Electronic Notes in Theoretical Computer Science 82 No. 4 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume82.html 12 pages

Neural Network Architecture for Synthesis of the Probabilistic Rule Based Classifiers

Dominik Ślęzak [1,2]

Department of Computer Science University of Regina
Regina, SK, S4S 0A2, Canada
and
Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland

Jakub Wróblewski [1,3]

Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland

Marcin Szczuka [1,4]

Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland

Abstract

We introduce a novel neural network architecture, referred to as the normalizing neural network (NNN), where the propagated signals take the form of finite probability distributions. Appropriately tuned NNN can be applied as the compound voting measure while classifying new cases on the basis of approximate decision reducts extracted from the training data. We provide a general scheme of such a classification process, as well as some theoretical issues concerning the NNN construction. We compare the performance of the appropriately learnt NNNs with the fixed voting measures, for some benchmark data sets.

1 Introduction

Within the rough set theory [8], one assumes that a universe of known objects is the only source of knowledge, which can be applied to construct models of

[1] This work was supported by KBN grant 8T11C02519
[2] Email: slezak@pjwstk.edu.pl
[3] Email: jakubw@mimuw.edu.pl
[4] Email: szczuka@mimuw.edu.pl

reasoning about new cases. Reasoning can be stated, e.g., as a classification problem, concerning prediction of a decision attribute under information provided for conditional attributes. Rough set classification systems are usually based on the notion of a *reduct* [8] – a minimal subset of attributes which is sufficient to discern between objects with different decision values. A set of short reducts (or approximate reducts, as in [15]) can be efficiently calculated [2] and used to generate a set of classifying components (sets of rules).

Having multiple components of decision model (multiple classifiers) poses a challenge when it comes to choosing the right ones. It is not uncommon that the single choice is not satisfactory and we should in fact employ a procedure that makes it possible to use several components at the time. To do that we need a mechanism for final decision making on the basis of sometimes conflicting sub-classifiers. A common approach is to set up a voting measure, which combines the outcomes of all classifiers applicable to each given new object according to a fixed mathematical formula. Further, one can design a kind of multi-agent framework for selecting optimal subset (*ensemble* [5]) of classifiers.

We propose to base not only on the selection of agents, but also on combining classification results (voting) using adaptive neural-based structure. It enables to derive the way of optimal synthesis of classifiers directly from the training data, within the framework of learning artificial neural networks (ANNs). The ANN-based approaches have been already applied to adaptive voting between rough set based decision rules in [16,17,18]. In this paper we apply ANNs at the level of synthesis of the whole rough set based classifiers – the sets of decision rules (generated by reducts) instead of single rules. For this purpose, we need to reformulate the standard ANN architecture to be able to handle with the whole vectors of values labelling decision classes.

In purpose of emphasizing the main difference between the standard ANN and the proposed neural network architecture, we refer to that latter one as to the normalizing neural network (NNN). From the sub-classifier outputs (distributions) NNN composes the final decision probabilistic distribution for the entire classification system. The NNN's learning scheme is inspired by classical feedforward neural networks with the inputs and outputs being approximate decision distributions.

2 Probabilities in data based logic

In the rough set theory [8] the sample of data takes the form of an information system $\mathbb{A} = (U, A)$, where each attribute $a \in A$ is a function $a : U \rightarrow V_a$ into the set of all possible values on a. Given arbitrary $a \in A$ and $v_a \in V_a$, we say that object $u \in U$ *supports descriptor* $a = v_a$ iff $a(u) = v_a$. We denote by $||a = v_a||_{\mathbb{A}} \subseteq U$ the set of all objects, which support $a = v_a$.

One can regard descriptors as *boolean unary predicates* and use them to construct logical formulas as their boolean combinations. It leads to the data based logics denoted further by τ_A. In many applications, additional techniques, such as discretization or grouping of values, are used for the purpose of obtaining more relevant descriptors. The sets $\|\alpha\|_{\mathbb{A}} \subseteq U$ of objects supporting boolean formulas $\alpha \in \tau_A$ are obtained from components $\|a = v_a\|_{\mathbb{A}}$ by using standard semantics of logical operators. The aim of the rough set theory is to *approximate concepts* $X \subseteq U$ by means of subsets $\|\alpha\|_{\mathbb{A}}$ [8,11].

The classification tasks usually concern a distinguished decision to be predicted under information provided over the rest of attributes. For this purpose, we represent data as decision systems $\mathbb{A} = (U, A \cup \{d\})$, $d \notin A$. Let $V_d = \langle v_1, \ldots, v_N \rangle$, $N = |V_d|$. For each $i = 1, \ldots, N$, we define the i-th decision class $X_i \subseteq U$ by $X_i = \{u \in U : d(u) = v_i\}$. One can extend the logic built over the attribute descriptors by analyzing probabilistic properties of the boolean formulas. The probability

$$P_{\mathbb{A}}(\alpha) = \frac{|\|\alpha\|_{\mathbb{A}}|}{|U|} \tag{1}$$

reflects the degree of truth of formula $\alpha \in \tau_A$ within $\mathbb{A}$. The probability

$$P_{\mathbb{A}}(d = v_i/\alpha) = \frac{|X_i \cap \|\alpha\|_{\mathbb{A}}|}{|\|\alpha\|_{\mathbb{A}}|} \tag{2}$$

reflects the chance that a randomly chosen object supporting α will drop into the i-th decision class. Quantities (1) and (2) can be regarded as, respectively, the strength and precision of the decision rule $\alpha \Rightarrow d = v_i$.

For any $B \subseteq A$ and $u \in U$, we consider the B-information vector $B(u) = \langle b_1(u), \ldots, b_{|B|}(u) \rangle$, where coordinates correspond to the values of attributes belonging to B. The set of all vectors of values on B, which occur in $\mathbb{A}$, takes the form of $V_B^U = \{B(u) : u \in U\}$. We can rewrite each $w_B \in V_B^U$ as $w_B = \langle v_1, \ldots, v_{|B|} \rangle$ and condition $B = w_B$ as $b_1 = v_1 \wedge \ldots \wedge b_{|B|} = v_{|B|}$. This is a simplified notation for the conjunctions of descriptors in τ_A. In such a case, probabilities (1) and (2) take, respectively, the form of

$$P_{\mathbb{A}}(B = w_B) = \frac{|\{u \in U : B(u) = w_B\}|}{|U|} \tag{3}$$

and

$$P_{\mathbb{A}}(d = v_i/B = w_B) = \frac{|\{u \in X_i : B(u) = w_B\}|}{|\{u \in U : B(u) = w_B\}|} \tag{4}$$

for $\alpha \in \tau_A$ equivalent to $b_1 = v_1 \wedge \ldots \wedge b_{|B|} = v_{|B|}$.

3 Probabilistic and rough membership distributions

Given $\mathbb{A} = (U, A \cup \{d\})$ and $B \subseteq A$, we can span over the set V_B^U the data driven probabilistic distribution, where each vector $w_B \in V_B^U$ is assigned with probability (3). Similarly, each $w_B \in V_B^U$ can be labeled with the conditional probabilistic distribution

$$P_{\mathbb{A}}(d/B = w_B) = \langle P_{\mathbb{A}}(d = v_1/B = w_B), \ldots, P_{\mathbb{A}}(d = v_N/B = w_B) \rangle \tag{5}$$

where particular values $v_i \in V_d$ are assigned with probabilities (4). These probabilities enable to look at subsets $B \subseteq A$ as the generators of the bunches of probabilistic decision rules, capable to be used while the new case classification in various ways. We can talk about a kind of probabilistic multi-rule $B \Rightarrow d$ written as

$$B \Rightarrow d \;\equiv\; \bigvee_{w_B \in V_B^U} \left[B = w_B \Rightarrow P_{\mathbb{A}}(d/B = w_B) \right] \tag{6}$$

Given a new object with the vector of values w_A on A, we can project it onto B and check whether the obtained vector $w_A^{\downarrow B}$ occurs in $\mathbb{A}$, i.e. if $w_A^{\downarrow B} \in V_B^U$. If not, then it means that B is not applicable. If yes, then we can attach to the considered object decision distribution $P_{\mathbb{A}}(d/B = w_A^{\downarrow B})$.

Within the theory of rough sets, probabilities can be also considered at the level of objects, instead of vectors of values. In [9] it was proposed to use the rough membership function, defined for each given $\mathbb{A} = (U, A)$, $B \subseteq A$ and $X \subseteq U$, as the function $\mu_X^B : U \to [0,1]$ with values

$$\mu_X^B(u) = \frac{|[u]_B \cap X|}{|[u]_B|} \tag{7}$$

where $[u]_B = \{u' \in U : B(u) = B(u')\}$ is the B-indiscernibility class of $u \in U$. Given $\mathbb{A} = (U, A \cup \{d\})$, we are interested in approximating decision classes $X_i \subseteq U$, for $i = 1, \ldots, N$. Then, the values of the form $\mu_{X_i}^B(u)$ are equal to (4) for v_i corresponding to X_i and w_B corresponding to $B(u)$. Going further, we can define the rough membership distributions

$$\vec{\mu}_{d/B}(u) = \left\langle \mu_{X_1}^B(u), \ldots, \mu_{X_N}^B(u) \right\rangle \tag{8}$$

which correspond to conditional probabilistic distributions over d, given condition $B = B(u)$, and enable to rewrite (6) as

$$B \Rightarrow d \;\equiv\; \bigvee_{u \in U} \left[B = B(u) \Rightarrow \vec{\mu}_{d/B}(u) \right] \tag{9}$$

Such distributions are regarded as expressing the most detailed information provided by B about d. For description of methods using this information, we refer to further sections and [13,14].

4 Probabilistic decision functions

Vectors $\vec{\mu}_{d/B}(u)$ and $P_{\mathbb{A}}(d/B = w_B)$ provide just an exemplary source of probabilistic decision distributions. One can derive them also by basing on probabilities $P_{\mathbb{A}}(d = v_i/\alpha)$, corresponding to the rules $\alpha \Rightarrow d = v_i$, for $i = 1, \ldots, N$ and *any* $\alpha \in \tau_A$. Further, one can derive decision distributions from the rules $\alpha \Rightarrow \beta$, for *any* formulas $\alpha, \beta \in \tau_{A \cup \{d\}}$, within the unified language $\tau_{A \cup \{d\}}$ based both on decision and conditional attributes. It is enough to localize the decision based descriptors $d = v_i$ within α and/or β and calculate the rule's precision under the foregoing substitutions $i = 1, \ldots, N$. Then, it remains to simply normalize the resulting vector.

In general, *any* classifier may be used to label new objects with the vectors of weights corresponding to particular decision classes. Such vectors can be regarded as object-oriented distributions and further analyzed in purpose of choosing the most accurate decision. In [12] the family of probabilistic functions, helpful in the processing of decision probabilistic distributions, was considered. Given $N \in \mathbb{N}$ decision classes, the set of all probabilistic distributions, which possibly label the considered cases, can be defined as the $(N-1)$-dimensional simplex

$$\triangle_{N-1} = \{s = \langle s[1], \ldots, s[N]\rangle : \min_i s[i] \geq 0 \wedge \textstyle\sum_i s[i] = 1\} \tag{10}$$

For instance, each distribution of the form (8) is an element of $\triangle_{N-1}$. As mentioned before, such distributions seem to express the most accurate knowledge about dependencies of the decision d on the selected attributes $B \subseteq A$. Thus, it should be possible to model various reasoning strategies as functions $\phi : \triangle_{N-1} \to \triangle_{N-1}$ acting over $\overrightarrow{\mu}_{d/B}(u)$ by "forgetting" a part of frequency information, which is redundant with respect to a given approach.

In [12] the following postulates for probabilistic decision functions $\phi : \triangle_{N-1} \to \triangle_{N-1}$ were proposed. They can be referred to as, respectively, the logical and monotonic consistency postulates:

$$\forall_{i,j} \; [\,(\, s[i] = 0 \Rightarrow \phi(s)[i] = 0\,) \wedge (\, s[i] \leq s[j] \Rightarrow \phi(s)[i] \leq \phi(s)[j]\,)\,] \tag{11}$$

Due to the first part of (11), a positive weight cannot be attached to a non-supported event. Further, the relative chances provided by the reasoning strategy cannot contradict those derived directly from an information source.

For instance, let us consider function $\phi_x : \triangle_{N-1} \to \triangle_{N-1}$, where $x > 0$ is a parameter and for each $s \in \triangle_{N-1}$ and $i = 1, \ldots, N$ we have

$$\phi_x(s)[i] = \frac{(s[i])^x}{\sum_{j=1}^{N}(s[j])^x} \tag{12}$$

Function ϕ_x satisfies (11) for any $x > 0$. Vectors $\overrightarrow{\mu}_{d/B}(u)$ and $\phi_x\left(\overrightarrow{\mu}_{d/B}(u)\right)$ differ to each other due to the choice of x. If x is close to 0, then the positive coordinates of $\phi_x\left(\overrightarrow{\mu}_{d/B}(u)\right)$ become more similar to each other, while for large x the differences increase. We discuss the applications of such functions in [15].

5 Probabilistic voting methods

Given decision system $\mathbb{A} = (U, A \cup \{d\})$, we can construct the collections of decision rules $\alpha \Rightarrow d = v_{i(\alpha)}$ for various $\alpha \in \tau_A$ and $v_{i(\alpha)} \in V_d$. Usually, the considered formulas α take the form of conditions $B = w_B$, for possibly frequent vectors $w_B \in V_B^U$ spanned over possibly small subsets $B \subseteq A$. The most valuable rules are these with both the strength and precision high enough.

Once we have the collection of decision rules, there is a big chance that for a given new object a conflict will occur. New objects, with new combinations of the attribute values, may fit the left sides of the rules pointing at different decision classes.To deal with this issue one can, e.g. *vote* between the outcomes of the rules applicable to each particular new object, using the rules' strength and/or precision factors (see [1] for details).

Formulas of the data based logic are usually optimized by means of targeting particular decision classes. However, $\alpha \in \tau_A$ can be also applied to deriving the whole probabilistic decision distributions, as described before, i.e. leading to probabilistic rules of the form $\alpha \Rightarrow P_{\mathbb{A}}(d/\alpha)$. An advantage of such an approach can be seen for highly inconsistent data, where there are no fine approximations of decision classes. For instance, in medical applications, we are often interested in very rare decision classes, not targeted by any reasonably precise decision rules. Then, only the whole probabilistic vectors enable us to analyze the chances for observing such values for new cases.

The easiest way of extracting decision distributions from data is to generate rough membership or probabilistic distributions conditioned by specified subsets of attributes $B \subseteq A$ (cf. Section 3). Let us assume that we have a family of attribute subsets, denoted by $\mathcal{B} \subseteq \mathcal{P}(A)$. Given a new object with the vector of values w_A on A, we can consider the vector of weights $W_{\mathbb{A}}(d/\mathcal{B} = w_A) \in \mathbb{R}^N$ with coordinates calculated, e.g., as follows:

$$W_{\mathbb{A}}(d = v_i/\mathcal{B} = w_A) = \sum_{B \in \mathcal{B}: P_{\mathbb{A}}(w_A^{\downarrow B}) > 0} P_{\mathbb{A}}(d = v_i/B = w_A^{\downarrow B}) \tag{13}$$

Then it remains to choose the decision class with the highest weight. Obviously, one can apply various algorithms for calculating $W_{\mathbb{A}}$. For instance, one can multiply probabilities $P_{\mathbb{A}}(d = v_i/B = w_A^{\downarrow B})$ by the rule strength factors $P_{\mathbb{A}}(w_B)$. Further, condition $P_{\mathbb{A}}(w_A^{\downarrow B}) > 0$ can be changed to $P_{\mathbb{A}}(w_A^{\downarrow B}) > \eta$, for some $\eta \in [0, 1)$. Finally, probabilistic distributions $P_{\mathbb{A}}(d/B = w_A^{\downarrow B})$ can be modified by using appropriately chosen probabilistic functions $\phi : \triangle_{N-1} \to \triangle_{N-1}$. Using the modified probabilities $\phi\left(P_{\mathbb{A}}(d/B = w_A^{\downarrow B})\right)[i]$ instead of $P_{\mathbb{A}}(d = v_i/B = w_A^{\downarrow B})$ in (13) may change a lot in the performance of the classifier.

6 Multi-reduct approaches

In many applications it is convenient to create independent classification algorithms. Even for a single data table, the creation of various classification systems may improve generality of the model. One can try to utilize advantages of different methods by creating a heterogeneous set of classifiers (rule sets, decision trees, k-NN classifiers etc.). One can also base on syntactically comparable models, like in Section 5, where we considered classification models based, for a given decision system $\mathbb{A} = (U, A \cup \{d\})$, on various subsets of attributes, forming the family $\mathcal{B} \subseteq \mathcal{P}(A)$.

A question is how to obtain an appropriate family of attribute subsets from data. As stated before, the most valuable rules are these with sufficiently high strength and precision. Such rules are more likely to be generated by subsets

$B \subseteq A$ with possibly small number of B-indiscernibility classes, corresponding to the elements of V_B^U. Precision of probabilistic rules of the form $B = w_B \Rightarrow P_{\mathbb{A}}(d/B = w_B)$ can be calculated in many ways, by considering various criteria of measuring information stored within distributions $P_{\mathbb{A}}(d/B = w_B)$. Provided with such tools, one is interested in keeping by B the probabilistic decision information at (almost) the same level as the whole A.

Keeping the decision information is concerned with a probabilistic modification of the fundamental rough set theory notion of a decision reduct. We say that $B \subseteq A$ is a μ-decision reduct for $\mathbb{A} = (U, A \cup \{d\})$, iff it satisfies

$$\forall_{u \in U} \; \left[\overrightarrow{\mu}_{d/B}(u) = \overrightarrow{\mu}_{d/A}(u) \right] \tag{14}$$

and none of its proper subsets does. The above condition can be rewritten as:

$$\forall_{w_A \in V_A^U} \; \left[P_{\mathbb{A}}(d/B = w_A^{\downarrow B}) = P_{\mathbb{A}}(d/A = w_A) \right] \tag{15}$$

which states that d is independent from $A \setminus B$ conditioned by B, in terms of probabilities derived from the data.

The problems of searching for optimal μ-decision reducts are NP-hard [14] and usually there exist a number of attribute subsets being (sub)optimal solutions. One can define the considered family $\mathcal{B} \subseteq \mathcal{P}(A)$ as gathering such reduct solutions, found by application of various rough set based heuristics (cf. [2]). Moreover, one can consider shorter (more simple) *approximate* μ-decision reducts [12,13,14], where probabilistic distributions may slightly (up to the choice of the approximation thresholds) vary even on the training data.

The usage of, e.g., formula (13) as synthesizing the μ-decision reduct related distributions may lead to reasonably good classification results. Still, it can be significantly enhanced using adaptive approaches. Examples of the adaptive information synthesis methods are described in [15,20,21]: They may involve the problem of extracting optimal subfamilies of $\mathcal{B}$ and/or the problem of learning optimal voting measures leading to the highest proper classification ratio over the training data.

In that latter case, we can parameterize the voting formulas similar to (13) by the choice of, e.g., probabilistic decision functions or minimal rule strength thresholds, as described at the end of Section 5. In purpose of handling a larger space of measures, we can try to optimize their mathematical structure by means of, e.g., artificial neural networks described in the following sections.

7 Rule based voting with neural networks

Recently, an interest in combining the methods of rough sets and neurocomputing has been growing (cf. [7]). The usage of artificial neural networks for synthesis of previously derived rough set classifiers is just a particular aspect of this tendency. We want to make use of the abilities of multilayer feedforward neural network paradigm. This method is in principle based on minimizing the global error of the neural network by performing weight adjustment determined by the gradient of error functional [4]. Therefore, in order to construct our version of neural network and corresponding learning procedure we propose the network architecture including transition functions for neurons, error measure determining the distance between desired and actual network output, and weight update formula for each neuron.

Our approach to combining neural networks with rough set based methods fits into the landscape of previous studies on such hybrid systems (see [10,16,17]). In particular, there were attempts to use neural network abilities to fine tune rule-based classifiers ([16,18]). In these approaches collections of rules constituted a source of input patterns for neural network. It is assumed that the rule base is somehow optimized before the network construction, in order to eliminate redundancies coming from the fact that the parts of rule base are constructed independently.

With each decision rule $\alpha \Rightarrow d = v_{i(\alpha)}$ there is associated an input neuron a_α. If a given object $u \in U$ supports α (i.e. u is recognized by the α-based decision rule), then it is activated with the real value specified in terms of the rule's strength $P_{\mathbb{A}}(\alpha)$ and precision $P_{\mathbb{A}}(d = v_{i(\alpha)}/\alpha)$. Otherwise, the α-related neuron is not activated (i.e. its output is 0).

The neural network is then constructed and trained to approximate the unknown mapping from the rule-based input neuron values to the original decision. The network itself is supposed to be devised in quite straightforward way. It is a single or multilayer feedforward network with sigmoidal transition functions in neurons, fully connected and sporting one output neuron for each decision value. The final decision in the simplest case is obtained by taking the one corresponding to the most excited neuron in output layer. To train this network simple backpropagation scheme is used ([3]).

The network construction and training methods sketched above address the situation in which we are with rules and want a single decision prediction in the end. Since we are interested in dealing with possibly inconsistent data sets, it would be nice to have similar neural-network-like mechanism capable of expressing not only decision values but the decision distributions (as in Section 5). One may envision this neural network, henceforth called Normalizing Neural Network (NNN), as a compound, multi stage voting mechanisms for solving inconsistencies that may emerge when taking into account several sources of classification distributions (as in Section 6).

8 Reduct based voting with NNNs

In Section 6, we noted that the adaptive approaches to the synthesis of local classifiers are potentially better than those based on the fixed voting formulas. We presented two examples of adaptive methods: searching for the best subfamily of the voting classifiers, and – in Section 7 – learning the voting scheme by means of the neural networks.

The question is whether one can use the neural network based approach for probabilistic decision rules of the form $\alpha \Rightarrow P_{\mathbb{A}}(d/\alpha)$ instead of the standard decision rules $\alpha \Rightarrow d = v_{i(\alpha)}$, as presented so far. Further question is whether we can consider neural networks with inputs understood as collections of rules, e.g. generated by subsets of attributes, or – in the most general case – inputs acquiring decision probabilistic distributions obtained for a new objects from particular classifiers.

In Section 4 we claimed that any classifier may be used to produce the decision distribution for a considered object. If we attach such a classifier to each neuron in the input layer, we obtain the natural scheme of the network combining probabilistic distributions instead of real values.

The difference between the proposed model and the standard artificial network is that we combine the probability vectors and use the probabilistic decision functions $\phi : \triangle_{N-1} \to \triangle_{N-1}$ as activation functions in the neurons.

Let us denote by $d = \langle d[1], \ldots, d[N] \rangle$ – the probabilistic vector, which is the output of the network for a given object and by $t = \langle t[1], \ldots, t[N] \rangle$ – the actual rough membership distribution for given objects. The error function can be expressed by:

$$E = \frac{1}{2} \sum_{i=1}^{N} (d[i] - t[i])^2 \tag{16}$$

The networks we are constructing are simple and always fully connected i.e. the output of a neuron goes to all neurons in next layer. To find weights we use gradient descent method i.e. we update weight vector according to the formula:

$$\overrightarrow{w}_{new} = \overrightarrow{w}_{old} + \eta p(\overrightarrow{w}_{old}) = \overrightarrow{w}_{old} + \eta(-\nabla E(\overrightarrow{w}_{old})) \tag{17}$$

where $\eta \in [0, 1]$ is a *learning ratio* and $p(\overrightarrow{w}_{old})$ denotes the direction in which we change the vector of weights $\overrightarrow{w}_{old}$. In our approach, as in classical gradient descent method, the $p(\overrightarrow{w}_{old})$ is the negative gradient of error (16) treated as function of weight vector.

The key issue in order to assure applicability of proposed neural network is to create an analogon of backpropagation procedure for NNNs. In a nutshell, in the classical neural network there exists effective method for calculating error (gradient) ratios used in weight updates. These values for output layer are easy to derive and for hidden layer they are calculated on the basis of linear combination of error components propagated from the next layer (whence the name backpropagation). Luckily enough it is possible to produce similar procedure in case of NNNs. Unfortunately, the actual proof of this fact exceeds the capacity of this paper as it requires advanced apparatus from the area of multi-dimensional real calculus.

9 Experimental results

In this section we compare the performance of the approximate μ-decision reduct based agents synthesized by using:

(i) The probabilistic voting measures similar to (13) (cf. Section 5);

(ii) The same probabilistic voting measures, applied to the optimized subsets (ensembles) of the initial set of reducts (cf. Section 6 and [15,20]);

(iii) The normalizing neural network with the inputs corresponding to the reducts, learnt over the training data (cf. Section 8).

We examined two benchmark data sets: DNA (60 attributes, 2000+1187 objects, 3 decision classes) and SAT (36 attributes, 4435+2000 objects, 6 decision classes) from [19]. Additionally, we considered the modified DNA (denoted as M-DNA), with only 20 attributes, which are known to provide the largest amount of information (cf. [19]). For DNA we generated 60 approximate μ-decision reducts with various approximation parameter settings. For both M-DNA and SAT, we were basing on 20 reducts.

We examined a number of probabilistic decision functions as the transition functions in the probabilistic network. We decided on $exp_\alpha : \mathbb{R}^N \to \triangle_{N-1}$, for $\alpha > 0$, where for each $s \in \mathbb{R}^N$ and $i = 1, \ldots, N$ we have

$$exp_\alpha(s)[i] = \frac{\exp(\alpha s[i])}{\sum_{j=1}^{N} \exp(\alpha s[j])} \tag{18}$$

This function satisfies only the second part of (11), i.e. the monotonic consistency postulate. Even if $s[i] = 0$, for some $i = 1, \ldots, N$, we obtain $exp_\alpha(s)[i] > 0$. However, for appropriately large α, the result $exp_\alpha(s)[i]$ is very close to 0. Moreover, in real applications, it is often better to add some noise while combining the distributions, which is obtained by labeling completely unprobable decision classes with some small positive weights. Finally, $exp_\alpha : \mathbb{R}^N \to \triangle_{N-1}$ has the derivative matrix analogous to the sigomoidal functions applied in the standard neural network framework, so it enables to look at the performance of the back-propagation algorithm in a similar way.

The results are briefly presented in Table 1

Table 1
Experimental result summary (Train set / Test set)

Data set	Approach (i)	Approach (ii)	Approach (iii)
M-DNA	91.8% / 84.7%	96.3% / 85.1%	91.9% / 89.7%
SAT	83.3% / 53.8%	86.8% / 54.0%	82.6% / 55.0%
DNA	97.2% / 85.8%	100% / 86.9%	96.0% / 93.0%

10 Conclusions

We have briefly presented the approach to data classification with use of data based probability distributions and NNNs advocating that this method provides additional knowledge otherwise inaccessible during classification. By performing initial experiments we verify potential usefulness of this approach in some applications. We sincerely plan to foster this direction of research. In the future, we want to publish the complete description of all steps taken in the paper, especially the full proof of the convergence of NNN's learning algorithm. More experiments are also planned to establish some form of common knowledge about the types of data most suitable for this approach.

References

[1] Bazan, J.: Approximate reasoning methods in synthesis of decision algorithms (In Polish). Ph.D. thesis, Institute of Mathematics, Warsaw University (1998).

[2] Bazan J., Nguyen H.S., Nguyen S.H., Synak P., Wróblewski J.: Rough Set Algorithms in Classification Problem. In: L. Polkowski, S. Tsumoto, T.Y. Lin (eds): Rough Set Methods and Applications. Physica-Verlag, Heidelberg, New York (2000) pp. 49–88.

[3] Hecht-Nielsen, R.: Neurocomputing, Addison-Wesley, New York (1990).

[4] le Cun, Y.: A theoretical framework for backpropagation. In: Neural Networks – concepts and theory, IEEE Computer Society Press, Los Alamitos (1992).

[5] Dietterich, T.: Machine learning research: four current directions. *AI Magazine* **18/4** (1997) pp. 97–136.

[6] Nguyen, H. Son, Szczuka, M., Ślęzak, D.: Neural networks design: Rough set approach to real-valued data. In: Proceedings of PKDD'97, LNAI **1263**, Springer-Verlag, Berlin (1997) pp. 359–366.

[7] Pal, S.K., Polkowski, L., Skowron, A. (eds): Rough-Neuro-Computing. Techniques for computing with words. Springer-Verlag, Berlin, Heidelberg (2003).

[8] Pawlak, Z.: Rough sets – Theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht (1991).

[9] Pawlak, Z., Skowron, A.: Rough membership functions. In: R.R. Yaeger, M. Fedrizzi, and J. Kacprzyk (eds.), Advances in the Dempster Shafer Theory of Evidence, Wiley , Chichester, (1994) pp. 251–271.

[10] Peters, J.F., Szczuka, M., Rough neurocomputing: a survey of basic models of neurocomputation. In: Proc. of RSCTC'02, LNAI **2475**, Springer-Verlag, Berlin (2002) pp. 309–315.

[11] Skowron, A., Pawlak, Z., Komorowski, J., Polkowski, L.: A rough set perspective on data and knowledge. In: W. Kloesgen, J. Żytkow (eds), Handbook of KDD. Oxford University Press (2002) pp. 134–149.

[12] Ślęzak, D.: Normalized decision functions and measures for inconsistent decision tables analysis. To appear in Fundamenta Informaticae (2000).

[13] Ślęzak, D.: Various approaches to reasoning with frequency-based decision reducts: a survey. In: L. Polkowski, S. Tsumoto, T.Y. Lin (eds): Rough Set Methods and Applications. Physica-Verlag, Heidelberg, New York (2000).

[14] Ślęzak D.: Approximate decision reducts (in Polish). Ph.D. thesis, Institute of Mathematics, Warsaw University (2001).

[15] Ślęzak, D., Wróblewski, J.: Application of Normalized Decision Measures to the New Case Classification. In: Proc. of RSCTC'00 (2000).

[16] Szczuka, M.: Symbolic methods and neural networks in classifier construction (in Polish). Ph. D. thesis, Institute of Mathematics, Warsaw University (1999).

[17] Szczuka, M.: Rough sets and artificial neural networks. In: L. Polkowski, A. Skowron (eds), Rough Sets in Knowledge Discovery 2. Physica Verlag, Heidelberg (1998) pp. 449-470.

[18] Szczuka, M.: Refining clssifiers with neural networks. International Journal of Intelligent Systems **16(1)**, Wiley Interscience (2001) pp. 39–56.

[19] UCI Repository of ML databases, University of California, Irvine,(1998) `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[20] Wróblewski J.: Ensembles of classifiers based on approximate reducts. Fundamenta Informaticae **47** (3,4), IOS Press (2001) pp. 351–360.

[21] Wróblewski, J.: Adaptive aspects of combining approximation spaces. In: [7] (2003).