Uncertainty of data, fuzzy membership functions, and multi-layer perceptrons.

Włodzisław Duch

***Abstract*— Probability that a crisp logical rule applied to imprecise input data is true may be computed using fuzzy membership function. All reasonable assumptions about input uncertainty distributions lead to membership functions of sigmoidal shape. Convolution of several inputs with uniform uncertainty leads to bell-shaped Gaussian-like uncertainty functions. Relations between input uncertainties and fuzzy rules are systematically explored and several new types of membership functions discovered. Multi-layered perceptron (MLP) networks are shown to be a particular implementation of hierarchical sets of fuzzy threshold logic rules based on sigmoidal membership functions. They are equivalent to crisp logical networks applied to input data with uncertainty. Leaving fuzziness on the input side makes the networks or the rule systems easier to understand. Practical applications of these ideas are presented for analysis of questionnaire data and gene expression data.**

***Index Terms*— Neural networks, multi-layer perceptrons, extraction of logical rules, fuzzy systems, neural output functions.**

I. Introduction

FUZZY logical rules found numerous applications in classification, approximation and control problems [1], [2], [3], [4], [5], [6]. Many useful algorithms to define and optimize fuzzy membership functions exist. Comprehensibility of these rules unfortunately decreases quickly with the growing size of the rule set, and the sophistication of membership functions and aggregation operators used to draw conclusions. Large sets of fuzzy rules form frequently classification or control systems as opaque as any black box solution based on neural networks.

There is a direct, although rarely explored, relation between uncertainty of input data and fuzziness expressed by membership functions. Various assumptions about the type of input uncertainty distributions change the discontinuous mappings provided by crisp logic systems into more smooth mappings that are implemented in a natural way by fuzzy rules using specific types of membership functions. On the other hand shifting uncertainty from fuzzy rules to the input values may simplify logical rules, making the whole system easier to understand, and allowing for easy control of the degree of fuzziness in the system.

Fuzziness of inputs has frequently natural interpretation and may be modeled directly, while an equivalent modification of the membership function may not be so obvious. For example, in many countries an age limit to see a movie in cinema is based on a crisp decision rule, If (True-age$\geq$ 16) then "Let the person in". In practice true age is not readily available and an Estimated-age, evaluated by visual inspection, is used. Estimated age is a fuzzy number $F(E_{age};\omega)$, a relatively broad bell-shaped function expressing the degree of belief that the age is around the estimated value. The shape of the $F(E_{age};\omega)$ function depends on parameters (ω) that include racial features, individual experience in age evaluation, and the age itself (getting broader for middle values of age). A crisp rule applied to the fuzzy input $F(E_{age};\omega) \geq 16$ is true to a degree described by some membership function $R_{16}(E_{age})$, and therefore this rule may be replaced by a fuzzy rule, If ($R_{16}(E_{age}) \geq$Th) then "Let the person in". The shape of this membership function depends on the parameters defining $F(E_{age};\omega)$ uncertainty distribution function. The threshold Th in the cinema example is shifted towards lower values to let younger customers in.

Although the theory developed below is applicable to any fuzzy system the focus will be on classification rules. Relations between input uncertainty and membership functions may in many important cases be estimated analytically. In particular most assumptions about localized distribution of input uncertainties lead to membership functions with sigmoidal shapes. Such functions are quite common in multilayer perceptron networks (MLPs), with two nodes forming a soft window to filter the data. Putting fuzziness on the input, rather than on the rule side, enables application of fuzzy concepts to any black box system. Sampling from input uncertainty distribution will be equivalent to the use of specific mutidimensional membership functions that may be estimated from such numerical simulations. The effects of increasing input uncertainty (or changing other assumptions about it) may be easier to understand and control than the effects of changing parameters of membership functions on sets of fuzzy rules. For large input uncertainties predictions of class memberships may reach the *a priori* rates, while for crisp input values predictions close to certainty may be possible.

This reasoning allows for an interesting interpretation of MLP networks in terms of fuzzy rules. Equivalence of radial basis function (RBF) networks with fuzzy systems has been well established [5], [7]. Much less work has been devoted to explore relationships between MLP networks and fuzzy systems. Benitez et all [8] showed that for a three-layer MLP network a fuzzy additive system may be constructed that calculates exactly the same mapping. Moraga and Temme [9] show functional equivalence between MLP networks and fuzzy models. In both cases aggregation operators are defined that lead to the replacement of nonlinear neural functions $\sigma(x+y)$ with additive arguments, by aggregation $\sigma(x)\star\sigma(y)$ of independent, single argument functions. Although aggregation operator proposed in [8] is interesting these papers do not

Author is with the Department of Informatics, Nicholas Copernicus University, Grudziądzka 5, 87-100 Toruń, Poland, and School of Computer Engineering, Nanyang Technological University, Singapore 639798; WWW: http://www.phys.uni.torun.pl/ duch

show why sigmoidal functions are so important, or how to find deeper connections of rule-based crisp logic systems with MLP networks. The approach proposed here is not based on a specific aggregation operator, and its practical consequences are quite different.

In the next section relations between the input uncertainties and membership functions are discussed, first for one dimensional problems (single input variable), and then for multidimensional problems. The third section shows some applications of these ideas. Section four presents relations with multi-layer perceptrons. Significance of these results is discussed in the last section of this paper.

II. Input uncertainties and membership functions.

In many applications crisp logic rules are sufficient. Accuracy of crisp rules extracted for a number of real world datasets proved to be higher than of any other classification methods, including fuzzy-rule based systems [10]. Since the number of parameters defining crisp rules is minimal, simple and understandable descriptions of analyzed data are obtained. Therefore a good strategy is to improve crisp rule-based systems without loosing their advantages.

There are several problems with crisp rules [10]. The yes or no answers are not acceptable in many situations, leading to sudden changes for small perturbations of the data samples that lie near decision boundary. Classification systems should provide an estimation of posterior probability $p(C_k|\mathbf{X})$ of assigning vector $\mathbf{X} = \{X_1, X_2, \ldots X_n\}$ to class C_k, or at least membership degrees that change smoothly between adjacent points in the feature space (assuming that some features are continuous). Crisp rules are difficult to optimize, requiring non-gradient procedures to minimize discontinuous cost function (usually the number of classification errors). Continuous values of membership functions should make the optimization process of a set of rules easier.

Introduction of fuzziness is not only desirable, but in most cases it is also unavoidable. Values of continuous inputs taken from tests or observations have finite accuracy. Finding fuzzy system that is equivalent to crisp rule system applied to uncertain feature values allows for controlled introduction of fuzziness.

A. One-dimensional situation.

The simplest situation involves a single input x, and a crisp logic rule premise $x > 0$, or $x \in (0,\infty)$. Suppose that x is measured with accuracy ± 1. Then the uncertainty of x is described by a uniform distribution $U(y-x;1) = 1$ for $y \in [x-1, x+1]$, and zero outside (here x is a parameter, y an independent variable). This is a rectangular membership function $U(y-x;1) = \Theta(y-x-1) - \Theta(y-x+1)$, centered on $y = x$, the average measured (or estimated) input value; $\Theta(\cdot)$ is the step function. If $x > +1$ then the rule is certainly true; if $x < -1$ it is false. Otherwise it may be true to a degree equal to $S_1(x;\Delta x) = S_1(x;1/2) = \max(0, \min(1,(x+1)/2))$. This is a semi-linear membership function, zero for $x < -1$, one for $x > +1$, and $1/2$ for $x = 0$ (see Fig. 1). Various assumptions for rule types, input uncertainty U, and resulting membership functions S, are considered below.

$x > a$ rule, uniform U, semi-linear S.

The use of crisp logical rule with uniform input uncertainty is equivalent to the use of semilinear membership functions for sharply defined input. The truth value of the $x > a$ rule is described by the semilinear membership function $S_1(x - a;\Delta x)$, linear in the $(a-\Delta x, a+\Delta x)$ interval centered at $x = a$ and constant outside of this interval. Using this membership function is equivalent to assumption that x may be anywhere in the interval $(x-\Delta x, x+\Delta x)$, that is it has uniform uncertainty function $U(y-x;\Delta x) = \Theta(y-x-\Delta x) - \Theta(y-x+\Delta x)$. Δx is used here and below to designate the interval around the center of the uncertainty distribution. Symmetric uncertainty functions $U(y-x;\omega)$ for $x > a$ rules lead to antisymmetric membership functions $S(x-a;\omega) - 1/2$ (or without the shift, $S(x-a;\omega) + S(-x+a;\omega) = 1$), and for $x \in (a,b)$ rules to symmetric functions $S(x-(a+b)/2;\omega)$. In general x should be treated as a parameter $U(y;x,\omega)$, but for functions considered here the dependence on x is taken into account by shifting y by x, that is using $U(y-x;\omega)$.

The fuzzy rule $S_1(x-a;\Delta x) > 1/2$ means that the degree of truth of the equivalent crisp rule $x > a$ with the uncertainty function $U(y-x;\Delta x)$ is $1/2$. The fuzzy rule $S_1(x-a;\Delta x) > \theta$ is equivalent to the crisp rule $x > a + (2\theta - 1)\Delta x$ with such uncertainty function.

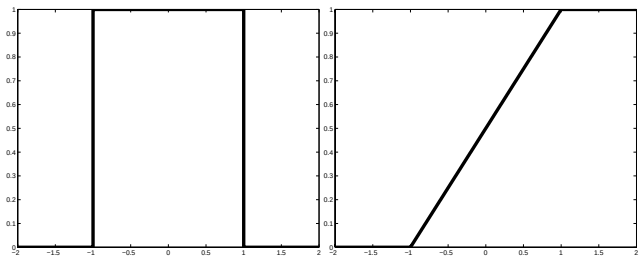

Fig. 1. Uniform input uncertainty for $x = 0$ and the semilinear membership function of the truth degree of the $x > 0$ rule.

$x \in (a,b)$ rule, uniform U, triangular S.

Generalization of $x > a$ rules to rules involving two interval, $R_{a,b}(x)$ true if $x \in (a,b)$ and false otherwise, is straightforward ($b > a$). Using expressions for $x > a$ all formulas for $x < b$ may be deduced from symmetry principles and final formulas obtained by subtracting the two cases.

Consider first the uniform uncertainty function $U(y-x;\Delta x)$ with $\Delta x = (b-a)/2$, matching exactly the rule support. For $x = x_m = (a+b)/2$ in the middle of the interval the degree of fulfillment is 1, but for smaller or larger x it decreases in linear way, reaching zero for $x = x_m \pm 2\Delta x$. Thus the rule $R_{a,b}(x)$ is true to the degree described by a triangular membership function $T_3(x-x_m;2\Delta x) = S_1(x-a;\Delta x) - S_1(x-b;\Delta x)$, centered at x_m, the middle of the (a,b) interval. Triangular membership functions arise in the unlikely situation in which the uniform uncertainty of input values matches exactly the interval defining the rule. In practical application there is usually no reason for such assumption to be true.

$x \in (a,b)$ rule, uniform U, trapezoidal S.

Trapezoidal membership functions are obtained for rule intervals that are either broader ($b-a>2\Delta x$) or more narrow ($b-a<2\Delta x$) than input uncertainty. The main difference between the two cases is that for the narrow rule intervals the degree of fulfillment is never 1, but reaches at most $(b-a)/2\Delta x$, so the trapezoid is not normal. The center of the rule interval $x_m=(a+b)/2$ will be the symmetry point of the trapezoid.

$$\begin{aligned} & T_4(x-x_m;a,b,\Delta x) = \\ = & \int_{-\infty}^{\infty} R_{a,b}(y)U(y-x;\Delta x)dy = \int_a^b U(y-x;\Delta x)dy \\ = & S_1(x-a;\Delta x)-S_1(x-b;\Delta x) \end{aligned} \tag{1}$$

This function is linear for $x\in[a-\Delta x,a+\Delta x]$, constant between $[a+\Delta x,b-\Delta x]$, and linear in $[b-\Delta x,b+\Delta x]$, with zero values outside of these regions. Trapezoidal membership functions result from crisp interval-based rule applied to inputs with uniform uncertainty. Triangular and trapezoidal functions may also be used to model feature uncertainty.

It is important to realize that triangular and trapezoidal functions appear in the dual role here: they may represent input uncertainty distribution (as a function of y, centered on x), or they may serve as membership functions (as a function of x) for fuzzy rules that provide the same results as the crisp rules applied to uncertain inputs. Typical fuzzy system uses this type of membership functions only in this second role, with positions and width parameters fixed as a result of explicit modeling or some optimization procedures. Uncertainty distributions are centered on the value of the input variable x, while membership functions are fixed at positions derived from the logical rule intervals a,b. Uncertainty distributions are of course also membership functions for fuzzy numbers.

The membership functions derived above should be normalized to facilitate standard interpretation. For a crisp rule $R_{a,b}$ (where a or b may be infinite) and any function $U(y;x,\omega)$ representing uncertainty of variable x, normalized membership function representing the degree of fulfillment $\tau(R)$ of the rule is given by the integral:

$$\tau(x;a,b,\omega)=\frac{\int_a^b U(y;x,\omega)dy}{\int_{-\infty}^{\infty}U(y;x,\omega)dy} \tag{2}$$

Fuzzy rule $\tau(x;a,b)>\theta$ is equivalent to a certain confidence in truth of the crisp rule $x>(a+b)/2$. More examples are given below.

$x>a$ **rule, triangular** U**, semi-quadratic** S.

Suppose that repeated measurements of some feature give the mean x with frequency of other values y decreasing symmetrically in a linear way with the distance $|y-x|$ until zero is reached for $y=x\pm\Delta x$. The uncertainty function $U(y-x;\omega)$ has then triangular shape $U(y-x;\omega)=T_3(y-x;\Delta x)$, centered on x and zero ouside the $x\pm\Delta x$ interval. Thus x is not a crisp, but a triangular number, with membership function equal to a difference of two semi-linear functions $T_3(y-x;\Delta x)=S_1(y-(x-\Delta x/2);\Delta x/2)-S_1(y-(x+\Delta x/2);\Delta x/2)$. Crisp rule $x>a$ with triangular uncertainty function $T_3(y-x;\Delta x)$ is true to a degree:

$$S_2(x-a;\Delta x)=\begin{cases}0 & x<a-\Delta x\\ \frac{1}{2}+\frac{(x-a)(2\Delta x+x-a)}{2(\Delta x)^2} & x\in[a-\Delta x,a)\\ \frac{1}{2}+\frac{(x-a)(2\Delta x-x+a)}{2(\Delta x)^2} & x\in[a,a+\Delta x]\\ 1 & x>a+\Delta x\end{cases} \tag{3}$$

The crisp rule with triangular uncertanty is equivalent to a fuzzy rule with $S2$ membership function; by analogy to the semi-linear function $S_1(\cdot)$ this function will be called semi-quadratic $S_2(\cdot)$. It has sigmoidal shape, similar to the error function and logistic functions (see below). It is much faster to compute than the logistic or other types of continuous sigmoid-shape functions, has very simple gradients and constant second derivatives. It should be useful as the neuron output function in MLP algorithms [21], significantly speeding up the calculations.

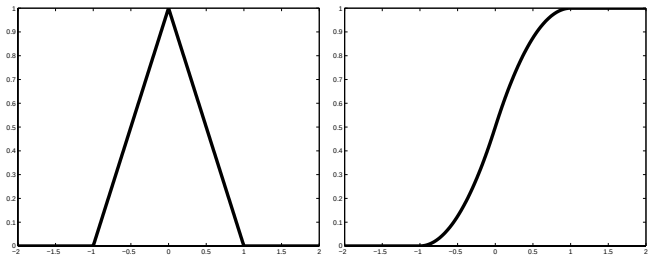

Fig. 2. Triangular uncertainty and the semiquadratic membership function that results from its integration.

$x\in(a,b)$ **rule, triangular** U**, semi-quadratic** S.

A crisp rule $R_{a,b}$ applied to the triangular input number $T_3(y-x;\Delta x)$ is true to a degree given by a combination of two soft trapezoidal functions $S_2(x-a;\Delta x)-S_2(x-b;\Delta x)$, which has very similar sigmoidal shape to the function shown in Fig. 5. If the support $2\Delta x$ of $T_3()$ is larger than $b-a$ the truth value is always lower than one.

$x\in(a,b)$ **rule, trapezoidal** U**, semi-linear-quadratic** S.

Trapezoidal uncertainty functions are constructed from a combination of uniform function centered at x, with $x\pm\Delta x$ flat top region, and linear slopes with non-zero values (support) between $[x-\Delta x-2t,x-\Delta x]$ on the left side and $[x+\Delta x,x+\Delta x+2t]$ on the right side. They may be constructed as a difference of two semi-linear functions:

$$U_4(y-x;\Delta x,t)=S_1(y-x+\Delta x+t;t)-S_1(y-x-\Delta x-t;t)$$

Crisp rules $x>a$ with such uncertainty are equivalent to fuzzy rules with semi-linear-quadratic membership function $S_{12}(x;\Delta x,t)$ that is a combination of piecewise constant, linear and quadratic functions resulting from integration of the $U_4(y-x;\Delta x,t)$ function (see Fig. 3).

The $S_{12}(x;\Delta x,t)$ function may also be useful for MLP training, because it is inexpensive to compute and the linear part gives the MLP network a possibility to find a linear solution, if it is sufficient. Regularization of network parameters [20] tends to make all weighted input values quite small; in effect only the linear part of the output function is used. For the linear part all second derivatives are zero, significantly simplifying

calculations of the Hessian matrix used in the second-order MLP training procedures [20].

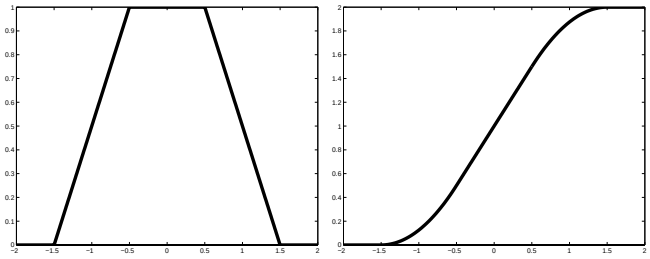

Fig. 3. Trapezoidal uncertainty and the sigmoidal quadratic-linear membership function.

$x > a$ **rule, Gaussian** U**, erf** S.

Gaussian distribution is quite commonly assumed for the uncertainty of real measurements. In this case the crisp x values are replaced by a Gaussian number centered on x with dispersion Δx. As in the case of triangular functions membership functions corresponding to Gaussian uncertainties have sigmoidal shape. Crisp logical rule $x > a$ with Gaussian number $G(y-x;\Delta x) = G(y;x\Delta x)$ as input is equivalent to a fuzzy rule with crisp x and $SG(x-a;\Delta x)$ membership function:

$$\begin{aligned} SG(x-a;\Delta x) &= \int_a^\infty G(y-x;\Delta x)dy \\ &= \frac{1}{2}\left[1-\mathrm{erf}\left(\frac{x-a}{\Delta x\sqrt{2}}\right)\right] \end{aligned} \tag{4}$$

where $\mathrm{erf}(u) = -\mathrm{erf}(-u)$ is the error function extended to negative values. Generalization of $x > a$ rules to rules $R_{a,b} = \{x | a < x < b\}$ involving interval is straightforward. For crisp rule $R_{a,b}$ the difference of the two $SG(x-a;\Delta x) - SG(x-b;\Delta x)$ functions has soft trapezoidal shape (compare Fig. 5), or a bell-shape for small $b-a$ difference,

$$\begin{aligned} &SG_2(x;a,b,\Delta x) \\ &= \int_a^\infty G(y-x;\Delta x)dy - \int_b^\infty G(y-x;\Delta x)dy \\ &= \frac{1}{2}\left[\mathrm{erf}\left(\frac{a-x}{\Delta x\sqrt{2}}\right) - \mathrm{erf}\left(\frac{b-x}{\Delta x\sqrt{2}}\right)\right] \end{aligned} \tag{5}$$

Error function erf is not used as neural output function because it is rather expensive to compute. Sigmoidal functions of the logistic type, $\sigma(x) = 1/(1+\exp(-x))$, are most commonly used in multilayer perceptrons. The function $SG(x-a;\Delta x)$ is approximated very well by the logistic function $SG(x-a;\Delta x) \approx \sigma(\beta(a-x))$, with $\beta = 1.7\Delta x$. The accuracy of this approximation is within 1% for all x and $\Delta x = 1$.

The assumption of Gaussian input uncertainty is thus equivalent to evaluation of the degree of truth by sigmoidal functions of the erf type, and to a very good approximation also by logistic functions. Thus the output of a typical MLP neuron is equal to the degree of fulfillment of the $x > a$ logical rule for input x that has Gaussian uncertainty.

A logistic membership function may be obtained from $x > a$ rule and input uncertainty distribution that is similar to a

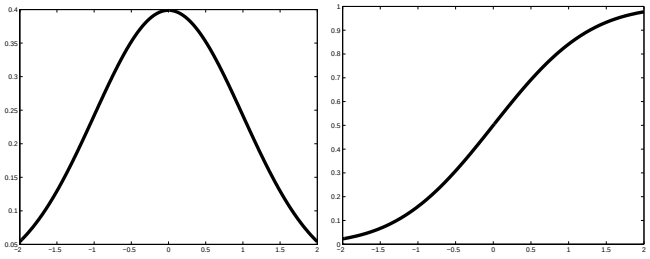

Fig. 4. Gaussian uncertainty and the erf sigmoidal membership function.

Gaussian $G(y-x;\Delta x)$. The product of logistic functions $U(y-x;\beta) = \sigma(\beta(y-x))(1-\sigma(\beta(y-x)))$ has bell-shape that for $\beta = 1.56$ slope of the logistic function differs from a standardized Gaussian function ($(\Delta x) = 1$) by less than 3.4% at each point. Taking $U(y-x;\beta,b) = \sigma(\beta x+b)(1-\sigma(\beta x-b))$ adds a flat maximum region around $x = 0$, changing the bell-shape into a soft trapezoidal shape. The difference of the two logistic functions, $STr(x;b) = \sigma(\beta x+b) - \sigma(\beta x-b)$, has the same soft trapezoidal shape. In fact these two functions are identical up to a normalization factor:

$$\frac{\sigma(x+b)-\sigma(x-b)}{\sigma(b)-\sigma(-b)} = \frac{\sigma(x+b)(1-\sigma(x-b))}{\sigma(b)(1-\sigma(-b))} \tag{6}$$

The proof is straightforward although a bit tedious. The denominator goes to zero for small b, but this expression is quite stable from numerical point of view even for $b = 10^{-6}$. Such soft trapezoid functions are useful as neural output functions [21]. It is also easy to prove that the logistic function of a sum of two variables is equal to a ratio of products:

$$\begin{aligned} \sigma(x+y) &= \sigma(x) \star \sigma(y) \\ &= \frac{\sigma(x)\sigma(y)}{\sigma(x)\sigma(y)+(1-\sigma(x))(1-\sigma(y))} \end{aligned} \tag{7}$$

The $\star$ operator may be regarded as a fuzzy aggregation operator [8]. The logistic form of uncertainty distributions is bell-shaped for $b = 0$ and has soft-trapezoidal shape for $b \approx 1$ or larger. Assuming crisp logic rule $x > a$ and soft trapezoidal input uncertainty $STr(y-x;b)$ with x as the middle point, the membership function for the fullfilment of the rule is obtained from integration:

$$\begin{aligned} SLE(x-a;b) &= \frac{1}{2b}\int_a^{-\infty} STr(y-x;b)dy = \\ &= \frac{1}{2b}\int_a^{-\infty} (\sigma(y-x+b) - \sigma(y-x-b)) \\ &= \frac{1}{2b}\ln\left[\frac{1+e^{a-x+b}}{1+e^{a-x-b}}\right] \end{aligned} \tag{8}$$

This logarithmic-exponential function has sigmoidal shape with the linear part in the middle, similar to the semilinear function with softened edges (Fig. 5). It is continuous and has almost linear central part, making it very suitable as the output function for MLP neurons. The linear part should prevent too quick convergence to the local minima of the MLP error

function, providing non-linearity only when they are necessary, and linear solutions when they are sufficient.

Gaussian functions are frequently taken as membership functions. Although approximations to Gaussians may be obtained from various natural assumptions about input uncertainties the exact form of the Gaussian functions is obtained only with an assumption of $U(y;x,\omega) \propto yG(y;x,\Delta x)$ type of uncertainty that seems to be hard to justify. Various other bell-shaped uncertainty distributions may be considered, and their dual membership functions found, but perhaps those mentioned above are the most important.

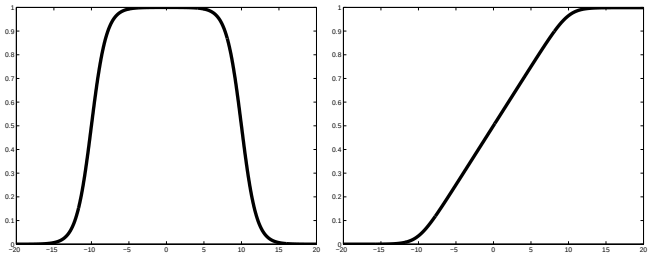

Fig. 5. Soft trapezoidal uncertainty centered at $x = 0$ with $b = 5$, and the log-exp membership function.

In this section crisp rules $x \in (a,\infty)$ and $x \in (a,b)$ were considered. Generalization of these results to the sums of intervals $x \in (a_1,b_1) \vee x \in (a_2,b_2)$ etc. is not difficult. Any S-norm may be used to aggregate the results. Such rules have network implementation with single input and a few nodes representing fuzzy rules or neurons in neural networks. Rules with $\leq$ instead of $<$ conditions are handled by placing intervals between discrete values.

B. Multidimensional situation.

Crisp conjunctive rules $R = R_1 \wedge \ldots \wedge R_k$, where each R_i is a condition of the $X_i \in (a_i,b_i)$ type (where a_i or b_i may be infinite), are easy to handle if all conditions are based on independent, uncorrelated features X_i. Each feature has its own uncertainty function $U(Y_i;X_i,\omega)$ and the probability that $R(\mathbf{X})$ is true, is equal to the product of the probabilities of $\tau_i(X_i)$ for $i = 1 \ldots k$. Thus a fuzzy rule with $\tau_i(X_i)$ membership functions may replace the crisp rule plus uncertainty functions. For example, if $(X_1 > a_1 \wedge X_2 > a_2)$ rule premises are used with uniform assumption about uncertainty $U(Y_i - X_i, \Delta X_i)$ then as the result of integration a product of two semi-linear functions $S_1(X_1 - a_1;\Delta X_1)S_1(X_2 - a_2;\Delta X_2)$ is obtained.

Thus a natural T-norm for fuzzy rules equivalent to crisp logic rules applied to uncertain inputs is based on simple product of membership functions. Products of various membership functions derived in previous section replace the need to calculate the degree of fulfillment of crisp rules by integration over input distributions. Each conjunctive rule may be implemented as a product node, and for independent rules the sum of outputs from nodes that share the same conclusion gives the final answer.

Some features occurring in the rule R may be mutually dependent. If a few strongly dependent features are used in a single rule, product of $\tau_i(X_i)$ probabilities may become quite small. Other T-norms may be useful in such cases, although simple probabilistic interpretation may be lost. De-correlation of input features, used frequently in signal analysis, solves that problem at the expense of introduction of linear combinations of features. Selection of input variables used in rule conditions partially solves the problem of strongly correlated features.

Many rule extraction algorithms (for example, decision trees) partition the feature space into disjoint areas. For any input vector $\mathbf{X}$ a single rule is active (although for inputs with uncertainties activities of different rules should be taken into account). Algorithms that generate a set of conjunctive crisp rules R^m covering the same regions of the feature space require special treatment. Summing and normalizing probabilities obtained from different rules may give results quite different from the Monte Carlo simulations. Care should be taken to count each region only once. Given two rules $R^1(X), R^2(X)$ for the same class C the probability $p(C|\mathbf{X};\hat{M})$ is $P(\mathbf{X} \in R^1) + P(\mathbf{X} \in R^2) - P(\mathbf{X} \in R^1 \cap R^2)$.

These probabilities are derived from classification system $\hat{M}$ rather then directly from data. Estimation of probabilities with Monte Carlo sampling from high-dimensional distributions is a slowly convergent process, therefore whenever possible analytical formulas should be used. In the limit of very large input uncertainty the whole data range is included. Asymptotic behavior of probabilities assigned by classifiers depends on many factors. Fuzzy rule systems with localized (compact support) membership functions, and radial basis function networks (RBF) that are equivalent to such systems, may give estimations that converge to the *a priori* class probabilities for the dataset. Most classifiers, including rule-based systems that use membership functions with non-compact support, decision trees, nearest neighbor methods and multi-layer perceptron neural networks do not have correct asymptotic behavior.

Input uncertainty for feature $x = X_i$ is given by a membership function $\tau(y;x,\omega)$, dependent on x, with ω parameters describing its shape. This function represents the degree of belief (or sometimes probability) that values y may still be taken as x. Membership functions (MFs) may in principle have arbitrary shape, estimated from observations. Because the features of the input vector $\mathbf{X} = \{X_1, X_2, \ldots X_n\}$ are uncertain instead of using a single input vector $\mathbf{X}$ it would be more appropriate to sample from the multidimensional distribution defined by all appropriate MFs $\tau_i(y;X_i,\omega_i)$ (symbolic features may not be appropriate because they usually cannot be fuzzified). From this multi-dimensional distribution a large number of input vectors $\mathbf{Y} \in O_\tau(\mathbf{X})$ may be generated in the neighborhood $O_\tau(\mathbf{X})$ defined by MFs around the query vector $\mathbf{X}$.

Any classification system $\hat{M}$ that predicts class labels $\hat{M}(\mathbf{X}) = C_k$, including sets of crisp decision rules, decision trees, neural networks or statistical black-box classification systems, may be applied to the set of $\mathbf{Y}$ vectors. If for N vectors $\mathbf{Y}^{(i)}$ from this set class C_k has been predicted N_k times, $p(C_k|\mathbf{X};\hat{M},\omega) = N_k/N$ is an estimation of the probability that a vector from the neighborhood $O_\tau(\mathbf{X})$ will be of assigned to class C_k by the classification system $\hat{M}$ with parameters ω (including parameters of the sampling procedure). Similar estimation may also be done if the $\hat{M}$ system predicts membership values or probabilities.

This Monte Carlo sampling procedure facilitates the reconstruction of multidimensional membership functions for any classification system, not just logical rule-based systems. Analytical results for single inputs obtained in the previous subsection may be approximated using such numerical simulations, with $U(y-x;\omega)$ input uncertainty distributions and classifiers based on rules with a single premise $x \in (a,b)$. Generalizing these results a good guiding principle is to require that probabilities generated from Monte Carlo sampling should be the same as those obtained from the equivalent fuzzy system. The goal here is to obtain the same results with crisp logic system applied to uncertain inputs as with the fuzzy system applied to crisp inputs.

III. Relation with multi-layer perceptrons.

Equivalence of fuzzy logic systems with radial basis function (RBF) networks is well known and has been formally proven for Gaussian and other localized functions [5]. Each node of the RBF network corresponds to a fuzzy rule. In practice feature selection in RBF networks is rarely done, while in fuzzy rule-based systems it is of primary importance. RBF networks are based on similarity evaluation, while multilayer perceptrons, the most popular neural network models, are based on non-linear soft discrimination. RBF nodes frequently use multidimensional Gaussian functions, calculating products of one-dimensional membership functions. Such nodes calculate the truth value of conjunctive logical rules applied to uncertain inputs.

Results of the previous section showed that membership functions based on various trapezoidal and soft trapezoidal functions arise for interval-based premises $x \in (a,b)$ under many assumptions about input uncertainty. Although products of these functions are not radial, they can still be used as output functions of neurons in basis function (RBF-like) network architectures [21]. Functions that are products of components depending on single variable, $f(\mathbf{X}) = f_1(X_1) f_2(X_2) \ldots f_N(X_N)$, are called separable. Radial basis functions are usually not separable, with an important exception of the multidimensional Gaussian function with diagonal covariance matrix that is a product of one-dimensional components. Feature Space Mapping networks [14], [15] are based on separable functions, therefore their neurons evaluate the truth of conjunctive logical rules. Products of $STr(X_i;\omega_i)$ soft trapezoidal functions are used in the Incremental Network (IncNet) neural network implemented in the Ghostminer package [16], [17].

Basis function networks with product nodes implement conjunctive rules. Multi-layer perceptrons are based on threshold logic. Increasing input uncertainty from zero to some finite value is equivalent to the transition from step-like threshold functions $\Theta(\mathbf{W}\cdot\mathbf{X})$ to the smooth sigmoidal functions $\sigma(\mathbf{W}\cdot\mathbf{X})$. This transition converts networks implementing crisp logical functions using threshold logic into MLP networks. The theory of logical networks has been well developed in the early days of neural models, starting with McCulloch and W. Pitts [19]. This theory became important for construction of digital circuits. Relations between fuzzy logic and their network implementations have never been analyzed to comparable extent.

Artificial neurons, or network nodes, are the basic building blocks of MLP networks [20]. Motivated by functions of biological neurons artificial neurons implement sigmoidal output functions, usually of the logistic type (for a survey of neural functions see [21]). Other types of sigmoidal neuron output functions, such as hyperbolic tangent or arctangent functions, give essentially the same results. Semi-linear functions are sometimes used as an approximation to the continuous sigmoidal functions. All these functions estimate the truth of a crisp logical rule $x > a$ under various input uncertainty assumptions, as derived in the previous section.

Consider first the simplest example: one-dimensional case, one neuron network. A threshold function $\Theta(x-a)$ implementing logical rule $x > a$ in case of uniform uncertainty is equivalent to a fuzzy rule with semi-linear membership function. For Gaussian uncertainties sigmoidal erf functions are obtained, and they are approximated quite well with logistic functions. Presenting fuzzy membership functions in graphical form as nodes of a network (Fig. 6) allows for implementation of the same fuzzy logical functions.

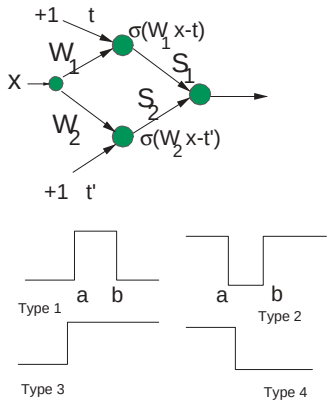

Fig. 6. Neural implementation of 4 types of crisp conditions $Wx \in (a,b)$, with $S_1 = -S2 = 1$ and $\sigma(\cdot)$ with infinite slopes: type 1 for $W_1 = W_2 = 1$, $t = a$, $t' = b$, type 2 for $W_1 = -1$, $W_2 = 1$, $t = -a, t' = b$, type 3 is obtained from type 1 with $b = \infty$ and type 4 is obtained from type 2 with $b = -\infty$.

Input weights provide scaling for feature values, and the sign of weight determines the type of inequality and the threshold of the neuron determines the value of a (see Fig.6). The truth value of premises is thus measured by the value of sigmoidal function $S(Wx-a)$ for input x. The network based on such nodes sums the conclusions of all rules referring to the same class in the output layer. The output weights estimate the relative importance of these rules in reaching the final conclusion. For $Wx \in (a,b)$ rule a combination of two $S(W_1x-t) - S(W_2x-t')$ neurons should be used, and this can either be implemented by a single node, or by a linear neuron in the output layer. For example, if $W_1 = +1, W_2 = -1$ and thresholds are a,b a rule with $x \in (a,b)$ is implemented by the network with one input, two hidden nodes and one linear output node, as in Fig. 6.

More hidden neurons may be added to implement other rule

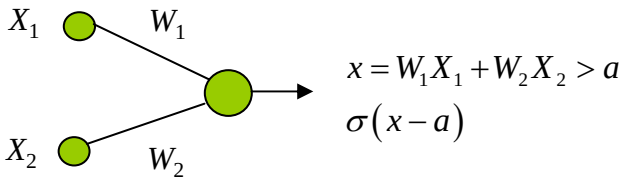

Fig. 7. Two independent inputs and one hidden neuron implementing threshold logic.

Conjunctive rules $R = R_1 \wedge R_2$, where each R_i involves an independent variable and interval $R_i = X_i \in (a_i, b_i)$, applied to X_i inputs with some uncertainty, lead to a product of two membership functions. This is what neural networks based on separable transfer functions, and fuzzy rule based systems, normally do: evaluate the evidence provided by inputs X_i using some membership functions, and get the conclusion combining the results using some T-norm, in this case a product. There is another option that is not so popular in fuzzy logic: using threshold logic, as it is done in mulit-layer perceptrons. Neurons in MLP networks implement fuzzy threshold logic to evaluate the truth of crisp threshold logic rules in presence of input uncertainties.

MLP transfer functions $f(\mathbf{X})$ map vectors $\mathbf{X}$ to scalar values $I(\mathbf{X})$ called activation, which are then processed by the threshold output function $o(I)$, so that $f(\mathbf{X}) = o(I(\mathbf{X}))$ [21]. For a single input, activation is simply the weighted input value. The output function has usually sigmoidal shape. For two or more inputs activation is usually taken as a linear combination $x = \sum W_i X_i = \mathbf{W} \cdot \mathbf{X}$. Thus N-dimensional threshold neurons are essentially single input neurons applied to some scalar activation values. Linear combination of inputs is sometimes used in fuzzy logic when rules are applied to the pre-processed signal, time series or image data, for example after extraction of principal components or independent components. Rules $x > a$ are then defined along the $\mathbf{W}$ direction in the feature space.

How should the input uncertainty distribution $U(y; x, \omega)$ for $x = W_1X_1 + W_2X_2$ variable be calculated? Given two independent random variables Y_1, Y_2 and their corresponding distributions $U(y; X_1, \omega_1)$ and $U(y; X_2, \omega_2)$, the distribution of random variable $Z = Y_1 + Y_2$ is given by the convolution:

$$U(z; X_1 + X_2, \omega) = \int_{-\infty}^{+\infty} U(z-y; X_1, \omega_1) U(y; X_2, \omega_2) dy \quad (9)$$

Convolution of two uniform distributions with identical width gives triangular uncertainty functions, and with different width trapezoidal functions. This shape comes from projection of two-dimensional rectangular joint distribution of the Y_1 and Y_2 variables on $W_1Y_1 + W_2Y_2$ line. Adding third input requires convolution of traingular and rectangular function, resulting in soft-trapezioidal function made from semiquadratic fragments (compare Fig.8). For larger number of inputs soft-trapezoidal shape of uncertainty distribution is preserved, but the higher-order polynomials should be used to approximate it. Analytical formulas for such distributions may be derived, but they will not be very useful, because these functions are composed from many fragments.

Thus although each of the original variables has uniform uncertainty distribution, their linear combination has uncertainty distribution of the soft trapezoidal shape. Integration of this distribution leads to the logarithmic-exponential type of sigmoidal function Eq. 8 that should serve as the membership function. The sum of several normalized inputs with similar uniform uncertainty has always bell-shaped uncertainty distribution, similar to Gaussian (Fig.9). This justifies the use of logistic or similar sigmoidal functions that result from integration of such distributions. In a typical situation weights of the linear combination of inputs $x = \mathbf{W} \cdot \mathbf{X}$ have different values (in the Bayesian approach to MLP training it is assumed that weight distribution is Gaussian) and final uncertainties $U(x; \omega)$ have shapes that range from triangular, through bell-shape to soft triangular. If uncertainties of input variables are significantly different, or if the weights are quite different, an approximation to the logarithmic-exponential functions with linear area around the rule threshold should be used.

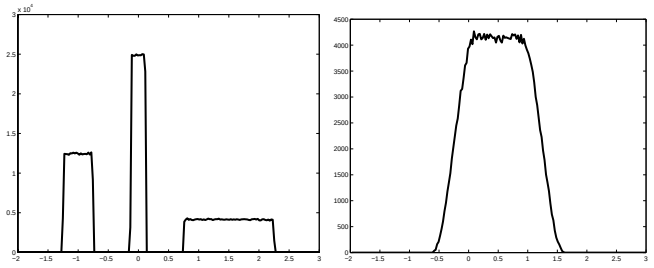

Fig. 8. 3 inputs with uniform uncertainty, but different centers and width, after convolution give semi-quadratic soft trapezoidal uncertainty distribution.

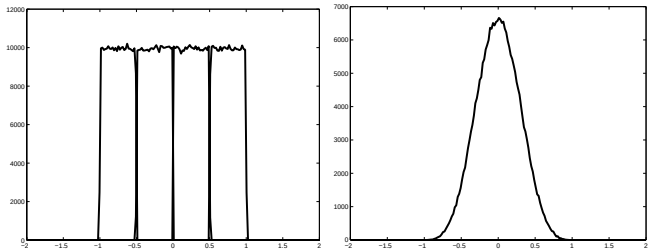

Fig. 9. 4 inputs with uniform uncertainty and identical width, centered at ± 0.25 and ± 0.75, after convolution giving Gaussian-like uncertainty distribution.

For triangular uncertainties or more complex types of uncertainties of input variables qualitatively similar behavior is observed. For example, taking 4 inputs with identical Gaussian dispersions for $X_1 = -0.75$, $X_2 = -0.25$, $X_3 = 0.25$ and $X_4 = 0.75$, and different weights (Fig. 10), gives after convolution bell-shaped Gaussian-like distribution. Thus linear combination of many input variables with any type of uncertainties, uniform, triangular, trapezoidal, or Gaussian, leads to bell shaped distributions that after integration give sigmoidal type of membership functions. Thus a rule $x > a$, with x equal to a weighted combination $\mathbf{W} \cdot \mathbf{X}$ with arbitrary uncertainties,

is always approximated by a fuzzy rule $F \leftrightarrow S(x-a) > \phi$, where $S(\cdot)$ is some type of sigmoidal function. This is a soft hyperplane used by MLP neurons.

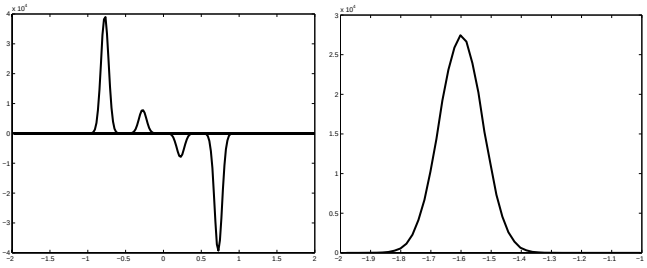

Fig. 10. 4 inputs with Gaussian uncertainty, identical dispersions, centered at ± 0.25 and ± 0.75, with $1, 0.2, -0.2, -1$ weights, after convolution give Gaussian-like uncertainty distribution.

Combination of inputs creates new linguistic variables that may not have sense, except for providing some discriminating hyperplanes. MLP nodes based on threshold logic divide the feature space into two half-spaces using hyperplanes, while neural networks based on separable functions divide the feature space into areas where products of membership functions are larger than some thresholds – for rectangular membership functions, these areas are hyperrectangles, providing rules of classical logic. Soft threshold logic rules $S(W \cdot \mathbf{X} - a) > \phi$ may sometimes simplify logical interpretation, and although they may be re-interpreted using conjunctive logic at the expense of special aggregation operators [8] they will not become more comprehensible. If all inputs reaching a neuron belong to the same type, linear combination, equivalent to rotation and rescaling, may provide new, interesting features that have some interpretation.

Adding more neurons in the hidden layer is equivalent to more fuzzy rules $F_k \leftrightarrow \sigma(f_i - a_i) > \phi_i$, with $f_i = \mathbf{W}^{(i)} \cdot \mathbf{X}$. Rules leading to the same conclusion (same class membership) are combined together in the output layer. MLP assigns weights to rule conclusions and makes final aggregation of evidence in two ways. Linear output neurons combine weighted evidence, and either a maximum is selected or final class memberships are calculated after some normalization. Alternatively, soft threshold logic is used to create conjunctive rules (for high thresholds) or disjunctive rules (for low thresholds). The need for soft threshold logic is motivated again by propagation of uncertainty through the hidden layer.

In fuzzy logic various forms of sophisticated aggregation operators are in use, for example ordered weighted aggregation (OWA) operators [22]. They may be more or-like or and-like, similarly as the weighted activation aggregation. Activity of hidden neurons, or the degree to which rules implemented by hidden neurons are fulfilled, form an image of input vectors in the hidden space. The goal is to create separable clusters of images of the input vectors in the hidden space. In MLPs output neurons provide discriminating functions that separate these clusters. From fuzzy logic perspective rule conclusions are aggregated using weights and thresholds that maximize the number of correct answers to a rule: $y > a$, where y is a weighted combination of rule conclusions. MLPs provide a hierarchical system of such rules. Adding more network layers is equivalent to more levels in this hierarchy that includes rules about intermediate rule conclusions (from previous hidden layers), not only about data. Such intermediate conclusions may have some sense, especially if the network is pruned leaving only most important connections [10].

Hierarchical fuzzy systems are an active research topic in fuzzy logic, aimed at reduction of the exponential number of rules arising in control and other applications. If m membership functions are defined for each of the n inputs then the number of possible fuzzy rules is m^n. There are several ways to go around this problem [23], [24], but the hierarchical fuzzy systems approached gains recently most interest [25]. Such systems process inputs in lower-dimensional subspaces, combining the results in a binary-tree fashion. In this process comprehensibility or the physical interpretation and the ability to design such systems without much training is easily lost, although there are some proposals to restore it [26]. MLP avoids problems with combinatorial explosion, but the price is sometimes high: weighted combinations of inputs may not be easy to understand, and optimal weights cannot be designed but have to be learned. A compromise is offered by neural architectures that enforce simple, skeleton networks structure, that frequently can be analyzed in details and converted to a set of logical rules [10].

IV. Examples of applications.

Good estimation of input uncertainty is in many cases possible. For example, medical tests have known accuracy and models of uncertainty distributions may be constructed. Explicit model for uncertainty of the test may include not only the actual measurement, but many other factors, such as the type of treatment, physical exercise, or food and drinks consumed prior to the test. Specific membership functions may then be constructed to evaluate more accurately various risks for measured input values.

Conjunctive logic rules are perhaps most frequently used, but in some situations M-of-N type of rules, employing threshold logic, are more natural. Rule conditions may be treated as constraints rather than absolute requirements. If not all constraints may be fulfilled solutions that satisfy most of the constraints are searched for. For example, information retrieval systems (including all of the internet search engines) are based on such approach. If documents with all N keywords are not found then links to documents with N-1 keywords are displayed, followed by links to documents with smaller number of keywords, until a minimum of M keywords is found. Thus the queries are handled by threshold-based logic rather than conjunctive logic. The uncertainty of inputs may be expressed in the alternative keywords and may be captured using fuzzy rules operating on context vectors.

Medical personnel frequently uses logical rules based on various thresholds for different tests. Medical textbooks are full of rules of M-of-N type:
If at least 3 symptoms of the 5 from the set: $\{s_1, s_2, s_3, s_4, s_5\}$ are present, then conclusion follows.
Each of the symptoms may be of the fuzzy linguistic variable type: high fever, high blood pressure, high cholesterol

level etc. Network that represents such rule should contain 5 pairs of nodes that filter measured inputs (Fig.6) to provide values of the membership functions, followed by the output neuron that combines the evidence and compares it with the threshold $x > 3$. Knowing the uncertainty of measured values slope of corresponding sigmoidal functions may be set. The backpropagation training algorithm will adjust the weights that in the original rule are all 1, tuning the rule to match its prediction to the data. As a result the network may put more emphasis on high blood pressure than on the cholesterol level. There is nothing mysterious about such networks. Their recommendations are at least as comprehensible as those that follow from fuzzy systems.

Uncertainties may have different origin (see [3]) and sometimes cannot be reliably estimated. For example, evaluation of questionnaires, such as census data, medical or psychological surveys, followed by averaging of some responses, leads to numerical values of observations of unknown accuracy. This problem may be approached via fuzzy sets of the second type [6]. On the other hand uncertainties $s_{\mathbf{X}}$ of the values of features may be used as additional adaptive parameters that may be optimized. This is done in several steps:

- Prepare a training data base containing results of surveys reduced to numerical coefficients and categorized in a reliable way.
- Extracted from this data initial crisp logic rules, using decision trees [12], [13], MLP2LN neural networks [11] or other approaches [10].
- Assume some type of uncertainty distributions, for example triangular or Gaussian, and use small initial uncertainties s_i to fuzzify crisp rules using membership functions that correspond to input uncertainties of the selected type.
- Optimize a cost function $E(\mathbf{s},\omega)$ to find the best values for model parameters, including the uncertainties.

Soft cost function may be based on a sum of predicted probabilities or normalized membership values:

$$E(\mathbf{s},\omega) = \sum_{\mathbf{X}} \sum_{i} \left(p(C_i|\mathbf{X};\mathbf{s},\omega) - CL_i(\mathbf{X})\right)^2 \qquad (10)$$

where ω includes intervals defining linguistic variables, weights and thresholds, s_x are uncertainties of inputs, $CL_i(\mathbf{X}) \in [0,1]$ is a label for the training vector $\mathbf{X}$ (several non-zero entries for different class may be used), and $p(C_i|\mathbf{X};\mathbf{s},\omega)$ is calculated using the neural network or a system of fuzzy rules. This error function may be optimized using backpropagation gradient-based techniques.

If all features represent measurements of the same type all s_i may be taken as a percentage of the range of each feature, $s_i = s(\max(X_i) - \min(X_i))$, and one-dimensional minimization of the error function over a single s parameter is performed. This minimization may either be added to the training procedure, or done by plotting the dependence of $E(s)$ and selecting the minimum. In the limit of a small s sigmoidal functions are very steep, acting as step functions, and minimization of the soft error function (10) becomes equivalent to minimization of the number of classification errors. Optimal s value that minimizes the error function gives an estimation of the unknown uncertainty.

This approach to extraction and optimization of rules has been applied to analysis of Minnesota Multiphasic Personality Inventory (MMPI) psychometric data, consisting of 550 questions with 3 possible answers (yes, no, don't know) each [18]. Computerized versions of this test assist only in information acquisition, but evaluation of results is still done by an experienced clinical psychologist. The raw MMPI data is used to compute 14 real-valued coefficients, called "psychometric scales". These coefficients are often displayed as a histogram (called "a psychogram") allowing skilled psychologists to diagnose specific problems, such as neurosis, drug addiction or criminal tendencies.

The data was collected in the Academic Psychological Clinic of Nicholaus Copernicus University, Torun, Poland (smaller version of this data has been analyzed previously [10]). Expert psychologists provided about 1600 cases belonging to 27 classes for women the same number of cases divided into 28 classes for man (about 60 cases/class). Rules were initially generated using C4.5 classification tree [12], and SSV decision tree [13], with another set of crisp rules generated by the Feature Space Mapping (FSM) neurofuzzy network [14], [15] using rectangular membership functions. Both SSV and FSM algorithms are implemented in the Ghostminer data mining package [16] used to generate all results described below. Only simple rules are of interest to psychologists, because each set of rules for a given class has to be commented upon, providing verbal interpretation useful for support of diagnosis. Some rules covered only a few cases from the database, therefore pruning and re-optimization was performed.

C4.5 creates 2-3 rules per class involving between 2 to 9 attributes, and achieving 93.0% of correct responses. Agreement between two human experts analyzing this type of data is usually below 80%. Gaussian distribution of uncertainty in inputs was assumed, and the corresponding erf membership functions 5 approximated by differences of logistic functions to simplify calculations. With dispersion around $s = 1\%$ of the data range improves results by about 1%. FSM network was used with rectangular membership functions to generate crisp rules. These rules may overlap, therefore high membership degrees in more than one class are possible. 3-4 per class were created, agreeing in 95% with original diagnosis. Gaussian fuzzification at the level of 1.1-1.5% increases this accuracy by 2.5%.

Rectangular membership functions of crisp rules are converted to the soft trapezoidal functions corresponding to the optimal uncertainty of about 1.5%. This uncertainty is sufficiently small to make the verbal interpretation of fuzzy rules still quite easy. The true uncertainty of psychometric scales is unknown and the reliability of the training data is also hard to estimate. For small input uncertainties rules predict one or more classes, while for large uncertainties many classes have comparable probabilities (Fig. 11). With input uncertainty set to zero crisp rules are used. The query case in Fig. 11) is found in the region where rules for two different classes overlap. Using crisp rules such solution should be preferable to predictions of a single class only – the evidence available in the data is not sufficient to favor any of the two classes. Assuming small uncertainty $s = 1\%$ breaks the tie between

the two classes, and increasing the uncertainty to 2 and 4% shows two more classes for which significant membership is predicted.

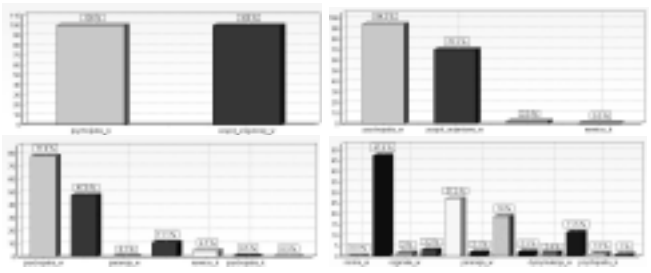

Fig. 11. Influence of input uncertainty on predictions of class memberships. Top left: no uncertainty - two equally probable classes are predicted; top right: optimal 1% uncertainty, first class becomes more probable than the second; bottom: 2% and 4% uncertainty assumed, leading to several new classes with smaller membership values.

The rule with largest membership, shown for $s = 3\%$ Gaussian uncertainties in Fig. 12, has 5 conditions (out of 14 possible). Feature values of the query case are connected with line segments, Gaussian distributions are attached to 5 feature values that appear in the rule under consideration. Two intervals (for Ps and Pt features) include the measured values for the evaluated case rather close to their boundaries, therefore only 56.4 and 66.7% of the Gaussian is captured inside the interval. As a result membership value of the actual case in this rule is only 38%.

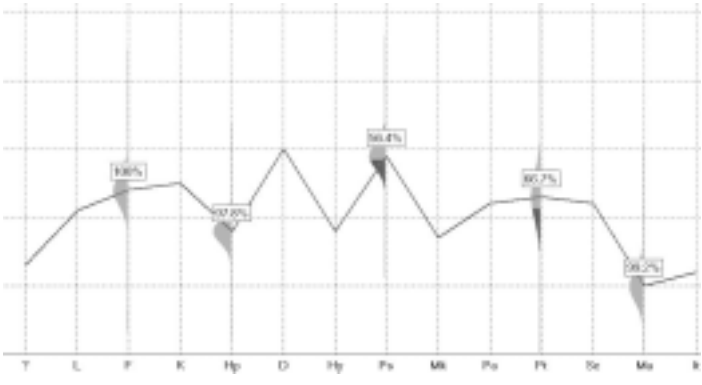

Fig. 12. Psychogram with rule conditions and fuzzified input for $s = 3\%$ displayed.

These estimates of membership values give an idea how strongly rules support the assignment of vector $\mathbf{X}$ to different classes C_k. The whole fuzzy rule based system may be presented as an MLP network.

Instead of displaying membership values for a given uncertainty it may be useful to plot $p(C_k|\mathbf{X};\mathbf{s};\hat{M})$ as a function of uncertainty of one or more input feature values. Such graph shows the stability of predictions of the system $\hat{M}$ around the input $\mathbf{X}$. Cases far from decision borders show only slow decrease of predicted membership functions, but cases near decision border show significant decrease of the dominant class memberships at the expense of new classes. This technique may be useful in evaluating the type of errors that the system makes.

For example, in the Leukemia gene expression data [27] two classes, acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML) should be distinguished, given 7129 features (gene expression levels from microarray experiments). Analysis of this data made with different methods available in the Ghostminer package [**?**] showed that using only one single feature all 38 training samples, and all but 3 of the 34 test samples (AML mistaken for ALL cases), are classified correctly. Using SSV decision tree a crisp logical rule based on a single threshold for feature 4847 was found giving such high accuracy. Can fuzzification help? It is easy to check that no assumptions about uncertainty of input data will create a fuzzy rule that makes less than 3 errors on the test set. Increasing the number of features (gene expression values) to the most promising 10 features, and using Support Vector Machine based on Gaussian kernels, a solution with no training and a single test error was found.

This one test error may result from insufficient input information, the inability of SVM to provide correct decision borders, or training data that is too small and does not represent the true data distribution. While the first two errors in the test set were indeed due to the insufficient information this last error seems to be of a different type. It may be a mislabeled data case, or an error in the diagnosis, a different type of leukemia that does not fit to any of the two classes. It may also be a very rare and untypical case of acute myeloid leukemia that should be distinguished as a new subtype, leading perhaps to the similar medical condition. This is indicated by the following observation. 5% input uncertainty for all of the 10 gene expression values has been taken (this is sufficiently large to cover in two-dimensional scatterograms most of the data from the ALL class), and 1000 vectors in the neighborhood of the selected test AML vector has been generated. All of these vectors are assigned by SVM and other methods to the (wrong) ALL class. This vector is not near the decision border, but placed firmly in the feature space area that all classification methods assign to the AML class, containing many vectors from this class and no vectors from ALL class. Generating 1000 points around the other two vectors shows that they were close to the decision border (significant number of vectors was assigned to the ALL class), therefore increasing the number of features from 1 to 10 helped to separate them correctly. It seems rather unlikely that new information (either training data or adding more features) could change classification of this one vector, because such change in the decision boarders would have to influence classification of other vectors in the neighborhood.

V. Discussion.

Sets of crisp logic rules applied to uncertain inputs are equivalent to fuzzy rules applied to crisp inputs. Integration of uncertainty distribution for a fixed rule threshold or interval gives probability, or degree of fulfillment, of a crisp rule. The same result may be obtained directly as a value of corresponding membership function for a given input. Different assumptions about input uncertainty lead in a natural way to different types of membership functions, but all of them have

sigmoidal shapes. Fuzzification of input values should give the same probabilities as the Monte Carlo procedure performed for input vectors sampled from uncertainty distributions centered at the measured values. In many practical cases analytical formulas for fuzzy membership functions have been derived.

With single input and uniform uncertainty $U(y;x,\Delta x)$ semi-linear membership function $S(x-a;\Delta x)$ should be used for estimation of the degree of truth of $x>a$ crisp rule. In all other cases smoother membership functions of sigmoidal shape are needed, justifying the use of sigmoidal functions from the logical point of view. The use of sigmoidal functions has also been justified using approximation theory [20], [5]. The fact that MLPs are universal approximators is in itself not surprising (it is difficult to make a basis set expansion that does not have universal approximation property). Favorable rates of convergence of expansions based on sigmoidal functions in highly dimensional spaces are more important [28]. These results tell us why the use of MLPs for approximation problems is a good idea, while the analysis done in this paper shows why it is natural to generalize hierarchical sets of logical rules using MLP network implementation with sigmoidal functions.

Several new types of membership functions have been introduced here, resulting from integration of uncertainty distributions. Of particular practical importance is the quadratic sigmoidal function resulting from integration of triangular uncertainties, and log-exp sigmoidal function with extended linear part resulting from integration of soft trapezoidal uncertainty distribution. Linear combination of many inputs has after convolution soft-trapezoidal uncertainty distribution that lead to such log-exp functions.

Crisp logic rules applied to uncertain inputs may be replaced by fuzzy rules with sigmoidal membership functions. Sets of crisp logic rules are equivalent to logical networks. Sets of equivalent fuzzy rules are equivalent to single hidden layer perceptron networks. Hierarchical sets of rules are equivalent to multi-layer perceptrons. Backpropagation training algorithm is a specific way to optimize this set. Threshold logic is implemented by neurons with sigmoidal output functions in MLP networks, while networks based on separable functions implement classical crisp conjunctive logic rules.

Functions performed by neural nodes may be understood at least in three ways. First, logical neurons implement threshold logic allowing for realization of $M-of-N$ rules: if M out of N premises are true than conclusion is also true. Some concepts, such as "majority of inputs", are much easier to express using $M-of-N$ threshold logic rules, than using prepositional logic rules. If input uncertainty is taken into account these threshold logical neurons should be replaced by soft sigmoidal neurons. Second, linear combination of inputs may provide new features, containing more information than original features. This combination is then propagated through sigmoidal membership function, giving the degree to which a rule is fulfilled, used as input to the next layer. Comprehensibility of rules is usually lost due to the linear combination of input features. However, pruning the network, or requiring explicitly that only some groups of inputs should be mixed, helps to restore comprehensibility [10]. Third, a specific form of aggregation operator may be used to interpret the $M-of-N$ rules as aggregation of individual conditions [8], [9]. Arguably, conjunctive rules obtained in this way do not make these rules easier to understand.

Type-2 fuzzy sets have membership functions that are themselves fuzzy [6]. Knowledge mining using surveys has been one of the main applications of the fuzzy systems of the second type. An alternative approach has been presented here. Leaving uncertainty on both the input and the rule side leads to similar effects without decreasing comprehensibility of the rule-based system. An application of these ideas to the analysis of questionnaire based surveys has been presented. Similar applications are possible whenever a set of crisp logic rules is given. For example, decision trees are very popular data mining tools that provide crisp rules. Methods developed in this paper may be used to fuzzify predictions made by decision trees and improve calculation of classification probabilities beyond evaluation based on the purity of tree leaves.

Input uncertainties provide a principled way to fuzzify sets of crisp rules and logical networks. Keeping fuzziness at the input side makes fuzzy systems and neural networks easier to understand. This idea may be used to convert MLP network into equivalent logical network, with input uncertainties proportional to the inverse of the norm of incoming weights. MLP neural networks are in many cases more comprehensible than hierarchical sets of fuzzy rules.

ACKNOWLEDGMENTS

The program for psychometric data analysis has been written by Krzysztof Grąbczewski and Rafał Adamczak from the Department of Informatics, Nicholaus Copernicus University.

REFERENCES

[1] B. Kosko, *Neural Networks and Fuzzy Systems*. Engelwood Cliffs, NJ: Prentice Hall, 1992.
[2] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press, 1996.
[3] Klir, G. J. and M. J. Wierman, "Uncertainty-Based Information", Heidelberg, Germany: Physica-Verlag, 1998.
[4] S.K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition*. New York: J. Wiley, 1999.
[5] V. Kecman, *Learning and Soft Computing*, Cambridge, MA: MIT Press, 2001.
[6] J. Mendel, "Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions". NJ: Prentice Hall, 2001.
[7] J-S. R. Jang and C.T. Sun, "Functional Equivalence Between Radial Basis Function Neural Networks and Fuzzy Inference Systems," *IEEE Trans. on Neural Networks*, vol. 4, pp. 156–158, 1993.
[8] J.M. Benitez, J.L. Castro and I. Requena, "Are artificial neural networks black boxes?" *IEEE Trans. on Neural Networks*, vol. 8, pp. 1156-1164, 1997.
[9] C. Moraga and K-H. Temme, "Functional equivalence between S-neural networks and fuzzy models", in: Technologies for constructing intelligent systems: tools. Heidelberg, Germany: Springer Physica-Verlag, 2002.
[10] 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*, vol. 12, pp. 277–306, 2001.
[11] W. Duch, R. Adamczak, and K. Grąbczewski, "Extraction of Logical Rules from Backpropagation Networks," *Neural Processing Letters*, vol. 7, pp. 1–9, 1998.
[12] J.R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufman, 1993.
[13] K. Grąbczewski and W. Duch, "A General Purpose Separability Criterion for Classification Systems," in Proc. 4th Conf. on Neural Networks and Their Applications, Zakopane, Poland, pp. 203–208, May 1999.

[14] W. Duch and G.H.F. Diercksen, "Feature Space Mapping as a Universal Adaptive System," *Computer Physics Communication*, vol. 87, pp. 341–371, 1995.
[15] W. Duch, R. Adamczak, and N. Jankowski, "New Developments in the Feature Space Mapping model," in Proc. 3rd Conf. on Neural Networks, Kule, Poland, Oct. pp. 65–70, 1997.
[16] W. Duch, N. Jankowski, K. Grąbczewski, A. Naud, and R. Adamczak, Ghostminer software, http://www.fqspl.com.pl/ghostminer/
[17] N. Jankowski, "Approximation and classification in medicine with IncNet neural networks." In: Machine Learning and Applications. Workshop on Machine Learning in Medical Applications, Chania, Greece, pp. 53-58, 1999.
[18] J.N. Butcher and S.V. Rouse, "Personality: Individual Differences and Clinical Assessment," *Annual Review of Psychology*, vol. 47, pp. 87–111, 1996.
[19] W. S. McCulloch and W. Pitts. "A logical calculus and the ideas immanent in nervous activity." Bulletin of Mathematical Biophysics, vol. 5, pp. 115–133, 1943.
[20] S. Haykin, *Neural networks: a comprehensive foundations*. MacMillian, 1994.
[21] W. Duch and N. Jankowski, "A Survey of Neural Transfer Functions," *Neural Computing Surveys*, vol. 2, pp. 163–213, 1999.
[22] Yager, R. R. and Kacprzyk, J., eds. The Ordered Weighted Averaging Operation: Theory, Methodology and Applications. Kluwer: Norwell, MA, 1997
[23] W. E. Combs, and J. E. Andrews, "Combinatorial Rule Explosion Eliminated by a Fuzzy Rule Configuration," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 1–11, 1998.
[24] M.K. Güven, K.M. Passino, "Avoiding Exponential Parameter Growth in Fuzzy Systems," *IEEE Trans. Fuzzy Syst.*, vol. 9, pp. 194–199, 2001.
[25] L.-X. Wang, "Analysis and design of hierarchical fuzzy systems", IEEE Trans. Fuzzy Systems, vol. 7, pp. 617624, 1999.
[26] M.-L. Lee, H.-Y. Chung, and F.-M. Yu, "Modeling of hierarchical fuzzy systems", Fuzzy Sets and Systems, vol. 138, pp. 343-361, 2003.
[27] T.R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring". Science, vol. 286, pp. 531-537, 1999.
[28] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoid function". IEEE Transactions on Information Theory, vol. 39, pp. 930–945, 1993.