaus Copernicus University
Grudziądzka 5, 87-100 Toruń, POLAND
Google: W. Duch

Abstract: Computational systems are still far behind biological systems in object recognition, reasoning or analysis of language structures. What kind of data structures can be learned from data with existing machine learning algorithms? Neurocognitive inspirations show why existing learning systems cannot compete with biological ones. They point the way to more efficient algorithms, generating simplest reliable models of data and capable of object recognition with undetermined number of features. The goal of learning in neural networks and other systems is to transform data into linearly separable data clusters. This is sufficient for relatively simple problems, but makes learning almost impossible if the logic inherent in data is complex. New non-separable targets for learning are introduced to simplify learning and to characterize non-separable problems into classes of growing complexity. Neurobiological and formal justification for new learning targets are given and the case of Boolean functions analyzed.

Key–Words: Neural networks, machine learning, Boolean functions, neurocognitive informatics.
November 7, 2007

1 Introduction

Learning is the key to general intelligence, a chance to bootstrap the existing computational machinery to a new level of human-like competence. Creating artificial systems that could learn in the same way as people may be considered as the most important challenge facing science. Many fields of computer science and engineering are concerned with learning from data: computational intelligence (CI), soft computing, neural networks, pattern recognition, evolutionary computing, and statistics, to name a few [1]. Machine learning has been a subfield of artificial intelligence (AI), initially concerned with learning rules from symbolic data, but now covering all other methods [2]. Despite great progress in development of machine learning algorithms problems that may be solved using existing methods are still relatively simple. In this paper a neurocognitive perspective is adopted, that is practical inspirations will be drawn from current understanding how brains cope with difficult problems that require learning.

The key concept that is used in learning is linear separability [3, 4]. The simplest goal is to learn to distinguish objects $\mathbf{O}$ that belong to some specific category from all other objects $\mathbf{O}'$, given a sufficient number of labeled examples. Objects are usually characterized by features that describe them and may be represent by feature vectors $\mathbf{X} = \mathbf{X}(\mathbf{O})$ or points in some feature spaces, while categories of objects form clusters in these spaces. Objects from category C_k are linearly separable from all others if a hyperplane $\mathbf{W}$ exists such that all points $\mathbf{X}$ representing objects from category C_k are on one side of $\mathbf{W} \cdot \mathbf{X} - \theta$ (where the θ parameter defines position where the hyperplane cuts the $\mathbf{W}$ line), and all points representing objects from different categories $C_i, i \neq k$ are on the other side of this hyperplane. Problems that are linearly separable (that is each class may be linearly separated from all others) are quite easy to solve, and there are numerous methods to do it [3, 4]. Problems that are "logically separable" may be separated by hyperplanes that are perpendicular to the axes and thus discriminate along a single feature value [2] (that is, $\mathbf{W}$ has only one non-zero coefficient), instead of using linear combination of features. Non-separable problems are changed to separable by one of the three types of transformations: 1) topological deformation of probability density distributions; 2) disconnecting data from at least one category into two or more sub-categories that may be separated individually; 3) a combination of the two. For many problems continuous topological transformation of data is sufficient to "flatten" non-linear decision boundaries and make them linearly separable; this is frequently achieved at the expense of compre-

hensibility by projections to high dimensional spaces (as in the kernel learning, popular in Support Vector Machines [5]), or by non-linear transformation using combinations of basis set functions. Some problems that are non-separable can be solved by creating subcategories that may be separated, if the number of cases in each sub-category is sufficiently large to ensure reliability.

This is not the most general approach, as objects may have internal structure that cannot be easily represented by vectors, or may be sufficiently diverse making a description using a common set of features quite difficult. A fruitful approach to such problems is based on similarity measures $S(\mathbf{O}, \mathbf{O}')$ that may be defined in quite general way, for example by operational procedures, or even by subjective judgments. Brains try to minimize energy; in object recognition only those features are evaluated that are needed to discriminate between a set of potential prototypes for different categories. Attention is paid to different aspects of objects, building estimation of (dis)similarity between those aspects of memorized objects that are most helpful for discrimination. This means that brains are actively searching for new features that will reduce neural activation of all those memorized prototypes that represent categories of objects of a different type. Selection of features and object recognition are thus dynamically coupled, in contrast to pattern recognition systems. Similarities are sufficient to categorize objects without the need for numerical representation in feature spaces [6]-[8].

Learning methods search for one particular view on data, and as a result current methods are restricted to problems that are either linearly separable directly in the input space, or in some topologically transformed space, or are relatively easy non-separable problems (like the XOR problem [3, 4]) where creating a few sub-categories is sufficient, or are solved using special architectures and methods designed especially for a given type of problems. For example, many specialized algorithms for learning parity problem, a difficult but rather special Boolean function (given a string of n bits determine if the number of bits equal to 1 is even or odd), have been created in recent years [9]–[18]. These methods are suitable only for the parity problem and will not work for other problems where complex logical functions are responsible for decisions. Dealing with difficult learning problems similar to parity all off-the-shelf algorithms (for example those collected in popular Weka [19] or Ghostminer [20] data mining packages) in the leave-one-out or crossvalidation tests for more than 3-bit parity problems give results at the baserate (50%) level. Knowing beforehand that the data represents parity problem allows for setting appropriate transformations (for example, a multi-layer perceptron, or MLP architecture [3, 4]) to solve it, but already for a modest n it will be impossible to guess how to choose appropriate transformations. In fact a vanishingly small percentage of Boolean functions may be learned by current systems! Learning Boolean functions similar to parity may indeed be a great test for methods that try to evolve neural architecture to solve a given problem, but so far no such evolving systems are in sight.

Biological neural networks are able to solve quite complex learning problems inherent in optimization of behavior, for example understanding of linguistic patterns. A sentence with the same meaning may be constructed in many different ways, but recognizing equivalent meaning in different sentences is a non-trivial task (see Recognizing Textual Entailment Challenge [21]. Therefore finding general algorithms capable of solving problems of similar complexity as the parity problem is an important challenge and is needed to open the doors for a new generation of ambitious machine learning applications. However, existing machine learning methods are not capable of finding good solutions for problems with inherent complex logic. Currently learning problems are divided into quite easy linearly separable problems, and more difficult linearly non-separable types, without any further analysis of how hard the learning of non-separable problems may be. It would be very useful to break the notion of non-linearly separable problems into well defined classes of problems with increasing difficulty. The two goals – characterization of the complexity of non-separable problems, and learning of complex functions – may be reached through introduction of new non-separable targets for learning. Success of learning depends on the type of transformations that create internal representations or image of data. Neural networks that have clear neurobiological motivation create sparse, simple representation in their hidden layers [22]. Popular MLP neural networks are much simpler, they do not use internal inhibition and their only bias towards simple solutions is based on regularization [23], smoothing the mapping implemented by the network. This is not an appropriate bias for problems with complex logical structure, therefore poor generalization should be expected. Analysis of other useful biases and realistic learning targets has never been attempted.

In the next section neurocognitive inspirations that help to solve the problem of learning difficult functions are outlined. The third section describes formal motivations, introducing learning targets and training algorithms. It is important to distinguish between capabilities of models that can provide simplest description of data and the training procedures. The

fourth section is focused on learning of Boolean functions, while the last section contains conclusions.

2 Biological motivations

Multi-layer perceptron (MLP) is the most successful neural network model. It is based on a perceptron model, or a neuron that performs soft threshold logic operation using weighted sum of input signals [24]. This is a rough but useful abstraction of activity of a single biological neuron. Logical threshold neurons, for various noisy input signal distributions concentrated around some average values, estimate conditional probabilities that change in a sigmoidal way, depending on the strength of the signal [25]. Perceptrons may thus be seen as logical devices operating on noisy data.

Hebbian learning leads to weight sharing, constraining parameters of neurons within layers. For example, when the network performs the same feature detection in different parts of an image shared weights are very useful [26]. Mutual inhibition of neuron activity within each layer is very important for achieving sparse internal representations, creating deep networks and solving complex problems. Inhibition may be introduced either by adding inhibitory neurons or by a simple k-winners-take-all (kWTA) mechanism [22]. What happens if inhibition is added to a layer of perceptrons trained using Hebbian principles? Two perceptrons working on the same problem but inhibiting each other share the same input signals x_i, and the same training targets y_j. Hebbian learning updates their weights by $\Delta w_{ij} \sim x_i y_j$, making these weights roughly identical. In effect inputs $\mathbf{X}$ will be projected on the same line $y = \mathbf{W} \cdot \mathbf{X}$, with the two perceptrons implementing $\sigma(y - b_1), \sigma(y - b_2)$ functions. Inhibition should switch off one perceptron when the other is active, leading to $(1, 0)$ or $(0, 1)$ output patterns, but changing biases alone is not sufficient to achieve it. For linearly separable data $\sigma(y - b_1), \sigma(-y + b_2)$ may give the desired outputs, but such solution cannot be obtained using Hebbian learning, requiring weights of opposite signs $\mathbf{W}, -\mathbf{W}$ for the two perceptrons. Incidentally such two perceptrons also solve the simplest non-separable problem (XOR) if their outputs are added and $W = [1, 1]$ projection is used.

Perhaps single perceptrons are not the best abstraction for processing units of networks that learn using biological principles. Not all neurons project their activity to the next layer. They rather form many strongly connected internal microcircuits found in cortical columns, resonating with different frequencies when an incoming signal $X(t)$ appears. This essentially projects the spatio-temporal signals into high-dimensional spaces [27]. Neurons in the next layer observing the activity of a column containing many microcircuits learns to react to signals in an interval around particular frequency in a supervised way based on Hebbian principles. It is sufficient to combine outputs from selected microcircuits correlated with the category that is being learned. In case of signals microcircuits may be treated as resonators specializing in discovering interesting signal structures, such as Gabor filters in vision.

A parallel array of one-bit threshold quantizers with sums of inputs is a crude approximation to such model. It achieves not only optimal signal detection, but even for suprathreshold input signals it improves its performance when additional noise is added, a phenomenon called "suprathreshold stochastic resonance" [28]. In case of abstract reasoning combination of disjoint projections on the $\mathbf{W} \cdot \mathbf{X}$ line is more useful than simple quantizers. Shared weights in network layers that analyze large images [26] may ignore inhibition at longer distances, but locally only one feature is present, therefore adjacent nodes should compete with each other.

What is the simplest abstraction for processing elements of networks that share weights and compete with each other? A single perceptron defining the direction of $y = \mathbf{W} \cdot \mathbf{X}$ projection, followed by a combination of pairs of neurons that capture the activity concentrated around particular values of y, in windows defined by $\sigma(y - b_1) - \sigma(-y + b_2), b_2 > b_1$ (see Fig. 1). Such elements restrict the non-local activity of perceptrons from half-spaces to smaller, but still unbound, areas of the input space. Partially localized neurons avoid pitfalls of localized networks [29]. As shown in [30] some problems require at least $O(n^2)$ parameters using networks with localized functions and only $O(n)$ parameters when non-local functions are used, and vice versa. Partially localized neurons provide elements to build simplest models for data with complex logic. A combination of two neurons serves as an elementary microcircuit that "resonates" only when a specific view of the data ($\mathbf{W} \cdot \mathbf{X}$) falls in a specific "window".

Many microcircuits may become active in the cortex when a new input is presented, but competition and local inhibition will finally leave only a small number of the most active circuits. This may correspond to several views on the data, in each case discovering a particular angle and projecting a large cluster of similar (from this particular angle) cases. A simple threshold neuron may then read out the level of activation of specific circuits, estimating to how many large clusters from particular category the new item belongs to. Similar idea is used in the liquid state machines [27] where many random oscillators are pos-

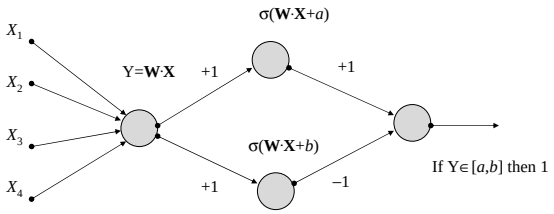

Figure 1: Basic processing element replacing several neurons that share weights and have inhibitory connections.

tulated, projecting the signal into highly dimensional space, and a threshold neuron is used to read out the activity of the column and discriminate between different categories. In this paper only initialization is random but weights are learned and the number of units does not have to be large, in fact simplest but reliable models of data are encouraged. These inspirations are used below to construct practical algorithms.

3 Formal motivations

Adaptive systems, such as feedforward neural networks, SVMs, similarity-based methods and other classifiers, use composition of vector mappings

$$Y(\mathbf{X}) = M^{(m)}(M^{(m-1)}...(M^{(2)}(M^{(1)}(\mathbf{X}))...))$$

to assign a label Y to the vector $\mathbf{X}$. To be completely general direct dependence of mappings on inputs and previous transformations should be considered, for example $M^{(2)}(M^{(1)}(\mathbf{X}), \mathbf{X})$, but for simplicity this will be omitted, considering only strictly layered mappings. These mappings may include standardization, principal component analysis, kernel projections, general basis function expansions or perceptron transformations. $\mathbf{X}^{(i)} = M^{(i)}(\mathbf{X}^{(i-1)})$ is the result of mapping after i transformations steps. For dichotomic problems considered below $Y = \mathbf{X}^{(m)} = \pm 1$.

If the last transformation $Y = M^{(m)}(\mathbf{X}^{(m-1)})$ is based on a squashed linear transformation, for example on perceptron mapping $Y = \tanh(\sum_i W_i \mathbf{X}_i^{(m-1)})$, then the values of Y are projections of $\mathbf{X}$ on the $[-1, +1]$ interval, and a perfect separation of classes means that for some threshold Y_0 all vectors from the Y_+ class are mapped to one side and from the Y_- class to the other side of the interval. This means that the hyperplane $\mathbf{W}$ defined in the $\mathbf{X}^{(m-1)}$ space divides samples from the two classes, and $\mathbf{W} \cdot \mathbf{X}^{(m-1)}$ is simply a projection on the line $\mathbf{W}$ perpendicular to this hyperplane, squashed to the $[-1, +1]$ interval by the hyperbolic tangent or similar function.

The role of the final transformation in the classification process is to compress information, find interesting views on the data from the point of view of certain goals, drastically reducing dimensionality. The learning targets used in most CI methods for classification are aimed at linear separability. The final linear transformation provides a hyperplane that divides the data, transformed by a series of mappings $\mathcal{T}_k(..\mathcal{T}_2(\mathcal{T}_1(\mathbf{X})...))$, into two halfspaces. Linear transformation is the simplest and quite natural if previous transformations increased the input dimensionality and flattened decision borders, making the data linearly separable. However, for difficult problems, such as learning of Boolean functions, this will not work.

Activity of hidden layer neurons visualized for all training data shows [31, 32] that MLP and RBF networks clusterize input data correctly even in complex cases when the network achieves only baserate accuracy and is not able to solve the problem. The failure may be traced to the use of simple perceptrons for the output. This observation led to a definition of the simplest target for good internal representations [33], which is projection on segments of a line. Looking at the image of the training data in the space defined by the activity of the hidden layer neurons [31, 32] one may notice that a perfect solution is frequently found in the hidden space – all data falls into separate clusters – but the clusters are non-separable, therefore the perceptron output layer is unable to provide useful results.

In two dimensions parity is known as the XOR problem, the prototypic non-separable problem. Most MLP training algorithms have already some difficulties to solve it, requiring multiple starts for convergence. The most widely used RBF network with Gaussian hidden units cannot solve it unless special tricks are used. This is an example of a situation in which solutions based on local functions require large number of nodes and examples to learn, while non-local solutions may be expressed in a compact way and need only a few examples (this has been already noted in [30], see also [38]). Consider the noisy version of the XOR problem, replacing each input vector with a small Gaussian cloud (Fig. 2). RBF network with two Gaussian nodes with the same standard deviation σ and linear output provides the following two transformations:

$$\mathbf{X} \to \mathbf{X}^{(1)} = (e^{-|\mathbf{X}-\mu_1|/2\sigma}, e^{-|\mathbf{X}-\mu_2|/2\sigma}) \quad (1)$$

$$\mathbf{X}^{(1)} \to Y = \mathbf{X}^{(2)} = \mathbf{W} \cdot \mathbf{X}^{(1)} \quad (2)$$

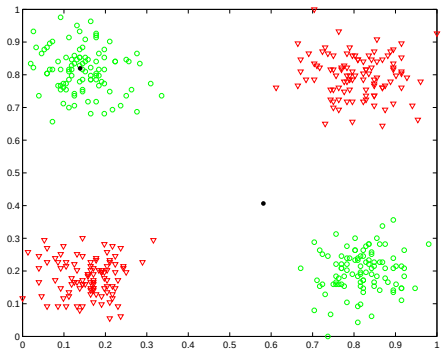

Figure 2: Noisy XOR problem; original data and centers (big dots) of two Gaussian functions after training using maximum likelihood principle is shown.

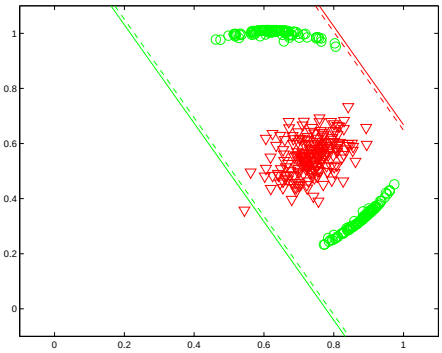

Figure 3: Noisy XOR problem mapped to the hidden space of neural activities shows perfect clusterization; lines represent linear weights of the two network output units.

The solution obtained using maximum likelihood approach [4] places one basis function in the middle of left-corner cluster, and the other close to the center, as shown in Fig. 2. If the target $Y = \mathbf{X}^{(2)} = \pm 1$ is used the linear network output provides a hyperplane (a line in two dimensions) that tries to stay at a distance one from all data points. If a separate linear output for each class is used lines representing both outputs are parallel, with identical weights but shifted on two units, as shown in Fig. 3. Projecting the image data $\mathbf{X}^{(1)}$ on these lines gives one interval with the data from first class surrounded by two intervals with the data from the second class, separating the data into 3 intervals. Although the standard RBF network with two neurons fails to learn the problem, achieving 50% accuracy (base rate), it is clear that all new data will be properly assigned to one of the 3 clusters formed in the hidden space. In crossvalidation test 100% correct answers are obtained using a rule that checks if the case lies in an interval that covers projection of the large middle cluster, or is outside. Thus the networks "knows, but is not able to tell", because it cannot provide linear separability. This is a common situation showing that linear separability is not the best target for learning.

Instead of thinking about the decision hyperplane it is better to focus on interesting projections or more general transformations of data. For linearly separable data $\mathbf{W} \cdot \mathbf{X}$ projection creates two distinct clusters. For non-separable data an easier target is to obtain several well separated clusters. In the Error Correcting Output Codes (ECOC) approach [35] learning targets that are easier to distinguish are defined, setting a number of binary targets that define a prototype "class signature" vectors. This helps to redefine the target for learning, with the final non-linear transformation comparing distances from actual output to class prototypes. Thus instead of aiming at linear separation using the final transformation based on hyperplane the goal of learning may be redefined by assuming another well-defined final transformation. Mapping into separate intervals "disarms" the remaining non-linearity in data, and greatly simplifies the task of all previous transformations. Periodic or quasi-periodic separability in one dimension is worth considering may help to avoid high number of intervals. Mapping on the chessboard targets, or on localized Voronoi cells defined by prototypes localized on a regular multidimensional grid, may handle directly quite difficult non-linearities. As long as clusters are rather large and separated such mapping will achieve high level of generalization and may create a fairly simple model of the data.

Thus two important ideas come from neurocognitive inspirations. First, many views on the same object should be considered, generating interesting transformations $\mathcal{T}_i(X)$ that involve non-local projections $\mathbf{W} \cdot \mathbf{X}$. Such projections are filtered through localized functions $\mathcal{T}_i(X) = G_i(\mathbf{W}_i \cdot \mathbf{X})$ discovering useful features specific for a given category. There is no need for fixed number of features, as they are dynamically generated until there is sufficient information to make decision. The interplay between local and global analysis has been missing in neural networks and other types of machine learning algorithms. Transformations $\mathcal{T}_i(X)$ indicate to which large one-dimensional clusters the input case belongs to and should be at least partially discriminative (exclude some categories). Even a single large projected cluster is sufficient for categorization if there is no strong competition. The winner-takes-most mechanism of biological networks should be approximated to make final decision based on memberships in projected clusters [22]. Second, changing the goal of learning from linear separability to other forms of separability should make the learning process much easier and should help to discover simplest models of

data. It is perhaps surprising that so far neural networks took only the simplest inspirations from biology. In the following section the case of learning complex Boolean functions is analyzed.

4 Parity problems, Boolean functions and k-separability

Determining parity of a string of bits without counting modulo 2 is quite hard. General parity problems can be solved in many ways. The simplest solution [9] is to look at the sign of the $\prod_{i=1}^{n}(x_i - t_i)$, with $t_i \in (0, 1)$, that is to use a single product neuron without any hidden neurons. This solution is very specific to the parity problem and it cannot be generalized to other Boolean functions. Many such solutions that work only for parity problem have been devised [10]-[18], but the challenge is to provide more general solutions that work also for problems of similar or higher complexity.

For n-dimensional parity problem a single linear unit $\mathbf{W} \cdot \mathbf{X}$ with all weights $W_i = 1$ projects data achieving their separation into $n+1$ groups, with 0, 1, 2 .. n bits equal to 1. The $W_i = 1$ weight vector represents diagonal connecting vertices $[0, 0, \ldots 0]$ and $[1, 1, \ldots 1]$, and $\mathbf{W} \cdot \mathbf{X}$ is the projection on this line. Obviously using a single node with $Y = \cos(\omega \mathbf{W} \cdot \mathbf{X})$ gives for $\omega = \pi$ correct answer to all parity problems, $+1$ for even and -1 for odd number of bits. This is the simplest general solution of the parity problem, using a single node network that gives the network ability to count, and adds modulo 2 counting through the periodic function. This simple solution has been missed by previous authors [9]-[18]. The importance of selection of appropriate transfer functions in neural networks is quite evident here (for taxonomy of transfer functions that may be used in neural networks see [30, 34]). In the context of Boolean functions periodic projection is useful only for parity and its negation obtained by symmetric transformations of the hypercube with vertices labeled according to their parity. Projections of other Boolean functions may not be periodic but certainly will show several clusters of vectors from alternating classes.

Linear separability as the target of learning is not so easy to achieve and in fact is not necessary. The last transformation $Y = M^{(m)}(\mathbf{X}^{(m-1)})$ may be designed in any way that will make learning easier. If a projection separating two clusters on a line $Y = \mathbf{W} \cdot \mathbf{X}$ exist the data is linearly separable. If it does not exist a projection forming 3 or more intervals containing clusters from a single class should be sought. However, a projection that alternates from one class to the other, or has a large number of small clusters will not generalize well.

Definition: dataset $\{\mathbf{X}_i\}$ of vectors that belong to two classes is called k-separable if a direction $\mathbf{W}$ exist such that all points $Y_i = \mathbf{W} \cdot \mathbf{X}_i$ are clustered in k intervals, each containing vectors from a single class.

Linearly separable data is called 2-separable, while the XOR problem belongs to the 3-separable category, with projection on the $\mathbf{W} = (1, 1)$ line from even, odd, and again even class. This is the simplest extension of the separability concept, replacing the final mapping $M^{(m)}(\cdot)$ by logical rule:
IF $(Y \in [Y_0, Y_1)]$ THEN even, ELSE odd,
and thus making the non-linearity rather harmless. More sophisticated mappings from one, two or higher number of dimensions may be devised as long as transformation $M^{(m)}(\cdot)$ is easy to set up, providing easier goals for the learning process. The Error Correcting Output Codes (ECOC) [35] tries also to define easier learning targets, but it is still based on linear separability, setting a number of binary targets that define a prototype "class signature" vectors, and comparing the distance from the actual output to these class prototypes. The change of the learning target advocated here is much more powerful.

A dataset that is k-separable may also be $(k+m)$-separable. The separability index for the data should be taken as the lowest k, and learning algorithms that generate solutions with small number of large and well separated clusters should be preferred. If the data is separable into k clusters with very small number of elements, or if the margin separating the intervals between two such clusters is very small, $(k+1)$-separability that leads to larger minimum size of small clusters and their margins may be preferred. Solving a k-separable problem requires finding the direction $\mathbf{W}$ and then setting appropriate $k-1$ thresholds defining intervals on the projection line.

Conjecture: the complexity of k-separable learning should be much easier then 2-separable learning.

This is rather obvious; transforming the data into k-separability form should be much easier because additional transformations are needed to achieve linear separability, and the number of adaptive parameters may grow significantly. For example, the number of hyperplanes that an MLP network needs for the n-parity problem is of the order of n (see comparison of solutions in [9]), giving altogether $O(n^2)$ parameters, while treating it as a k-separable problem requires only $n + k$ parameters. In general, cases when transformation of decision borders in the original input space $\mathbf{X}$ based on continuous deformations may flatten them linear separability will be sufficient, but if clusters are disconnected, as in the case of learning most Boolean functions, transformations that map

data into k-separable form should be easier, providing simpler model of the data.

An interesting question is how many Boolean functions belong to the k-separable category. For n-variables there are 2^{2^n} possible functions; only the bounds for the number of separable Boolean functions are known: the number is between $2^{n^2-O(n)}$ and 2^{n^2} [39], a vanishing fraction of all functions. Unfortunately such estimations are not yet known for the k-separable case. For linearly separable data projections on $\mathbf{W}$ and $-\mathbf{W}$ generate symmetrical solutions (Y_+, Y_-) and (Y_-, Y_+); in case of k-separability additional symmetries and permutations are possible.

It is instructive to analyze in detail the case of learning Boolean functions with $n = 2$ to $n = 4$ bits with the simplest model based on linear projections. Several interesting questions should be investigated: how many k-separable cases for a given direction $\mathbf{W}$ are obtained; which direction gives the largest separation between projected clusters; how many k-separable cases for each direction $\mathbf{W}$ exist; how many different directions are needed to find all these cases. This has recently been done in [33], therefore only a brief summary is provided below.

The Boolean functions $f(x_1, x_2, \ldots x_n) \in \{-1, +1\}$ are defined on the 2^n vertices of n-dimensional hypercube. Numbering these vertices from 0 to $2^n - 1$, they are easily identified converting decimal numbers to bits, for example vertex 3 corresponds to b-bit string 00..011. There are 2^{2^n} possible Boolean functions, each corresponding to a different distribution of ± 1 values on hypercube vertices. There are always two trivial cases corresponding to functions that are always true and always false, that is 1-separable functions. Each Boolean function may be identified by a number from 0 to $2^n - 1$, or a bit string from $00...0$ to $11...1$, where the value 0 stands for false or -1, and 1 for true or $+1$. For example, function number 9 has 2^n bits 00...1001, and is true only on vertex number 0 and 3. For $n = 5$ there are already $2^3 2 = 4294967296$ functions and the percent of linearly separable functions is close to zero. It is rather difficult to estimate the exact number of k-separable functions already for four bits (Table 1); the number of linearly separable functions is less than 3% while the number of functions that are 5 and more separable is about 60%. This is bad news showing that even for relatively simple problems learning of most logical functions will be quite hard.

Linear projection combined with k-separability already gives quite powerful learning system, but almost all computational intelligence algorithms may implement in some form k-separability as a target for learning. It is recommended to search first for linearly separable solutions, and then to increase k searching for the simplest solution, selecting the best model using crossvalidation or measures taking into account the size and separation between projected clusters. Distribution of $y(\mathbf{X}; \mathbf{W})$ values allows for calculation of $P(y|Y_\pm)$ class distributions and posterior probabilities using Bayesian rules. Estimation of probability distributions in one dimension is easy and may be done using Parzen-windows kernel methods.

Table 1: Total number of Boolean functions that are k-separable, for $n = 4$ only estimations.

n	No. func.	2-sep	3-sep	4-sep	higher
2	16	12	4	0	0
3	256	102	126	26	0
4	65535	1880	~ 6836	~ 19110	~ 38360

The main difficulty in formulating a learning procedure is the fact that targets are not fully specified; instead of a single target for Y_+ class two or more labels Y_{+1}, Y_{+2} may be needed. This may actually be of some advantage, allowing for a better interpretation of the results. It is clearly visible in the case of parity problems: each group differs not only by the parity but also by different number of 1's, providing an additional label. Learning should therefore combine unsupervised and supervised components. Network with a single neuron and $n + 2$ parameters and is able to separate a single class bordered by vectors from other classes. For n-dimensional problem that is 3-separable standard MLP architecture requires at least two hidden neurons connected to an output neuron with $2(n+1)+3$ parameters. In k-separability approach one neuron may solve all problems for higher k, simply adding more intervals. The problem may be solved by a fully neural architecture that provides "windows" for intervals (as in Fig. 1), for n-bit parity problems using only n neurons (one linear perceptron and $n - 1$ neurons with frozen weights but adaptive biases for intervals), while in the standard MLP approach $O(n^2)$ parameters are needed [9]. Several new learning algorithms based on these ideas will be published soon (Grochowski and Duch, in preparation).

5 Discussion and open problems

Neurocognitive informatics draws inspirations from neurobiological processes responsible for learning. So far only a few general inspirations have been used in computational intelligence: threshold neurons that perform parallel distributed processing, organized in networks. Even with our limited understanding of the brain many more inspirations may be drawn and

used in practical learning and object recognition algorithms. Here several such inspirations were identified, and recently some interesting algorithms based on neurolinguistic inspirations were introduced [36].

Neurons in association cortex form strongly connected microcuircuits found in minicolumns, resonating with different frequencies when an incoming signal $X(t)$ appears. A perceptron neuron observing the activity of a minicolumn containing many microcircuits learns to react to signals in an interval around particular frequency. Combination of outputs from selected perceptron neurons is used to discover a category. These outputs may come from resonators of different frequencies, implementing an analogue to the combination of disjoint projections on the $\mathbf{W} \cdot \mathbf{X}$ line. The simplest abstraction of this process is given by a combination of two neurons that creates a particular filter for certain aspects of perception. The brain does not use fixed number of features, as most pattern recognition algorithms do, but starting from a small number of features actively searches for new, most discriminative features that neural filters may provide. Objects are recognized using different features that characterize them. Thus feature selection and construction is not separable from the actual process of categorization and learning. The final goal of learning is to categorize, but the intermediate representation is also important. Finding interesting views on the data, or constructing interesting information filters is the most important thing. Instead of using networks with fixed number of inputs systems that actively sample data, trying to "see it" through their filters, are needed. Once they have enough information to categorize data structures they have done their job. This opens the way to new algorithms that may learn from objects that have diverse structures, or many missing values.

Another idea that follows from neural inspiration is the change in the learning targets. It is much easier to achieve non-linear separability in the hidden layers of neural networks then linear separability. If the structure of non-linear mapping that creates image of data is known it may be then analyzed and understood. The most important part for good generalization in learning systems is to create large clusters, as small clusters are not reliable and will be washed out by neural noise. The simplest concept abstracted from that is k-separability, a powerful concept that is very useful for computational learning theory, breaking the space of non-separable functions into subclasses that may be separated into more than two parts. A radically new approach to learning has been proposed, simplifying the process by changing the goal of learning to easier target and handling the remaining nonlinearities with well defined structure. Even the simplest linear realization of k-separability with interval nonlinearities is quite powerful, allowing for efficient learning of difficult Boolean functions. So far there are no systems that can routinely handle such cases, despite a lot of effort devoted to special Boolean problems, such as the parity problem [9]-[18]. One should not suppose that humans are able to learn all Boolean functions from samples. In fact even simple categorization problems with more than 3 dimensions are rather difficult to learn if complex logic is behind decisions [37]. Thus in this area a superhuman learning abilities should be possible, opening many new applications.

In this paper new solutions (based on linear projections and neural architectures) have been proposed not only to the parity problem, but also to learning all kinds of data with inherent complex logic. We are building a new generation of data mining tools based on these and other principles [**?**]. Many specific algorithms are going to be based on the neurocognitive informatics approach.

Acknowledgements: I am grateful for the support by the Polish Committee for Scientific Research, research grant 2005-2007.

References:

[1] W. Duch, What is Computational Intelligence and where is it going? In: W. Duch and J. Mandziuk, Challenges for Computational Intelligence. *Springer Studies in Computational Intelligence* 63, 2007, pp. 1-13.

[2] W. Duch, R. Adamczak, K. Grąbczewski, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks* 12, 2001, pp. 277–306.

[3] T. Hastie, R.Tibshirani and J. Friedman, *The Elements of Statistical Learning* . Springer Verlag 2001

[4] Duda, R.O., Hart, P.E., Stork, D.: Patter Classification. J. Wiley & Sons, New York (2001)

[5] B. Schölkopf and A.J. Smola, *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.

[6] W. Duch, Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics **29** (2000) 937–968

[7] W. Duch, R. Adamczak, and G.H.F. Diercksen, Classification, association and pattern completion using neural similarity based methods. Applied Math. & Comp. Science **10** (2000) 101–120

[8] E. Pękalska and R.P.W. Duin, The dissimilarity representation for pattern recognition: foundations and applications. New Jersey; London: World Scientific 2005

[9] Iyoda, E., Nobuhara, H., Hirota, K.: A solution for the n-bit parity problem using a single translated multiplicative neuron. Neural Processing Letters **18** (2003) 233–238

[10] Wilamowski, B., Hunter, D.: Solving parity-n problems with feedforward neural network. In Kącki, E., ed.: Proc. of the Int. Joint Conf. on Neural Networks (IJCNN'03). Vol I., Portland, Oregon, IEEE Computer Society Press, Los Alamitos, CA (2003) 2546–2551

[11] Liu, D., Hohil, M., Smith, S.: N-bit parity neural networks: new solutions based on linear programming. Neurocomputing **48** (2002) 477–488

[12] Arslanov, M., Ashigaliev, D., Ismail, E.: N-bit parity ordered neural networks. Neurocomputing **48** (2002) 1053–1056

[13] Torres-Moreno, J., Aguilar, J., Gordon, M.: The minimum number of errors in the n-parity and its solution with an incremental neural network. Neural Processing Letters **16** (2002) 201–210

[14] Lavretsky, E.: On the exact solution of the parity-n problem using ordered neural networks. Neural Networks **13** (2000) 643–649

[15] Setiono, R.: On the solution of the parity problem by a single hidden layer feedforward neural network. Neurocomputing **16** (1997) 225–235

[16] Brown, D.: N-bit parity networks. Neural Networks **6** (1993) 607–608

[17] Minor, J.: Parity with two layer feedforward nets. Neural Networks **6** (1993) 705–707

[18] Stork, D., Allen, J.: How to solve the n-bit parity problem with two hidden units. Neural Networks **5** (1992) 923–926

[19] Witten, I., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2nd Ed, 2005)

[20] N. Jankowski, K. Grąbczewski, W. Duch, A. Naud, and R. Adamczak, Ghostminer data mining software. Technical report (2000-2005) http://www.fqs.pl/ghostminer/.

[21] . B. Magnini and I. Dagan, Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, PASCAL 2006.

[22] R.C. O'Reilly and Y. Munakata, *Computational Explorations in Cognitive Neuroscience*, Cambridge, MA: MIT-Press 2000.

[23] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press 1995.

[24] S. Haykin, *Neural networks: a comprehensive foundations*, New York: MacMillian Publishing, 1994.

[25] W. Duch, *Uncertainty of data, fuzzy membership functions, and multi-layer perceptrons*, EEE Transactions on Neural Networks 16, 2005, pp. 10–23.

[26] Y. LeCun, B. Boser, J.S. Denker, B. Henderson, R. E. Howard, W. Hubbard, and L.D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation* 1, 1989, pp. 541–551.

[27] W. Maass, T. Natschläger and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Computation* 14, 2002, pp. 2531–2560.

[28] D. Rousseau and F. Chapeau-Blondeau, Constructive role of noise in signal detection from parallel arrays of quantizers, *Signal Processing* 85, 2005, pp. 571–580.

[29] Y. Bengio, M. Monperrus and H. Larochelle, Non-Local Estimation of Manifold Structure, *Neural Computation* 18, 2006, pp. 2509–2528.

[30] Duch, W., Jankowski, N.: Survey of neural transfer functions. *Neural Computing Surveys* 2, 1999, pp. 163-213.

[31] Duch, W.: Visualization of hidden node activity in neural networks: I. Visualization methods. II. Application to RBF networks. *Springer Lecture Notes in AI* 3070, 2004, pp. 38–49.

[32] Duch, W.: Coloring black boxes: visualization of neural network decisions. In: Int. Joint Conf.

on Neural Networks, Portland, Oregon. IEEE Press Vol I (2003) 1735–1740

[33] W. Duch, K-separability, *Lecture Notes in Computer Science* 4131, 2006, pp. 188–197.

[34] Duch, W., Jankowski, N.: Taxonomy of neural transfer functions. In: International Joint Conference on Neural Networks. Como, Italy, IEEE Press Vol III (2000) 477–484

[35] T.G. Dietterich and G. Bakiri, Solving Multiclass Learning Problems via Error-Correcting Output Codes, *Journal Of Artificial Intelligence Research* 2, 1995, pp. 263–286.

[36] W. Duch, P. Matykiewicz and J. Pestian, Towards Understanding of Natural Language: Neurocognitive Inspirations. *Lecture Notes in Cognitive Science* 4669, 2007, pp. 953962.

[37] R.M. Nosofsky, M.A. Gluck, T.J. Palmeri, S.C. McKinley and P. Glauthier, Comparing models of rule-based classification learning. *Memory and Cognition* 22, 1994, pp. 352-369.

[38] Bengio, Y., Delalleau, O., Roux, N.L.: The curse of dimensionality for local kernel machines. Technical Report Technical Report 1258, Dṕartement d'informatique et recherche opérationnelle, Université de Montréal (2005)

[39] Zuyev, Y.: Asymptotics of the logarithm of the number of threshold functions of the algebra of logic. Soviet Mathematics Doklady **39** (1989)

[40] K. Grbczewski, N. Jankowski, Versatile and Efficient Meta-Learning Architecture: Knowledge Representation and Management in Computational Intelligence. IEEE Symposium Series on Computational Intelligence (SSCI 2007), Honolulu, HI, IEEE Press, pp. 51-58.