Computer Physics Communications xx (1994) xxx
North-Holland

Computer Physics
Communications

Feature Space Mapping as a Universal Adaptive System

Włodzisław Duch
Department of Computer Methods, Nicholas Copernicus University, Grudziądzka 5, 87-100 Toruń, Poland

and

Geerd H.F. Diercksen
Max-Planck-Institut für Astrophysik, Karl-Schwarzschild Strasse 1, 85 740 Garching b. München, Germany

The most popular realizations of adaptive systems are based on the neural network type of algorithms, in particular feedforward multilayered perceptrons trained by backpropagation of error procedures. In this paper an alternative approach based on multidimensional separable localized functions centered at the data clusters is proposed. In comparison with the neural networks that use delocalized transfer functions this approach allows for full control of the basins of attractors of all stationary points. Slow learning procedures are replaced by the explicit construction of the landscape function followed by the optimization of adjustable parameters using gradient techniques or genetic algorithms. Retrieving information does not require searches in multidimensional subspaces but it is factorized into a series of one-dimensional searches. Feature Space Mapping is applicable to learning not only from facts but also from general laws and may be treated as a fuzzy expert system (neurofuzzy system). The number of nodes (fuzzy rules) is growing as the network creates new nodes for novel data but the search time is sublinear in the number of rules or data clusters stored. Such a system may work as a universal classificator, approximator and reasoning system. Examples of applications for the identification of spectra (classification), intelligent databases (association) and for the analysis of simple electrical circuits (expert system type) are given.

1. Introduction

In this paper a model of a universal adaptive system based on multidimensional localized functions is presented. This model facilitates the classification-approximation and employs a new way of knowledge representation by storing complex and fuzzy facts directly in the feature space. At present most popular classificators and approximators are based on models of adaptive systems of the Artificial Neural Network (ANN) [1-7] type, on statistical decision approaches such as Learning Vector Quantization (LVQ) and on self-organizing mappings (SOM) [8-10]. The popularity of these approaches, especially of neural networks, has been so great in the past decade that almost every parallel algorithm is called "neural". Most of these models have no resemblance to real neural models and therefore in this paper the term "adaptive systems" will be used more often than the term "neural models".

An adaptive system A_W is a system with internal adjustable parameters $\mathbf{W}$ performing vector mappings from the input space $\mathbf{X}$ to the output space $\mathbf{Y} = A_W(\mathbf{X})$. Neural networks are the most popular but not the only adaptive systems known. Some of the most successful adaptive systems (like LVQ), strictly speaking, are not of the neural network type, although they are usually classified as such. On the other hand expert systems based on

[1] duch@phys.uni.torun.pl
[2] ghd@mpa-garching.mpg.de

the production rules are usually perceived as systems for reasoning with knowledge stored in the variables bound to some linguistic labels. They rarely learn from examples requiring instead an introduction of new rules for each new fact. These two technologies are not as distinct as one may think. Neurofuzzy systems are combining features of adaptive systems such as learning from examples and meaningful generalization with the rule based fuzzy logic reasoning. Feature Space Mapping belongs to this family of the new generation hybrid systems.

Several factors have contributed to the popularity of artificial neural networks. First, there were hopes for commercial applications: neural networks should help to solve problems in artificial intelligence, pattern recognition, in vision and speech, challenges for which techniques based on logical analysis are not suitable. These hopes were at least partially justified as it is evident from the special issue of the "Communications of the ACM" devoted to artificial intelligence [11]. Second, the development of computer hardware and the availability of general programs for neural network simulations encouraged many scientists to try these new techniques for solving various problems in their particular fields. The majority of papers on neural networks belong to applications of well developed artificial neural network models and are not always written with a real understanding of what these adaptive systems are really capable of [12].

The design of artificial neural networks was motivated by the parallel processing capabilities of the real brain but the processing elements and the architectures used in artificial neural networks frequently have nothing to do with biological structures. Artificial neural networks are composed of simple processing elements (usually called "neurons") operating on their local data and communicating with other elements via links with adjustable strength called "weights". In most artificial neural networks these strength of connections between the elements are the only adjustable parameters $\mathbf{W}$ of the adaptive system and allow the network to realize a variety of functions. Some models introduce also adjustable internal parameters for each node. In fact, also "weightless" networks with fixed connections and adjustable internal parameters may be constructed. Feature Space Mapping may work in such a way for tasks requiring association or autoassociation. One of the main applications of artificial neural networks is to simulate associative memory [9]. In the typical case a set of input and output patterns is presented to the network and the weights are adjusted (this is called "learning" or "adaptation") until the outputs given by the network are identical to the required output.

The most common architecture of the artificial neural networks used at present is of the multilayered feedforward type, a simplification of the network composed from many perceptrons. In this architecture, called also the multilayered perceptron (MLP) network, nodes (neurons, processing elements) are arranged in layers. There are no connections within a layer and only elements belonging to adjacent layers are connected. Input signals are propagated in one direction (feedforward), from the input to the output layer, with each processing element being responsible for the integration of signals coming from the lower layer and affecting all processing elements of the next layer to which it is connected. The network is called "fully connected" if all possible connections between the consecutive layers are allowed. In some cases it is better to use an Artificial Neural Network that is not fully connected. The reduction of the number of adjustable weights (network complexity) may improve not only the timing of computation for training the network but also the accuracy of learning. Selection of the best architecture (number of layers, network parameters) is still more an art rather than science.

Perhaps the most important contribution that has made models of neural networks so popular was the backpropagation of errors training algorithm, described already in 1971 and rediscovered several times until it finally became widely known in 1986 [13]. Backpropagation of errors (BP) is still the most commonly used algorithm for supervised training of artificial neural networks (in a supervised training the answers for some training data set are known, in an unsupervised training the system tries to form some internal descriptions of classes in a self-organizing way). Backpropagation of errors is a universal learning rule that can be applied to a number of different architectures of adaptive systems. The term "backpropagation net" is commonly used to designate those networks that are trained using backpropagation of errors algorithm and the term is almost synonymous with the multilayered perceptron network. The backpropagation learning rule compares the sample output with the achieved output and the error signals (differences between desired and achieved outputs) are propagated layer by layer back to the input layer. The weights are changed, using some form of minimization (originally gradient descent) method, in such a way that the global error should be reduced after the next presentation of the same inputs/outputs. Although the backpropagation algorithm is rather slow and requires many iterations it enables learning of arbitrary associations and therefore is widely used. Over 40 other learning rules for different network architectures exist and new rules are still proposed. In particular many learning algorithms are based on using least squares minimization (via simulated annealing, genetic algorithms or standard minimization methods) of an error function. The correction of errors is performed in small steps, for each pair of input/output data, although provided all data are available, methods using a matrix formulation based on numerical procedures such as pseudoinverse matrices or singular value decomposition converge faster.

Artificial neural networks are especially suitable for problems where a high error rate is acceptable, the conditions are ill-defined and the problem mathematically ill posed. The brain was developed by the evolution to process the data from the senses and works much better at solving problems requiring perception and pattern recognition than problems involving logical steps and data manipulation. Most artificial neural network architectures share this quality with the real brains - it may be called the "intuitive approach". Such architectures are not suited for tasks that sequential computer programs can do well, such as manipulation of symbols, logical analysis or solving numerical problems (although many parallel algorithms for solving matrix equations are easily presented in the form of a network, making the hardware implementation feasible). The solution to many problems depends on the ability to use both intuitive and logical approaches and requires adaptive systems that are integrated with the logical and numerical computer programs. Neurofuzzy systems offer an attractive solution in such cases.

Artificial neural networks are adaptive systems with the power of a universal computer, i.e. they can realize arbitrary mappings (associations) of one vector space (inputs) to another vector space (outputs). Many problems in physics and chemistry require finding the global mappings from a set of known data points (training examples). Given a statistical sample of data points adaptive systems construct such global mappings. It is of great importance to understand what the neural networks can do and when their application may lead to new results hard to obtain with standard methods. Unfortunately, rarely relevant mathematical theories such as the theory of statistical decisions or approximation theory are invoked. As we have recently shown [12] in most physical and chemical problems direct fitting techniques should be more accurate than predictions of trained neural networks. Universal approximators, such as artificial neural networks, should be used only in cases when no approximate physical models exist and even then the accuracy of their results should always be compared with statistical methods for data modeling.

Artificial neural networks are of interest to physicists as an example of complex systems, more general than the Ising or the spin glass models [14]. From this point of view, as interesting dynamical systems, their evolution is investigated and methods of statistical physics applied to such problems as network capacity, chaotic behavior or network learning rules [15]. Artificial neural networks are also of interest as models of various sensory subsystems and as simplified models of the nervous system. As an example of the enthusiasm with which artificial neural networks are received by the scientific community it is noted that the number of papers in the section "neural networks" in *Physics Abstracts* has approximately doubled comparing the 1992 to the 1991 entries. Since a number of reviews appeared recently (including some in Computer Physics Communications [6]) we refer the reader to these reviews for general information on neural networks.

In the second section a short description of the most common adaptive systems, the feedforward and feedback artificial neural networks is given. The two main applications of adaptive systems, i.e. classification and approximation, are presented. The development of Feature Space Mapping is motivated by the analysis of the global mapping (the landscape of solutions) changes during training of such networks and the desire to construct this mapping explicitly. The solution is inspired by quantum chemistry where floating gaussian functions are used since many years as one-electron basis sets for molecular calculations, although the present model is a generalized version of the older model [16] that was based on floating gaussian functions. In the third section Feature Space Mapping is introduced and various aspects of the model are presented in details. Various forms of local transfer functions of processing elements are considered, biological motivation for localized functions is briefly described, connection with other approaches is outlined, algorithms for growing the network (increasing the number of processing elements) are described and local learning algorithms applicable to the Feature Space Mapping model are given. The last three subsections deal with the information retrieval, with Feature Space Mapping as fuzzy inference (expert) system and with the complexity and scaling of the network for large problems.

In the fourth section several examples of Feature Space Mapping applications are given. The associative capabilities of Feature Space Mapping are illustrated using some simple examples taken from the neural network literature. Classification capabilities are presented using an example of recognition of molecular spectra. In the next subsection Feature Space Mapping is used as a fuzzy inference system, deducing the qualitative behavior of electric circuits from the knowledge obtained from general laws, acquired directly or inferred from a number of specific examples used for training the system. Unsupervised learning for a large number of data items is discussed and time series forecasting is briefly mentioned. In the last section some properties of the Feature Space Mapping neurofuzzy system are briefly summarized.

2. Neural networks and other adaptive systems

Fully connected networks are the most general type of architecture and lead to all other architectures as special cases when appropriate restrictions for the connections are imposed. Hopfield [17] has analyzed such fully connected feedback networks with symmetric weights from the statistical mechanics point of view and proved that the dynamics of such networks has stationary points. In the original Hopfield model activations of neurons were binary but the extension to continuous activations was straightforward. An energy function E(**W**,**a**) may be defined as a weighted sum of activations **a** of neurons and the minima of this function are identified with the stored pattern X_i. Given a set of patterns **X**, weights **W** of the network can be set up in such a way that the energy minima will correspond to these patterns.

The Hopfield network is capable of autoassociation. Given a pattern or a part of it as the initial activation values of neurons the network will evolve to a stable state by activating other neurons corresponding to the missing parts of the pattern. Autoassociation is more general than association of input **X** with output **Y** because it allows to recover the whole pattern from any part of the (**X**,**Y**) vector. Storing many patterns in the Hopfield networks may lead to the appearance of spurious states and to the destabilization of the desired states. This happens because the energy landscape E(**W**,**a**) is to a large degree arbitrary and there is no control over the basins of attractors of the stationary states, their size, depth and relative placement. These problems are present as well in the more complex models of feedback networks, such as Boltzman machines [2,18].

In practice simpler solutions may work better. One way to reduce the number of connections and to simplify the network computation is to arrange processing elements in layers and to allow connections only between the layers, but not within a layer. The input signals are than propagated to the output layer in an unidirectional feedforward way. In particular, one or more hidden layers are present in the network except for the input and the output layer. The lack of feedback implies also the lack of dynamics. Feedforward networks are adaptive systems directly mapping the input to the output vectors. Such networks are trained to guess the mapping hypersurfaces from the sample points presented to the network (examples, training data). This amounts to using sigmoidal functions to fit the data. It is questionable whether this form of function realized by the network and by the adaptation procedure employed to determine it is indeed the most optimal. Often it is not the best solution to the problem of reconstructing hypersurfaces from sample data since some hypersurfaces are not well represented by sigmoidal functions:

$$\sigma(x)=(1+e^{-x/T})^{-1} \tag{2.1}$$

The explicit form of the mapping realized by typical feedforward neural networks is:

$$F_W(X)=\sigma(\Sigma_{i_1} W_{i_1}^{(1)}\ \sigma(\Sigma_{i_2} W_{i_2}^{(2)}\ \sigma(...(\Sigma_{i_k} W_{i_k}^{(k)}\ x_{i_k}\ )...))) \tag{2.2}$$

A geometrical interpretation, similarly as for recurrent feedback networks, is also possible for feedforward networks. There is no energy function and the training procedure is based on a gradient descent on the error surface. The error is defined as the sum of squares of the differences of the sample outputs **Y** and the achieved outputs $\mathbf{O}=\mathbf{F}_W(\mathbf{X})$:

$$E(W)=\frac{1}{2}\sum_p\sum_i\left(Y_i^p-\mathbf{F}_W\left(X_i^{(p)}\right)\right)^2 \tag{2.3}$$

where the sum runs over all training patterns p and vector components i. The error depends on the weights **W** that are adjusted until minimal error (in some cases zero) is reached. Backpropagation of errors, the most frequently used learning algorithm, is based on the gradient minimization of this error function. In the error formula given above the explicit form of the network function (2.2) should be inserted and the gradient over all weights computed. The formula for gradients differs depending on the layer to which the weights are connected, leading to the standard backpropagation of errors equations. Most of the improvements in the multilayered perceptron learning algorithms arises from the application of various minimization methods to the error function.

At the start of the training the initial mapping from the input space to the output space is almost random (the weights are set up randomly and constraints depend on the topology of the network). During learning the

Name	Decision surface	Discriminant function $g(X)$
Linear, threshold logic, minimum distance	hyperplanes	$\sum_i W_i x_i + W_{N+1}$
Piecewise linear	hyperplane fragments	maximum over a set of linear discriminant functions
Nearest Neighbor Classifier (NNC), k-Nearest Neighbor Classifier (k-NNC)	hyperplane fragments	most common class within the distance k from X
Quadratic	second-order functions	$\sum_{i \geq j} W_{ij} x_i x_j + \sum_j W_j x_j + W_{N+1}$
Gaussian, with optimized centers and dispersions	hyperellipsoids, centered at **C**, with dispersions σ	$\sum_i W_i \exp\left[-\sum_j \frac{(x_j - c_{ij})^2}{\sigma_{ij}}\right]$
Potential function	combination of 1/r factors	$\sum_i W_i / \sqrt{\sum_j (x_j - c_{ij})^2}$
Compound Classifier, Restricted Coulomb Energy (RCE)	arbitrary, constructed from hyperspheres	$const$ for $\lVert X - C_i \rVert \leq \lambda_i$ 0 for $\lVert X - C_i \rVert \geq \lambda_i$
Basis set functions (Φ functions)	arbitrary	$\sum_i W_i \Phi_i(x_1, x_2 ... x_N)$

Table 1. Summary of the main methods of pattern classification

output landscape of the feedforward network as a function of all input parameters is changing to accommodate the associations forced on the neural network by the training data. Since the starting point is random it cannot be expected that the generalization (interpolation) for the unknown input will be in any sense optimal. A similar situation is found in the recurrent feedback networks. The basins of attractors around the desired associations are formed but the size and the depth of the attractor basins is by no means controlled and the landscape of the whole network function is very complicated and contains many local minima corresponding to false associations.

Since the number of weights may be rather large the problem of avoiding the local minima in multidimensional space is really severe. Although many improvements of the original algorithm have been proposed finding the global minimum requires a rather complex searching procedure. In fact it is an NP-hard problem [19] and neural networks are giving interesting results only because some hard problems are solved in polynomial time when randomized computations that do not guarantee the best possible solution are used. In the off-line learning case, i.e. when all data is available to the system, direct fitting in sigmoidal functions may give more accurate results than the best network function that may be found after a long training. In fact fitting using sigmoidal functions is sometimes more accurate than expected. In the on-line learning case the data is continuously incoming and the fitting methods cannot be used.

2.1 Networks for classification

Another way of looking at adaptive systems is from the point of view of their capabilities for classification. **Pattern classification** is a branch of statistics developed especially for vision-related problems [20-23]. Many of the algorithms used in pattern analysis, such as the nearest neighbor algorithms, are strong competitors of neural network algorithms. The simplest classifier is based on linear units and computes hyperplanes as decision surfaces. More flexible decision surfaces are obtained if the input is first transformed by using nonlinear functions (for example by taking polynomial terms) and then presented to the linear classifier. This results in decision regions with arbitrary shapes and linear combination coefficients. Even more flexible are potential function classifiers, starting with the simplest gaussian model of Sebestyen [21] and the electrical charges in multidimensional space [22] model and progressing to the compound classifier of Batchelor [23] and the very similar Restricted Coulomb Energy (RCE) classifier [24] using many hyperspheres filling the decision regions. Fuzzy clustering [25] is equivalent to the gaussian model with optimized positions and dispersions. All these pattern classification techniques may be presented in a form of a network. A summary of the main methods of pattern classification is given in Table 1.

A very interesting and universal system for classification, called SONET, has been recently described by Nigrin [26]. This system, described as an autonomous pattern classification system, is an example of the new generation of universal classification systems. It should be able to:

1) self-organize using unsupervised learning.
2) form stable category codes.
3) operate under the presence of noise.
4) operate in real-time.
5) perform fast and slow learning.
6) scale well to large problems.
7) use feedback expectancies to bias classifications.
8) create arbitrarily coarse or tight classifications that are distortion insensitive.
9) perform context-sensitive recognition.
10) process multiple patterns simultaneously.
11) combine existing representations to create categories for novel patterns.
12) perform synonym processing.
13) unlearn or modify categories when necessary.

Feature Space Mapping is also a universal classification system. It fulfills the conditions given above as well as many others, related to a representation of knowledge and making inferences.

2.2 Networks for approximations

Many of the tasks performed by neural networks may be looked upon from the point of view of approximation theory. Various types of adaptive systems suitable for function approximation have been proposed. It has been proven that neural networks using sigmoidal functions are universal approximators [27]. General radial basis functions are also suitable as the basis for universal approximation [28], and even more general kernel basis functions can be used for uniform approximation by neural networks [29]. In fact, many other types of functions may be used, for example rational functions [30] lead for some approximation problems to networks of lower complexity (smaller number of parameters) than the multilayered perceptrons or the Radial Basis Functions (RBF) [31] type of networks.

Approximation theory is a very large field and only a few of the approaches most relevant in the context of neural networks will be mentioned here. The biggest drawback of multilayered perceptrons based on sigmoidal or step functions is the speed of learning: difficult global minimization problem has to be solved for this case and finding a solution to such problems is NP-hard [19]. The speed of learning of localized approaches is much higher since only local optimization problems are solved, similarly as in the well known “divide and conquer” strategy. Approximating noisy data using radial basis function networks gives good results [32] without deterioration of the speed of training. Ripley has compared standard multilayered perceptron neural networks with such nonlinear regressive methods used in statistics as MARS, CART or BRUNO [32] and has concluded that multilayered perceptron networks only rarely have advantages over nonlinear regression. Other methods that are presented in network form, having their origin in statistics, are probabilistic (Bayesian) neural networks (cf. Wasserman [4]) developed recently by Specht from probabilistic classification systems [33] in a “General Regression Neural Networks” [34].

Approximation of time series is of particular importance because of the industrial and financial applications, where “advanced technology” applications are not only neural networks, but also genetic algorithms, fuzzy logic, statistical methods (nonlinear, time-series, optimization, etc.), complexity theory, artificial life (a-life) theories (cellular automata simulations), nonlinear dynamical theory and chaos theory. The first bimonthly journal on the subject, called NEUROVE$T JOURNAL, was recently started, books are being written [35] and a number of interesting articles in physics journals have appeared [36].

Extrapolation of the results is performed by training the networks in an autoregressive way. Many networks designed for approximation are therefore tested on such problems as prediction of the Mackey-Glass chaotic differential delay equation. A comparison of different methods applied to this equation is described by Sanger [37].

3. Feature Space Mapping Model

Problems with multiple minima and the slow speed of learning of backpropagation multilayered perceptrons (MLPs) may be avoided by constructing the network function in an explicit way. The simplest functions with suitable properties are of the gaussian type. Although some other functions, like a product of the two sigmoidal functions σ(x)(1-σ(x)), or the sigmoidal functions $\sigma(-||X-D||^2)$ are very similar to the gaussian functions, the great advantage of the gaussian functions is their factorability:

$$G(\mathbf{X},\mathbf{D},\sigma) = \exp\left(-\sum_{i=1}^{N}(X_i - D_i)^2/\sigma_i\right) = \prod_{i=1}^{N}\exp\left(-(X_i - D_i)^2/\sigma_i^2\right) = \prod_{i=1}^{N} g(X_i, D_i, \sigma_i) \qquad (3.1)$$

Although the Feature Space Mapping model developed below may work with other functions the factorization property is very useful for the reduction of multidimensional searches to a series of one-dimensional searches. Gaussian functions should be centered on the data vectors $\mathbf{D}=(D_1, D_2, \ldots D_N)$ with dispersions proportional to the error or uncertainty of the variables D_i. Each variable defines a new dimension. The data vector defines a point and the data vector together with the associated uncertainties defines an ellipsoid in N-dimensional space (i.e. the contours of the constant value of this function are ellipsoids), described by the density of the $G(\mathbf{X},\mathbf{D},\sigma)$ function. From another point of view gaussian functions are a fuzzy representation of the data points. Generalization to asymmetric gaussian functions is straightforward:

$$\begin{aligned} G(\mathbf{X},\mathbf{D},\sigma) &= \prod_{i=1}^{N} g(X_i, D_i, \sigma_{i+}, \sigma_{i-}) = \qquad (3.2) \\ &\prod_{i=1}^{N}\left[(1-\theta(X_i - D_i))e^{-(X_i-D_i)^2/\sigma_{i-}^2} + \theta(X_i - D_i)e^{-(X_i-D_i)^2/\sigma_{i+}^2}\right] \end{aligned}$$

where θ(x) is a step-function (0 for x<0 and 1 for x>0). These asymmetric gaussian functions give greater flexibility in modeling various density distributions but the fact that their derivatives are discontinuous may become a problem in some error minimization procedures.

The term “fact” will be used for a collection of input and output values that are to be stored in the Feature Space Mapping adaptive system. Facts belong to the feature (conceptual) space which has many dimensions but is finite. In case of human knowledge the number of concepts, or elements of reality that are distinguished by humans, is perhaps of the order of 10^5 -10^6 (including sensory qualities, words and abstract concepts). Combinations of these elements of reality, including sensory elements, create facts. The *FSM* function for a collection of facts $\mathbf{D}=\{D^p\}$ has the following general form:

$$FSM(\mathbf{X},\mathbf{D},\sigma) = \sum_p W_p G(\mathbf{X},\mathbf{D}^p,\sigma^p) = \sum_p W_p \prod_i e^{-(X_i - D_i^p)^2/\sigma_i^p} \qquad (3.3)$$

It is different from zero only in the neighborhood of the data vectors **D** that are parameters of this function. The weights **W** and the dispersions **σ** are the adaptive parameters defining Feature Space Mapping for a given set of **D** input values. In some applications when the number of facts is large and data compression is required or when data values are noisy also the gaussian centers **D** and their widths should be treated as adaptive parameters.

The values of the *FSM*(**X**,**D**,σ) function may be in most points **X** of the parameter space very small, depending on the separation of the data points and the dispersions of the gaussians. This means that no data similar to **X** is known to the Feature Space Mapping system. For large dispersions the values of the *FSM* function at most points of the feature space will be non-zero and gradients of the function will help to find the closest matching facts.

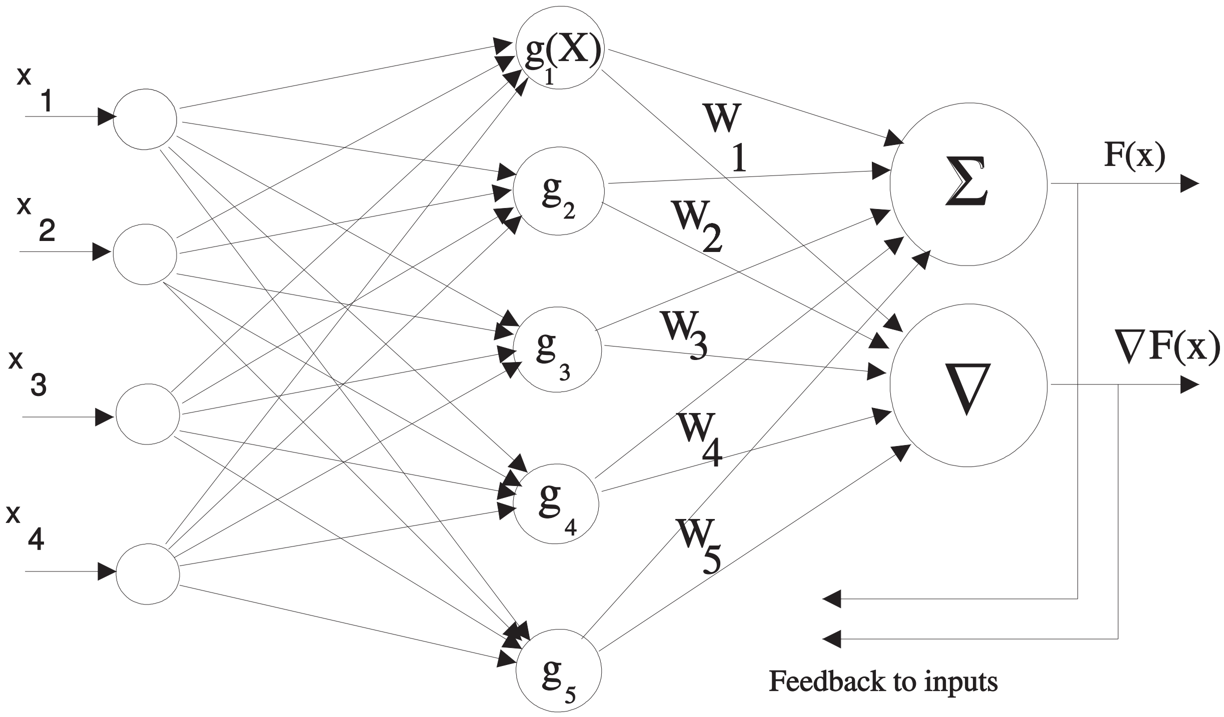

Fig. 1. Example of a network realizing the Feature Space Mapping (Feature Space Mapping)

The model allows for many choices and extensions. For example, more internal parameters may be defined, like rotation axis of the gaussians (this will destroy the separability of functions). However, even in its simplest version the Feature Space Mapping model is interesting: the incoming signals are not weighted, parameters W_p may be taken as binary (though for some classification purposes their values may be useful) and true facts are equivalent to nonzero $FSM(\mathbf{X},\mathbf{D},\sigma)$ values. There is no problem with storing negative knowledge, i.e. facts that must not be true because one of the axis may be labeled as true/false and gaussians position on this axis will correspond to the true/false facts. Binary signals lead to simplifications in the network realization, but in this paper such special cases will not be discussed.

3.1 Network representation and the neuron output functions

It is possible, although not necessary, to present Feature Space Mapping in form of a network (Fig. 1) in which the number of processing elements is growing with the number of remembered facts. The processing elements of such networks are not of the threshold type but rather define a "window" for the incoming data. It is important to notice the difference between processing elements used in the Feature Space Mapping and in neural networks of the multilayered perceptron type. In the latter case these elements have only two internal parameters (controlling nonlinearity and threshold), usually fixed for all network elements. Such neurons use external parameters (incoming weights) to determine the angle of rotation of the sigmoidal hypersurface and their output is determined by a single variable, the total activation. In Feature Space Mapping each processing element is multidimensional and the output may also be multidimensional, or there may be no output at all (cf. Fig. 1 and Fig. 9). The information processing capability of these elements is contained in the internal parameters rather than in the external weights.

It may be argued that the biological neurons are closer to such processing elements rather than to the threshold devices. It is not of interest here to model neural systems at the lowest level of single neurons, but rather at the functional level of groups of neurons specializing in feature detection. Real neurons react and send series of impulses (spikes) rather than a single impulse. The spatial and temporal summation of signals in real neurons depends on the phases and frequencies of incoming signals, giving a total input of the form:

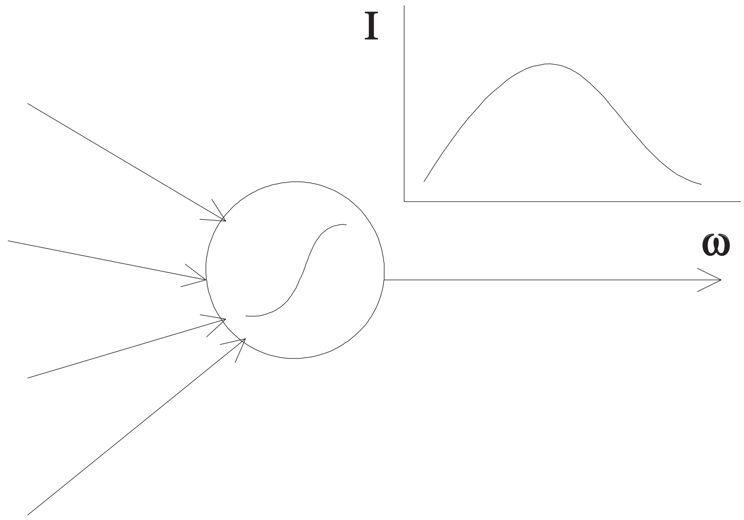

Fig. 2. Sigmoidal neurons work as filtering devices, transmitting signals selectively

$$I(t) = \sum_k a_k \sin(\omega_k t + \theta_k)\, e^{-t/\sigma_k} \tag{3.4}$$

This signal is integrated over time and is very sensitive to the changes in relative phases and frequencies. Fixing several frequencies in the 10-100 Hz range (typical spiking frequencies for biological neurons) and varying the frequency of only one incoming signal, a resonance behavior is observed, indicating that a specific input is necessary rather than just a strong one. Thus in frequency dependent networks sigmoidal functions help to recognize certain pattern in the incoming signals (Fig. 2).

Frequency dependent processes are hard to model in networks composed of many processing elements and almost always amplitudes are used instead of frequencies. In such a case sigmoidal transfer functions should not be used because they are not recognizing patterns in signals. Therefore the processing elements in time-independent models of neural networks should filter the data rather than act as simple threshold devices. Only in more biologically oriented networks, where synchronization of spikes and pulse modulation is important, sigmoidal functions should be used. For most cognitive tasks a time averaged formulation may be sufficient. The locality of neuron response may be due either to the network architecture itself or to the biophysical properties of neurons. A large percentage of the neurocortex neurons are involved in the local circuits (cell assemblies) working as feature detectors. For example, neurons in the visual cortex may respond selectively to the position and the angle of an object as a consequence of the structure of connections in the brain. Other neural cells, like cochlear stereocilia cells, also react in a localized way to sensory input data. For more evidence on the biological plausibility of product units see [38].

The transfer function of the processing elements of the network should allow for the representation of complex shapes using a small number of parameters. Such neuron functions in the N-dimensional case may be approximated by the following functions:

1. Gaussian functions with different dispersions for different dimensions. These functions have $2N$ parameters (centers and dispersions) defining ellipsoidal densities in the feature space. Angles of rotation may be introduced for maximum flexibility, leading to a total of $3N$-1 parameters. Except for the simple gaussian of Eq. (3.1) the following function may also be used:

$$G_s(\mathbf{X};\mathbf{D},\sigma) = \frac{1}{1+\exp\left(\sum_{i=1}^{N}(X_i-D_i)^2/\sigma_i^2\right)} = \frac{1}{1+\exp\left(G_g(\mathbf{X};\mathbf{D},\sigma)\right)} \tag{3.5}$$

In practice, simplifications of this function like

$$h(X,D,\sigma) = 1/\left(1+(X-D)^2/\sigma^2\right); \quad H(\mathbf{X},\mathbf{D},\sigma) = \prod_{i=1}^{N} h(X_i,D_i,\sigma_i) \tag{3.6}$$

or

$$h_1(X,D,\sigma)=\begin{cases}\left[1-(X-D)^2/\sigma^2\right]^2; \text{ if } (X-D)<\sigma \\ 0 \quad \text{otherwise}\end{cases} \tag{3.7}$$

are faster to compute, give a good approximation to the gaussian functions and offer the same adaptive parameters as well as separability.

2. Combinations of N one-dimensional gaussian functions (called "gaussian bar functions" [39]) instead of the products of one-dimensional gaussians:

$$G_b(\mathbf{X};\mathbf{D},\sigma,\mathbf{W})=\sum_{i=1}^{N} W_i \exp\left(-(X_i-D_i)^2/\sigma_i^2\right) \tag{3.8}$$

These functions have $3N$ adjustable parameters. Such an approach makes the elimination of irrelevant input variables, i.e. dimensionality reduction, easier than in the case of multidimensional gaussians (although, if dispersions are adjustable and rotations are allowed, some dimensions should reduce to zero). These functions have the same disadvantages as the sigmoidal bar functions described below.

3. Products of pairs of sigmoidal functions rather than a single sigmoidal function. This is the most flexible approximation giving decision regions of arbitrary shape for classification. The product of $2N$ sigmoidal functions has the following general form:

$$s(\mathbf{X};\mathbf{D},\Delta)=\prod_{i=1}^{N}\sigma(X_i-D_i)(1-\sigma(X_i-\Delta_i)) \tag{3.9}$$

For increasing input X_i the value of the first sigmoidal function in the product grows while the value of the second function decreases to zero, forming a localized function. Shape adaptation of the density s($\mathbf{X}$;$\mathbf{D}$,Δ) is possible by shifting $\mathbf{D}$ and Δ around some central value and by rotating the input variables $\mathbf{X}$ by the incoming weight vector $\mathbf{W}$. The number of parameters in this case (not counting the weights W) is again $2N$ but additional parameters controlling nonlinearities of each sigmoidal function may be added for maximum flexibility. This increases the number of parameters to a total of $4N$. Classification regions may have sharply defined boundaries or may be smooth and similar to hyperellipsoids. Dimensionality reduction is possible as in the case of "gaussian bar" functions. These functions are again separable and are well suited for use in the Feature Space Mapping system. Since they have two centers they may be called "biradial".

4. Weighted combinations of one-dimensional sigmoidal functions:

$$s_b(\mathbf{X};\mathbf{D},\Delta)=\sum_{i=1}^{N} W_i\ \sigma(X_i-D_i)(1-\sigma(X_i-\Delta_i)) \tag{3.10}$$

called "sigmoidal bar functions". The description of a single fact (data point) represented by a high density around a point $\mathbf{D}$ in N dimensions requires just one gaussian function. To create a high density area in a one-dimensional case two sigmoidal functions are combined together, $\sigma(X)-\sigma(X-D)$. In the case of N dimensions a combination of $2N$ sigmoidal functions is required. Although the representation of data clustered around a few points using bar functions may seem rather inefficient for large number of data points, gaussian bar or sigmoidal bar representations may have some advantage because a smaller number of adaptive parameters is necessary to represent the data. If the data points are placed regularly every k units in a square mesh the number of gaussians necessary to model such data is proportional to the number of clusters k^2 while the number of bar functions needed is only $2\times 2k=4k$. In addition to the desired density, bar functions create N waves (bars) in N-1 dimensional subspaces. This is a minor disadvantage since with some care the unwanted topographical features can be filtered out with the help of a threshold output function.

There are of course other options for the output functions. Separable localized transfer functions are preferable for high efficiency of Feature Space Mapping networks. Even on massively parallel computers localized interactions among processing elements of neural networks can make a big difference (cf. example extremely high speeds of computation in Cellular Neural Networks [40]).

3.2 Connection with other approaches

Feature Space Mapping using gaussian functions may be treated as a special case of regularization networks, more precisely of the radial basis function networks [40] that have been employed recently for classification problems as alternative to the multilayered perceptron neural networks. The idea of using radial basis functions is due to the theory of function approximation and regularization. Given a set of (X_i, Y_i) pairs, a function of the form

$$F_W(X) = \sum_{i=1}^{K} W_i\, h(|X - D^{(i)}|), \tag{3.11}$$

where W_i and $D^{(i)}$ are parameters and the function h, called "the radial basis functions", depends only on the distance between the **X** and **D** vectors. The approximating function should minimize the error function

$$E[F_W] = \sum_{i=1}^{N} (Y_i - F_W(X_i))^2 + \lambda ||PF_W||^2 \tag{3.12}$$

which, except for the usual least squares error measure, contains a stabilizer (regularization) operator P, whose relative importance is scaled by the λ parameter. This allows to take into account additional conditions, such as the smoothness of the approximating function. In particular, for noisy data this parameter with appropriate stabilizer (squares of the second derivatives for example) may filter the noise quite effectively. The minimization problem for the error function may be solved in an elegant way [42]. Many types of radial functions have been considered like surface splines, multiquadratics, shifted surface splines and other functions [41].

Gaussian functions belong also to the radial basis function family. Feature Space Mapping with gaussians should be considered as a special case of the radial basis function approach, although there are many differences in comparison to the radial basis function networks designed for approximation, as described by Poggio and Girosi [41]. The general radial basis function theory neither the more general regularization network theory is not used in this paper since it is not necessary for the type of problems considered here. Moreover, gaussian bars, asymmetric gaussians and products of sigmoidal functions are not radial functions. However, it should be remembered that gaussian functions, as well as sigmoidal functions, may be used as basis for the construction of the approximating functions for an arbitrary mapping and are therefore sufficient although not always the most convenient [28,29].

In contrast to the original radial basis function and similar approaches [41] Feature Space Mapping is build from examples, presented in the teaching or network training phase, by growing the network and by using local learning algorithms. For new facts new nodes $G(\mathbf{X}, \mathbf{D})$ centered around distinct data clusters are added. Fact **X** similar to the existing facts are accommodated by modifying the existing facts closest to them in the feature space. This algorithm allows to avoid rapid growth of the number of network nodes with increasing number of data facts.

There are several possibilities to develop the Feature Space Mapping network: adding new node centers, removing isolated node centers, changing internal parameters of the nodes to move positions and change width of the gaussian functions realized by these nodes. For example, resolution of the gaussian function $G(\mathbf{X}, \mathbf{D})$ in the direction k may be defined as roughly the distance $0.83\sigma(D_k) \approx \sqrt{\ln 2}\, \sigma(D_k)$ from its center. If a new data point is not further from the existing fact **D** in the feature space there are two choices. The origin **D** of the existing gaussian function may be shifted to account for the new fact in a better way, or asymmetric dispersions may be defined to assign this region of the feature space to the existing fact **D**:

$$\sigma(D_k) \leftarrow \sigma(D_k) + |D_k - X_k| / \sqrt{\ln 2} \tag{3.13}$$

where σ is σ_+ or σ_-, depending on the position of the new data point X_k in relation to the center of the gaussian D_k, leading to asymmetric gaussian function. These functions represent distorted ellipsoids and allow for a much greater flexibility in the modeling of facts in feature space. Combinations of products of sigmoidal functions are even more flexible and allow for modeling densities of arbitrary shapes. In the Adaptive Resonance Theory (ART) of Carpenter and Grossberg [1] only one parameter (called "vigilance") controls the importance of all new facts in relation to what the system already knows while in the Feature Space Mapping model selective attention to individual facts is possible.

3.3 Growing the Feature Space Mapping network

Among the many types of adaptive systems those that are capable of growing are the most interesting ones. Learning based on modifying the network structure gives greater capabilities than learning based on the modification of parameters. The idea of incremental learning by a network that grows with the new facts, while trying to minimize the network complexity at the same time, is not new. These "constructive models" allocate new computational resources whenever new or unusual data appears and learn rapidly forming compact representation of the data. Among the models based on the local representation of data the Restricted Coulomb Energy [22,24] and Grow And Learn (GAL) model [43] should be mentioned. A Resource Allocating Network (RAN) introduced by Platt [44] is among the most popular networks of such type. It is essentially the growing radial basis function network, with output obtained as a linear combination of values from a single hidden layer of locally tuned units. The Function Estimation Approach [45] introduced recently is similar to the RAN and aims at finding criteria to limit growth of the number of radial basis functions used by the network. The challenge of finding the best approximation (or classification) function may be regarded from the point of view of the input data space (feature space), the parameter space or the infinite space of functions (Hilbert space). Recently, Fritzke developed a very interesting "growing cell structures" model based on the radial basis functions [46].

Some models also try to optimize the network structure by computing the importance of the processing nodes. Many constructive network models are based on distributed representations. The "cascade-correlation" algorithm [47] is another well-known model of such kind and uses sigmoidal units in a multilayered perceptron. Several other models of this sort, as well as models that start from large network structures and remove unimportant nodes and connections or modify the error function to generate simpler network structures, are discussed by Alpaydin [43].

The problem of learning on-line, with the new data patterns constantly presented to the system, may be stated in the following way: given the approximating function $F^{(n-1)}$ realized by the adaptive system and the new data $(\mathbf{X}_n,Y_n)$, find the best new estimate $F^{(n)}$. In the context of the Feature Space Mapping system this may require a new unit:

$$F_W^{(n)}(\mathbf{X};\mathbf{D},\sigma)=\sum_{k=1}^{K-1} W_k G(\mathbf{X};\mathbf{D}_k,\sigma_k)+\left(Y_n-F_W^{(n-1)}(\mathbf{X}_n;\mathbf{D},\sigma)\right)G(\mathbf{X};\mathbf{X}_n,\sigma_n)=\sum_{k=1}^{K} W_k G(\mathbf{X};\mathbf{D},\sigma) \quad (3.14)$$

The dispersion of this new unit should be optimized to minimize the local error in the neighborhood of the $\mathbf{X}_n$. The weight of this unit is equal to the error of the existing network with K-1 units at the new data point $\mathbf{X}_n$. If this weight is smaller than a given threshold the unit is not added. A second criterion is also useful: the new unit should not be too close to the existing units, i.e.

$$\min_k \|\mathbf{X}_n-\mathbf{D}_k\|>d_{\min} \quad ; \quad Y_n-F_W^{(n-1)}(\mathbf{X}_n;\mathbf{D},\sigma)>\varepsilon \quad (3.15)$$

Here d_{min} is the resolution of the data in the input space. The value for the dispersion σ_k is frequently based on the nearest neighbor heuristic. When the new data does not satisfy both criteria given above, gradient adaptation of the weights and gaussian centers is performed. Only the local gradient estimation is used here for the $(\mathbf{X}_n,Y_n)$ data (as is also done in RAN and in the function estimation approach [44,45]):

$$\mathbf{W}\leftarrow\mathbf{W}+\eta\left(Y_n-F_W^{(n-1)}(\mathbf{X}_n;\mathbf{D},\sigma)\right)\nabla_{W,D}F_W^{(n-1)}(\mathbf{X}_n;\mathbf{D},\sigma) \quad (3.16)$$

where η is the adaptation step size. The dispersions of the node functions should be rather large to obtain a smooth approximating function and avoid overfitting of noisy data. An improvement over the gradient procedure is obtained by using the extended Kalman filter algorithm [45], although it is not clear how much the speed and the accuracy of learning is improved due to this more complex algorithm and how much it is improved because dispersions are added to the list of adapted parameters.

Critique of artificial intelligence is frequently based on the theory of formal automata and the incompleteness theorems for formal systems, descending from the Gödel theorem (for example see the book of Penrose [48]). Although such a critique was not treated seriously already by Alan Turing (cf. his article "Computing machinery and intelligence" [49]) it is interesting to note that networks that can modify their structure are not restricted by such theorems. Therefore their computing power should be greater than those of formal computing devices, such as Turing machines. Although we do not know about any precise mathematical analysis of this interesting situation there are indications that problems which are NP-complete and cannot be solved using conventional algorithms can be solved in polynomial time by networks modifying their structure [19]. The

Feature Space Mapping model integrates some of the ideas from the papers quoted above, giving a universal adaptive system that can be used for learning from examples, can store knowledge in form of laws and can serve as a fuzzy expert system.

Another idea which is worth exploring is based on the following observation: the system trying to form an accurate approximation to the incoming data in form of (X,Y) pairs should not only focus on the regions of feature space where the density of the incoming points X is high but also pay attention to the areas where the values Y change most rapidly. This may be done by initially creating more nodes than is needed and trying to merge some of these nodes later, selecting for merging those nodes that lead to the smallest increase of the error. Initial data distribution is approximated using some simple tessellation scheme (for example balltree tessellation). In the "melting octree network" [50] an interesting method of moving the centers towards the Bayes class boundary is given. The positions of the node processing functions in the feature space may be regarded as the position of pseudoparticles that are moving, using gradient descent, towards local minima placed on the Bayes (or rather estimated Bayes) class decision borders. The *a priori* probability required by the probabilistic approaches of the data being in class k in the volume of the feature space covered by the $G(X,D_i,\sigma_i)$ function is estimated as:

$$P_k(D_i,\sigma_i) = Class_k(D_i,\sigma_i)/N_d(D_i,\sigma_i) \tag{3.17}$$

i.e. in the volume assigned to the function $G(X,D_i,\sigma_i)$ the number of data points belonging to the class k is counted and divided by the total number N_d of all data points in this area. This is an example of the local learning algorithm very useful in the supervised learning case.

3.4 Local learning algorithms

Training algorithms of a typical artificial neural network require presentation of all data many times. Each series of complete data presentation is called an epoch and in some cases tens of thousands of epochs are required to adjust learning parameters (weights) of a feedforward multilayered perceptron network using backpropagation of errors training procedure. Why? Suppose that a high accuracy solution to a numerical problem is sought using finite element method. To assure the required accuracy the density of integration points should be sufficiently high in the regions where the change of potential is rapid. In the other, "flat potential" regions less point are sufficient for the same accuracy. An equivalent of the adaptive, multiscale integration methods in neural learning should similarly focus on the regions of the input space where the density of data is large and increase the capacity of the learning systems in these regions. Excellent results obtained recently using local learning algorithms for character recognition problems [51] support this reasoning. The theory of local risk minimization [52] gives solid mathematical foundation to the development of local learning algorithms.

Adaptive systems that use the local training algorithms are trained on a few input patterns that are close to the pattern selected. One of the best known methods in pattern recognition, the k-NN or k-Nearest-Neighbor method, works in this way. However, selection of neighbors is a slow process (later in this paper the concept of a bumptree will be used to speed this selection in Feature Space Mapping networks). Although it is possible to reformulate the learning procedure of the artificial neural networks based on nonlocal functions (such as sigmoidal functions) by introducing modular networks in the form of "adaptive mixtures of local experts" [53] the learning procedure is more natural if locally-tuned processing units are used.

From a formal point of view a neighborhood function K (for example a gaussian or a square function) centered around the N test data points C_i is introduced and an error function becomes:

$$E[F_W] = \sum_{i=1}^{N} \sum_{j\in O(Ci)} K_i(X_j - C_i)(Y_j - F_W(X_j))^2 \tag{3.18}$$

This allows for separate minimization in each neighborhood of the training data. If the neighborhood function K is adjusted in such a way that it contains k nearest patterns and the parameters W are constrained to select the most frequent class among these patterns kNN algorithm is obtained. Parzen window method [20] is obtained by taking gaussian neighborhood function. Networks based on Parzen estimators were proposed recently [34]. They are related to the probabilistic neural networks [33] and in practice (although the theoretical justification is quite different) lead to the same learning algorithm as the radial basis function networks [41] and to the algorithm of Moody and Darken [42]. In practice simplest solutions for unsupervised data clustering work quite well: if a large amount of data is presented to the system the nearest node functions centered at **D** are moved towards the incoming data item **X** by a small amount:

$$\mathbf{D} \leftarrow \mathbf{D} + \eta(\mathbf{X} - \mathbf{D}) \tag{3.19}$$

This solution leads to self-organization of data clusters reflecting the probability distribution of the incoming data [9]. Another solution to the minimization of complexity of the network is to add to the error function a term that penalizes the number of non-zero weights in a fixed structure network. Quite sophisticated methods of this kind were proposed [54] aiming at dynamical reduction of the number of clusters necessary for a description of the incoming data to given level of accuracy. These new formulations are based on the maximum entropy principle and statistical physics analogies and should be also very useful in optimization of the Feature Space Mapping model.

A new formulation of the local learning is based on the tree-structured algorithm [37] and is particularly relevant in the context of the Feature Space Mapping model since it requires separable basis functions, such as gaussian functions. The basic idea of this method, related to such statistical approaches as the k-d trees [55], CART [56] and MARS [57] methods is to train the network separately in each dimension of the input data to avoid "the curse of dimensionality", i.e. the problem of combinatorial growth. With the growing dimensionality of the input data the number of data points necessary for reliable approximation of this data grows exponentially. Another way to avoid this problem is based on the gaussian bar functions [39].

A very interesting approach to clustering, based on the information theory and statistical mechanics, has been recently described [58]. Heating up the system the dynamics of the melting of clusters may be followed, from each data point treated as a separate cluster at low temperature to all data treated as one cluster at high temperature. This "melting" process is a reversal of the annealing procedure. The melting algorithm finds the optimal number of clusters by exponentially fast iterative procedure.

3.5 Information retrieval in Feature Space Mapping

Perhaps the most distinct characteristic of the Feature Space Mapping, comparing to a typical neural network systems, is the way associations are retrieved or inferences from partial input data are made. If there are many unknown factors and a few known ones humans frequently make temporarily an assumption that only one additional factor is important, fix the value of this important factor and try to determine the next factor. This type of reasoning by making a series of one-dimensional searches is equivalent to the steepest ascent linear search. An expensive alternative, used in some neural networks models, is searching in many dimensions, trying to determine all missing factors at once.

The linear searching strategy used in Feature Space Mapping is as follows:

1. Fix the value of the known factors $(X_1, .. X_k)$
2. Search for the value of X_{k+1}

   2.1 Examine all nodes assuming that $(X_{k+2}, .. X_N)$ are irrelevant

   2.2 Note the values of X_{k+1} for which $FSM(X_1, .. X_{k+1}) > \varepsilon$. Separability of FSM function allows to drop temporarily all terms that depend on $(X_{k+2}, .. X_N)$

   2.3 Fix X_{k+1} at one of the possible values and repeat the search for X_{k+2}.
3. If searches for all $m=k+1,.. N$ give positive results than new fact has been found;
   if for some m no values of X_m lead to positive results move back to the m-1 level and
   try the next X_{m-1} value for which the positive result was obtained (step 2.3)

Here ε is a small threshold. This algorithm will find all facts consistent with the fixed values of the known factors. If no factors are fixed the linear searching algorithm will find all facts stored in the Feature Space Mapping function. The depth of the search is equal to the number of unknown factors, which is usually not large. The number of facts checked is at most equal to the number of relevant facts, since not all nodes are

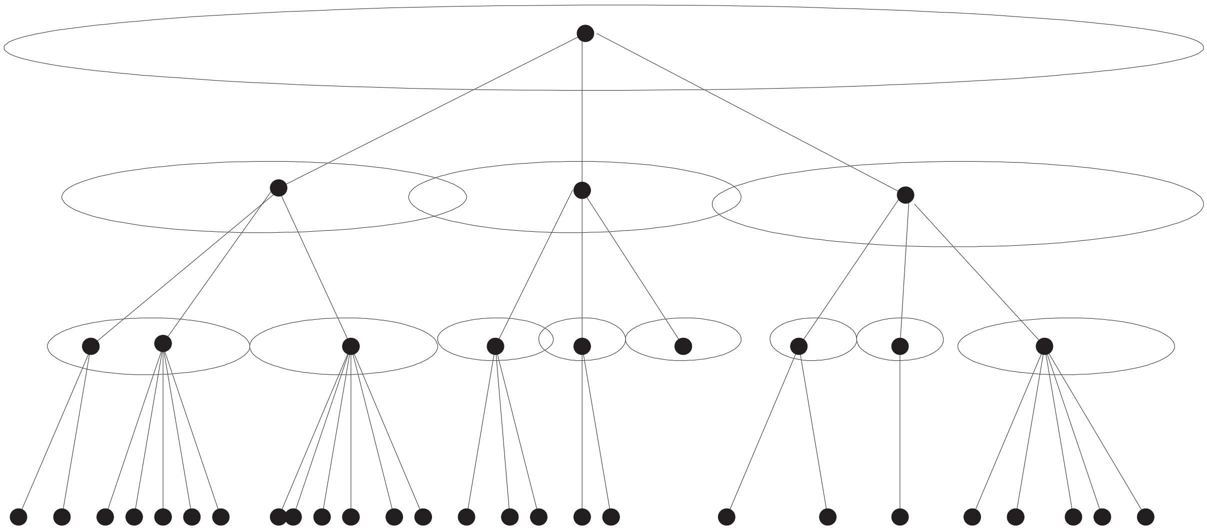

Fig. 3. A 3-level bumptree hierarchy. The highest, the input level, covers all feature space, the lowest level correspond to fuzzy facts and the volume of the feature space covered by the children nodes is totally contained in the volume covered by the parent node.

connected to all inputs: some facts may be totally irrelevant since their input space may be completely orthogonal to the input space of the question at hand. Searches are one-dimensional, therefore at a given stage the value of only a single factor in the separable node function is computed. Although the total dimensionality of this function may be quite high some factors are fixed and others are dropped from the product, giving in effect a truly one-dimensional function evaluation.

Another way to simplify searching is to use a multi-scale approach to find quickly all possible areas where *FSM* function does not vanish. Before the steepest ascent search is started the dispersions are temporarily set to large values, making it easy to identify broad regions of the feature space where detailed search should be performed with the original values of dispersion. Connection of this idea with the hierarchical bumptree structures of Feature Space Mapping network nodes will be discussed below, in the subsection on complexity.

More specific meaning for the value of *FSM* function may be assigned. For example, *FSM* values may estimate certainty of facts facilitating ordering of the final results of searches according to confidence assigned to the facts used. In the simplest case *FSM* is used as a logical function, with the values $FSM(\mathbf{X}) > \varepsilon$ as confirmation of the true facts, the negative values $FSM(\mathbf{X}) < -\varepsilon$ for false facts and with the $-\varepsilon < FSM(\mathbf{X}) < \varepsilon$ values for the “don't know” answer.

An easy way to estimate the importance of various factors at arriving at solution is to look at dispersions: those factors with small dispersion are probably decisive. The value and the gradient of the *FSM* function are used is used to find all facts close to a given fact (associations with this fact) or to a given input value **X**. If the *FSM* gradient at the point **X** is too small to define search direction dispersions of all nodes are temporarily increased by a fixed factor, making all facts more fuzzy and helping to find the closest fact in the feature space. If more associations are required the facts found so far are temporarily “switched off” and a new gradient is computed. In this way all facts that are close to **X** are quickly found even in the feature spaces of high dimensions.

3.6 Fuzzy inference systems

Another approach relevant to the Feature Space Mapping is based on abductive reasoning networks [59]. Abduction is a reasoning process, or deductive process, under uncertainty. It may be described as the process of inferring causes from effects, or the process of inductive deduction, since deduction and hypothesis forming is based on induction (examples). In this approach numeric functions and measures are used in the reasoning and description of relationship among the data items. A network of functional nodes performing numerical operation on data items is called the abductive network. In practice [59] a hierarchical, feedforward layered network structure is used, with approximations by linear, quadratic and cubic functions of the spline type. To avoid too many adjustable parameters in the networks of this type data items are grouped together and relationships in each data group summarized in a node that makes these values available to the next layer. This subdivision of problems is an approximation that sometimes breaks down and therefore it is avoided in the Feature Space Mapping model.

The Abductory Induction Mechanism (AIM) is the machine learning procedure that attempts to determine automatically the best network structure, type of nodes, connectivity and adaptive parameters minimizing the combined error measure and network complexity. The complexity measure contains terms proportional to ratio of the number of network parameters and the number of training data. Abductive networks are a particular type of fuzzy expert systems (FES) based on network realizations. The rules of the fuzzy expert systems are of the following type:

$$\text{IF } (x_1 \in X_1 \wedge x_2 \in X_2 \wedge \ldots x_N \in X_N) \text{ THEN } (y_1 \in Y_1 \wedge y_2 \in Y_2 \wedge \ldots y_M \in Y_N) \tag{3.20}$$

The rules in fuzzy expert systems are unique, i.e. the same IF part should not have a few different THEN parts. These rules may be directly programmed in the Feature Space Mapping network if many outputs from a given node are allowed. Frequently a simplification:

$$\text{IF } (x_1 \in X_1 \wedge x_2 \in X_2 \wedge \ldots x_N \in X_N \wedge y_1 \in Y_1 \ldots \wedge y_M \in Y_M) \text{ THEN true} \tag{3.21}$$

is sufficient. More general rules of the type

$$\text{IF } \left(\left(x_1 \in X_1^{(1)} \wedge \ldots x_N \in X_N^{(1)} \right) \vee \left(x_1 \in X_1^{(2)} \wedge \ldots x_N \in X_N^{(2)} \right) \vee (\ldots) \right) \text{ THEN } (y_1 \in Y_1 \ldots \wedge y_M \in Y_M) \tag{3.22}$$

may also be used in the Feature Space Mapping system. More than one conclusion Y, given conditions X, may be found using the retrieval mechanism discussed in the previous section, with the value of *FSM* function ranking the importance or confidence of these conclusions. Functional equivalence between fuzzy expert systems and radial basis function networks has been recently established [60] and holds also in the case of the Feature Space Mapping model. Therefore Feature Space Mapping belongs to the category of neurofuzzy systems.

A neural model CONSYDER for commonsense reasoning based on similar localized representation as used in the Feature Space Mapping was recently described in the literature [61]. This model uses fuzzy evidential logic covering all traditional rule based reasoning and is capable of commonsense reasoning. Distributed representation was used to capture similarities among facts stored in localized nodes. In the Feature Space Mapping hierarchical structure of nodes described below facilitates rapid finding of associations and the unreliable distributed representations are not needed. In addition recursive connections of inputs to outputs that iteratively bring second and higher order associations (see the Section 4.3 for an example) are allowed.

3.7 Complexity issues

A number of geometrical hierarchical data structures were introduced in the last few years supporting fast construction and access to data. Bumptrees are trees with leafs that correspond to facts stored in the form of localized functions. Each lower level of the hierarchy (Fig. 3) is contained in the higher level, allowing for multiscale resolution of the modeled data. Bumptrees are generalization of many other hierarchical data structures, such as oct-trees, kd-trees, boxtrees and balltrees [62]. Bumptrees are used for cluster analysis, building clusters from data (bottom up) or melting clusters (top down). In this paper bumptrees are used only to speed up queries and organize the data represented by the *FSM* function. Higher nodes in the bumptrees may represent metaconcepts or general facts while lower nodes represent specific facts. One of the simplest applications of bumptrees is to use them in pruning search trees looking for functions which have non-zero value at a specified point in the feature space. One technique that is used in the Feature Space Mapping model is to set a hierarchy of gaussian dispersion values, starting from a very diffused functions (high-level bumptree nodes) and decreasing the dispersions in a few steps to their normal values, simulating in this way a hierarchical bumptree structure (Fig. 3).

Basic operation in recognition or retrieval of facts is checking if in a given point in the feature space the value of the *FSM* function is non-zero. To speed up the retrieval a hierarchical bumptree structure of nodes is introduced. The top layer of the network contains nodes that are very fuzzy and functions realized by these nodes have nonzero values in a large volume of the feature space. The bottom layer is composed of nodes that are most focused. Nodes at each layer are connected to the next layer's nodes by a single control link and are directly connected to all inputs. If the control signal is zero then the whole search branch is deleted. For N final nodes sublinear retrieval times are obtained, proportional to the number of points in which the *FSM* function is checked at each level, times the number of levels which is of the other of log N. If the volume of feature space covered by a node is equal to half of the volume covered by the node belonging to the upper layer there are $\log_2 N$ levels. The Feature Space Mapping model should therefore be quite efficient even for a large number of facts. In practice this explicit bumptree structure is replaced by the dynamical hierarchy of bumptrees created on demand by changing the dispersion values of the processing nodes.

The network form of the Feature Space Mapping model, the localized functions performed by its nodes and sparse connectivity in a large network make hardware implementations of this model relatively simple. Implementation on a massively parallel computer is also straightforward. Feature Space Mapping system analyzes data and makes an internal representation of the data and relations among the data items. How does this network scale with the size of the problem? More precisely, how many network nodes are needed to represent with high accuracy the complexity of the data? The answer obviously depends on a problem: if linear relationship is approximated with nonlinear system, such as neural network (which is quite frequently the case [12]) many data points for training and a complex network is needed! The constructive algorithm (Section 3.3) used by Feature Space Mapping system guarantees that the complexity of generated network is minimal.

A program designed as a universal approximator should not only optimize parameters of each new node added to the system but also use optimal type of functions for the nodes. To some degree this is realized in the functional link approximation networks [63]. Separable functions (Section 3.1) used in the Feature Space Mapping allow for approximation of an arbitrary complex function but not necessarily in the most economical way (in the sense of the number of parameters used versus accuracy obtained). In the networks of fixed size large errors may arise if the number of data items available for training is not large enough because many parameters have to be set. There are two independent sources of errors. First, an intrinsic error of adaptive system that is not always capable of representing all relations. This is due to the finite number of adaptive parameters and to the convergence properties of functions performed by the nodes. Second, an estimation error due to the finite number of examples presented to the system. The intrinsic error, the estimation error and the total error (called sometimes “the generalization error”) are related in a non-obvious way. If the number of parameters and the number of data items increase to infinity the generalization error in a well-constructed network should decrease to zero. In the network of fixed architecture a small number of parameters leads to a large intrinsic error. On the other hand large number of parameters for a fixed number of data samples will increase the estimation error. If the number of data samples is arbitrarily large than the estimation error for a network with large number of parameters may be very small. Precise quantification of these statements follows below.

The rate of convergence of the Feature Space Mapping model with gaussian node functions should be similar to that of the Radial Basis Function networks. For fixed dispersions this convergence rate was found very recently [64]. Since the true function is unknown an error may only be measured in respect to the best possible (Bayes) estimate of this function, called the regression function $f_0(\mathbf{X})$. The distance between the regression function and the function realized by the radial basis functions network with n nodes, each of d dimensions, given l examples, estimated with the confidence (probability) 1-δ, is

$$E\left[(f_0(\mathbf{X})-F_{n,l}(\mathbf{X}))^2\right]=\int_X d\mathbf{X}P(\mathbf{X})(f_0(\mathbf{X})-F_{n,l}(\mathbf{X}))^2 \le O\left(\frac{1}{n}\right)+O\left(\sqrt{\frac{nd\ln(nl)-\ln\delta}{l}}\right) \tag{3.23}$$

Approximation theory gives the first factor, $O(1/n)$, while statistics gives the second factor. The error vanishes only if the network complexity, expressed by the number of its nodes n, tends to infinity slower than the number of data samples l. For any fixed number of data points there is an optimal number of network nodes that minimizes the generalization error. Since a constructive algorithm is used here, adding new nodes only for the new data, the optimal complexity is guaranteed from the beginning.

A few examples of the Feature Space Mapping applications will be given below. Problems involving learning associations, generalization, inference and logical relations are solvable by simple, nonrecursive versions of the Feature Space Mapping model.

4. Applications

4.1 Learning arbitrary associations

The simplest nontrivial problem for neural networks is the XOR (exclusive OR) problem [2], since it defines mapping that is not linearly separable:

$$(0,0) \rightarrow 0, \quad (1,1) \rightarrow 0, \quad (1,0) \rightarrow 1, \quad (0,1) \rightarrow 1$$

In the Feature Space Mapping linear separability is never an issue since all data points are defined in the space of inputs and outputs, in this case they are in the corners of a cube shown in Fig. 4.

Many examples of associations and retrieval of information from partial inputs are given in the PDP books [2] to illustrate the association and generalization capabilities of neural networks. Learning of schemata, like schemata for rooms or schemata in the "Jets and Sharks" example, are solved by the Interactive Activation Model, constraint satisfaction and competitive learning models of neural networks. The same examples are solved in a trivial way by the Feature Space Mapping model. Although association of properties of atoms or molecules with their structure would be of a more direct interest to physicists to illustrate certain characteristics of the Feature Space Mapping approach an example taken from the PDP book will be analyzed in some details.

In the room schemata (cf. Vol. II, p. 22, [2]) 40 descriptors are given for a five different kinds of rooms: living room, kitchen, office, bathroom, bedroom. Feature space is easily created from descriptions of the room furniture and other descriptors for these schematic rooms and prototype room descriptions are retrieved from partial descriptions. Most of these descriptors, like oven, computer, toilet, are of the binary type: present - not present, while others have few values (for example, room size may be very large, large, medium, small, very small). Treating all descriptors as binary discrete feature space is constructed, represented by the 40-dimensional hypercube with 2^{40} corners (possible states). The 5 schemata for rooms correspond to more than 5 corners of this hypercube since such descriptors as walls or ceiling are always present and other descriptors, like television set, may be present or not present in several types of rooms. In this 40-dimensional space there are only 5 areas, overlapping in some dimensions but well resolved in other dimensions, defining the schemata for rooms. Rumellhart *et.al.* [2] estimated the probability that each of this descriptor is present in the schemata for a given kind of room. These probabilities may be used to set dispersions for room descriptors (they are fuzzy linguistic variables in the sense of the fuzzy set theory [65]) of the 5 network nodes that correspond to the 5 schemata. One extra dimension for the room type is kept with the 40 other descriptors forming a complete fact in the feature space. After forming facts the irrelevant factors may be dropped and the network simplified. For example, there are no nodes whose output is affected by the values of such descriptors as ceiling

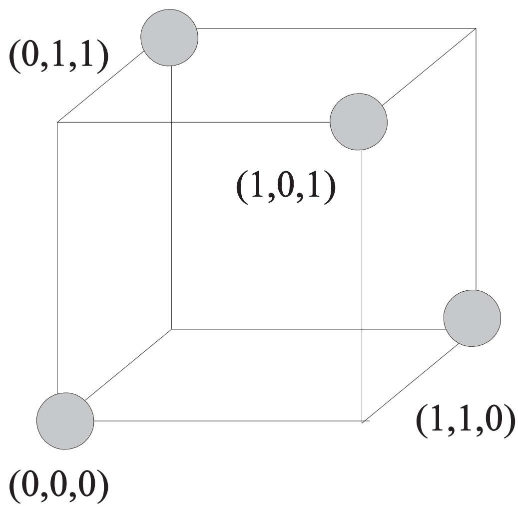

Fig. 4. Solution to the XOR problem in the Feature Space Mapping model

or walls, therefore the connections of the input nodes corresponding to these descriptors to the internal nodes may be removed.

This example is interesting because it illustrates rather well how the Feature Space Mapping searches are made and how in this model schemata are described. If only one from the 40 descriptors is given the depth of the search space is 40 and one may be afraid of combinatorial explosion. This is not the case because there are just 5 different search paths. What kind of questions can this "intelligent database" answer? First, it may recreate all facts, or schemata for a given room type. Fixing the room type all descriptors forming a room schemata are easily recovered in 40 search steps, where each step is a binary checkup answering the question: does it have such-and-such descriptor? Second, starting from a partial description a given type of room is recreated. Fixing one descriptor that is unique to some room schemata, like an oven, immediately recreates the whole schemata for kitchen since already at the highest level of searching only one type of the room gives positive answer. Fixing descriptors that could apply to many rooms, like telephone, will activate several searching paths (at most 5) and for each one again a simple search involving just 40 steps is made. A question like "can a kitchen have a telephone" requires fixing the type of room variable at the kitchen - that leaves only one active fact - and checking the telephone variable. The dispersion of the telephone variable at this node estimates the probability of positive answer to the question.

The Feature Space Mapping representation of the room schemata may be regarded as an example of an intelligent database, returning the results of queries with probabilistic interpretation. In physics and chemistry there is a need for many databases of such kind. Consider for example computational molecular physics or quantum chemistry: thousands of papers are published every year, containing calculations of properties for many atomic and molecular systems, in different basis sets, using different computational models and achieving different levels of accuracy. It should be very useful to construct a knowledge base that could answer questions like: given a molecule, property and desired accuracy suggest a method of computation and select the basis set and other parameters. Such knowledge base may be constructed from the data collected from literature. It should not contain explicit references to the results of calculations, but rather a fuzzy knowledge of what is possible. A single fact may correspond to many different results of calculations giving similar results.

What variables should such a knowledge base use? First, general definition of a system, requiring several variables, such as the charge, chemical formula or geometry. Second, description of the state of the system: space and spin symmetry, configuration, single or multireference character etc. Third, known properties of the system: ground state energy, potential energy surfaces, excitations, various moments etc. This data is still independent of computational quantum chemistry. Next step is to add knowledge related to basis sets, computational methods and estimated accuracy of calculations. Such knowledge base could tell the user what is possible using state-of-the-art techniques. A more modest version could be associated with a particular quantum chemistry package, allowing the user to estimate what a given program has to offer. Such knowledge base may be automatically created by the Feature Space Mapping system if good reference data concerning results of computations is available to create facts in the feature space. Once created the system may be continuously improved by additional training with new results. To enable predictions of new facts not only learning but also self-organization (see below) of the facts is necessary.

Many interesting problems in science and technology are so complex that fundamental theories are not very helpful in predicting or elucidating the solutions based on the first principles. Whenever a theory is not helpful experts are called for, making "educated guesses" on the basis of their experience. However, cognitive psychologists have found that experience is always connected with a specific domain [66] and that "transfer of experience" from one broad domain to another is a myth. Learning by making associations is quite effective way of creating an expert system that may replace human experts in selected domains. Human brain, having no direct "numerical sense", is not good in analyzing multidimensional data and making associations of numerical data. Computer programs are more powerful in processing large amount of data than humans are and systems like the Feature Space Mapping may be more powerful in problems requiring associations.

4.2 Classification

As an example of a nontrivial classification that Feature Space Mapping is capable of consider a spectrum stored in the form of a histogram. It is convenient to perform first Fourier or Hadamard analysis and store the coefficients of spectra instead of the original spectra. In such a case 128 to 256 numbers allow to recreate the original spectrum with high precision. The feature space has therefore 128 or 256 dimensions to store histogram components and extra dimensions to store additional information like the system identification labels (for example, names of molecules). Facts, or objects in this feature space are gaussian functions representing a

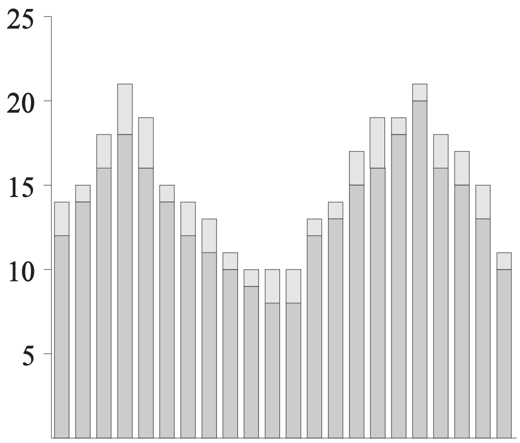

Fig. 5. Histogram of a spectrum with error bars corresponding to a gaussian function in 22 dimensions.

range of spectrums (a fuzzy spectrum) connected with the same molecular system. Since a spectrum corresponding to a given chemical system may be distorted each value of the histogram is given with an error bar (Fig. 5) equal to the dispersion of a gaussian factor in the multidimensional node function. Once created the system can learn by adding new spectra to its database and gain experience in recognition of spectra by modifying the stored representations of spectra that are already known to it. Such database of spectra allows for identification of many different systems from distorted spectra or partially known spectra.

Searching for k free parameters, each one with N values, is a search problem in k dimension requiring in a random procedure around N^k calculations. In this example both k and N may be quite large. Fortunately it is possible to factorize global searches into a few one-dimensional searches. An alternative way of searching, if many facts are closely overlapping, is by using gradient search techniques to find the maxima of *FSM* function. In multidimensional searches many problems arise with avoiding false maxima. Thanks to the separable form of the Feature Space Mapping node transfer functions factorized searches do not lead to combinatorial explosion and guarantee that all facts consistent with a given information will be found. Specific constraints and restrictions, like restrictions in the range of values that a variable can take, are also easily taken into account.

The quality of classification obviously depends on the accuracy of the representation of spectra. In the Feature Space Mapping context probability of classification errors in a space with larger number of dimensions is lower. A simple argument shows the unintuitive properties of multidimensional spaces. Suppose that input vectors are discreet, i.e. the real values are converted to the binary representation with some precision, taking for example 8 bits for each number. The dimension of the space storing histograms grows in such a case from 128 to more than one thousand. Suppose that a given spectrum is described by N bits. The probability of each of the bit strings is $1/2^N$ and the probability of finding a string at the Hamming distance (i.e. distance measured by the number of different bits in two strings) d from our spectrum is:

$$p(d) = \binom{N}{d} / 2^N \qquad (4.1)$$

This probability distribution has maximum for $d=N/2$ and dispersion equal to $N/4$. It means that on average other spectra are very far (in our example 500 bits apart) from any given spectrum, although effects of data clustering may in practice change this conclusion. The more precise is our description of the spectrum the easier it is to distinguish it from other spectra prototypes in the feature space. Feature Space Mapping takes into account the effects of data clustering by using different scales in different parts of the feature space: the dispersions of the gaussian hyperellipsoids used in this case are smaller around the points where many training data are present while in the regions of space where the data points are well resolved only a few gaussian functions with large dispersions may suffice. For some classification problems (for example in case of spectra) dispersions are given by the experimental data and should not be arbitrarily manipulated. In such cases the only way to improve the classification accuracy is to increase precision of the data by taking more details into account. Identification is made easier if not only the spectra but also other information is added to the Feature Space Mapping system. This information may be given in the form of spectra of different type, information about

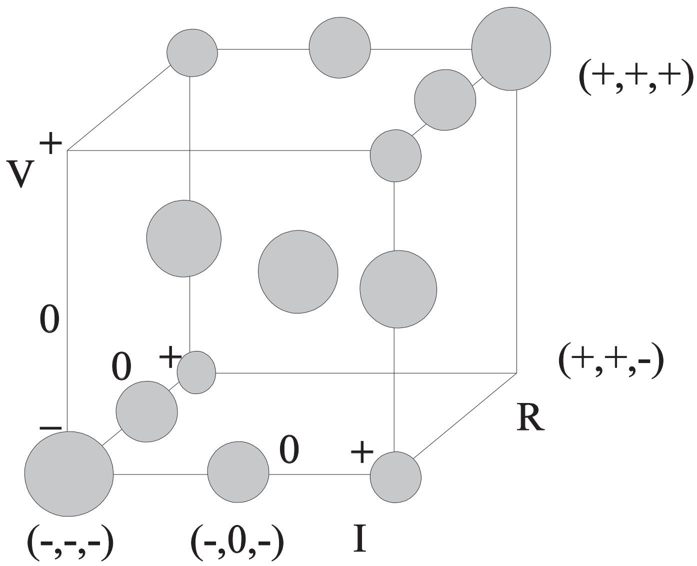

Fig 6. Representation of Ohm's law $V = I \times R$ in the Feature Space Mapping model. This feature space shows

specific properties of the substance or in any other form. It is easy to extend the feature space to combine many sources of information together in the classification process.

4.3 *Learning from general laws: fuzzy inference system*

Frequently the knowledge that human experts use is not only derived from a set of examples but, especially in physics and chemistry, also from general laws that may be applied to a given situation. These laws may be either deduced from examples or stored as *a priori* knowledge. Neural networks are usually trained on examples while expert systems are based on rules. When thinking people use the knowledge derived from equations in an intuitive, qualitative way. How can such thinking be modeled ?

Consider a specific example: Ohm's law $V=I\times R$. It involves 3 parameters, voltage V, current I and resistance R. Geometrical interpretation of this law involves a hyperboloid or a set of hyperbolas on a plain, for different values of V. This interpretation does not help in using Ohm's law in practical cases. A set of training facts should be derived and internalized as “intuition” from this law: when the current grows and the resistance is constant, what happens to the voltage? If the changes of V, I and R are designated as + for increase, 0 for staying constant and – for decrease than the number of all possible combinations of the 3 values for the 3 variables is $3^3 = 27$. Ohm's law says in effect that only 13 of them are true, for example if V is constant than I and R may not decrease or increase simultaneously. Confidence in a given fact is proportional to the number of training examples presented to the system. Therefore the fact that when V and R both decrease than I may also decrease is more probable than the fact that I may also increase (if R is decreasing faster than I is increasing).

A convenient way of expressing these intuitions about Ohm's law, or any other similar law of the form $A=B\times C$ or $A=B+C$, is to list the 13 true facts describing changes of the variables:

$$
\begin{aligned}
(A,B,C) = \quad & (+,+,+),\ (+,+,0),\ (+,0,+),\ (+,+,-),\ (+,-,+) \\
& (0,\ 0,\ 0),\ (0,+,-),\ (0,-,+), \\
& (-,-,-),\ (-,-,0),\ (-,0,-),\ (-,+,-),\ (-,-,+)
\end{aligned}
$$

These facts are shown in the three-dimensional feature space in Fig. 6. Dispersions of the gaussian functions representing facts are proportional to the confidence in a given fact, where confidence is based on the number of examples presented. Representation in Fig. 6 uses 13 gaussians localized in the three-dimensional space.

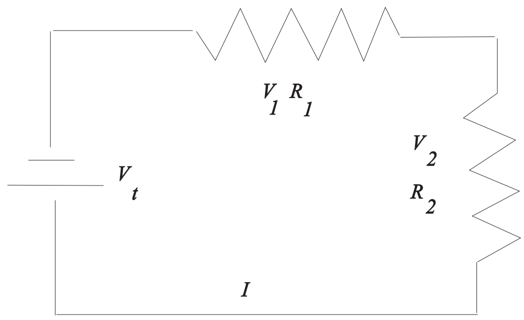

Fig. 7. Simple electrical circuit. What will happen to V_1 if R_2 grows while R_1 and V_t are kept fixed? Feature Space Mapping solves this problem in an "intuitive" way.

Other representations are also possible, for example by 6 gaussians in the two-dimensional space (extended along the axes) and one spherical gaussian in the three-dimensional space (in the middle). On the basis of such feature space it is easy to answer questions like: what will happen with the current when the voltage grows and the resistance is constant?

It may seem that the feature space representation with half of the facts that are true and half false will not be very useful in a reasoning process. Such is not the case. A more sophisticated example of the Feature Space Mapping for the representation of qualitative knowledge necessary to understand simple electrical circuits will be presented now. Although the circuit shown in Fig.7 is very simple untrained people need some time to answer questions related to the behavior of this circuit. Instead of using their intuition they try to use Kirhoff's and Ohm's laws. There are 5 equations applicable in this case:

$$\begin{gathered} V_t = V_1 + V_2 \\ R_t = R_1 + R_2 \\ V_1 = I \cdot R_1; \quad V_2 = I \cdot R_2; \quad V_t = I \cdot R_t \end{gathered} \tag{4.2}$$

The Feature Space Mapping system is trained on single elements of electrical circuits, creating for each of the five laws internal representations such as shown in Fig. 6. The full problem is defined in the 7-dimensional $(V_t,V_1,V_2,R_t,R_1,R_2,I)$ space. The question that is posed to the Feature Space Mapping system is of the following type (Smolensky, 1984, in: [2]): if R_2 increases and V_t and R_1 are constant what happens with I and V_1, V_2? This example was originally formulated for the Boltzman machine and the harmony model type of neural networks and is not so trivial to solve for these and similar models. Answering such questions using neural networks of this type requires multidimensional minimization usually done via simulated annealing method. Results from many runs have to averaged to avoid wrong answers and the process is computationally very intensive. In the Feature Space Mapping the answer requires searching for a few non-zero values of the function $FSM(V_t{=}0,V_1,V_2,R_t,R_1{=}0,R_2{=}{+},I)$ along the four (V_1,V_2,R_t,I) directions. Conventional computer program capable of answering arbitrary questions related to the change of values of variables either has to enumerate all possible cases or it has to use a complicated algorithm based on the if-then rules to apply all five laws Eq.(4.2). Feature Space Mapping solves the problem without referring to the rules. Intuitive knowledge is stored in the feature space and is used in a way that is not unlike what people do when they solve this type of problems.

Since all 5 equations have to be fulfilled simultaneously *FSM* function is taken as a product of 5 functions, each containing 13 terms corresponding to the internalized facts about each of the equations. The search process now goes as follows: take the first unknown variable V_1 and change it from –1 to +1, while the other 3 unknown variables (V_2, R_t, I) are temporarily dropped from the *FSM* function (this fact is designated by putting # in place of the dropped variable). Since node functions are separable dropped variables are simply not used in the product defining the *FSM* function:

$FSM(V_t{=}0,\ V_1,\ \#\ ,\ \#,\ R_1{=}0,\ R_2{=}{+},\ \#)$

For $V_1 = -$ the *FSM* function with irrelevant variables dropped has nonzero value, therefore search for the second variable is initiated. The function now is:

$FSM(V_t{=}\ 0,\ V_1{=}\ {-},\ V_2,\ \#,\ R_1{=}\ 0,\ R_2{=}\ {+},\ \#)$

This function does not vanish only for V_2= +; now R_t is considered:

$FSM(V_t= 0, V_1 = –, V_2 = +, R_t, R_1 = 0, R_2 = +, \#)$

This function is nonzero for R_t =+, and in the final step I = – is found. A step back is now made, to check if there are no other solutions. V_1 = 0 is taken and *FSM* does not vanish for V_2= 0 and R_t = + but vanishes for all values of I, therefore a step back is made and V_1 =+ taken. For V_2= – again R_t has to be + but *FSM* vanishes for all values of I. Therefore the only values of (V_1, V_2, R_t, I) for which *FSM* function does not vanish are (–,+,+,–).

A very tedious search in the 4-dimensional space (performed by Boltzman machine or harmony theory models [2]) has been replaced by a series of searches in one-dimensional spaces. The knowledge stored in the Feature Space Mapping is used in the search procedure in a similar way as the heuristic knowledge is used in the expert systems to reduce the blind search procedures. In the example considered here only 10 nodes have been visited. The process has some similarity to the usual human reasoning in such cases. More experienced people will first choose the ordering of variables: if R_1 = 0 and R_2 = + than it is obvious that R_t = + and since V_t= 0 therefore I has to decrease and V_1 = –, V_2 = +. This type of experience may also be used in the Feature Space Mapping. If the variables in the *FSM* function are reorder the search is shorter:

$FSM(R_1 = 0, R_2 = +, R_t, V_t= 0, I, V_1, V_2)$

In this case the “reasoning process” is more ordered. Both graphs generated by the search procedure are illustrated in Fig.8. Although Feature Space Mapping model is not biologically plausible in this case it is possible to make the dependence of the reasoning, or drawing inferences, on the search strategy psychologically plausible: changing strategies if the next step on the way to solution is too difficult. How to select the best search strategy? This meta-knowledge about strategies of solution can also be coded in the Feature Space Mapping nodes and may be learned from examples if a number of similar problems is solved. The system can learn and improve its abilities to solve problems by noting its successes and failures. The simplest heuristic rule for ordering of variables is to check if more than one choice is possible for a value of an unknown variable. The variable that is most uniquely or almost uniquely defined should be selected as the next one. In the example considered above all three choices, growing, decreasing and being constant, are possible for V_1 and V_2. Taking R_t as the first search variable only + leads to possible solution, and for the current I only – giving unique answers for the two voltages in the end. Only one search path is generated using this strategy (Fig. 8).

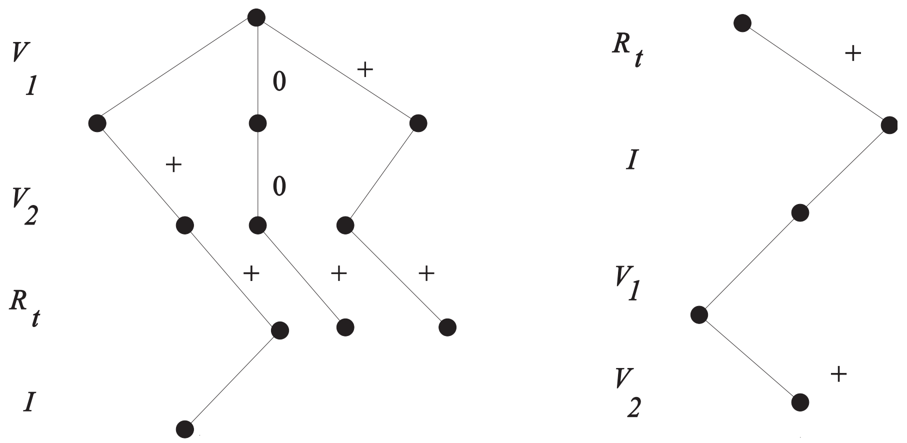

Fig. 8. Graphs generated by the search procedure with different orderings of the search variables.

Another way of solving the problem by using explicitly the knowledge about the simple subproblems connected with Ohm's and Kirhoff's law is achieved by using the two-layer Feature Space Mapping network structure presented in Fig. 9. The network consist of the input-output data nodes in one layer and the internal atoms-of-knowledge nodes in the second layer. This network works in an iterative way, trying to complete the input patterns using the knowledge, contained in the atoms-of-knowledge nodes, about the internal relations among the data variables. At the start some inputs have definite values while the state of other inputs is undefined. The task of the network is to find the undefined values. A full iteration consist of a parallel propagation of all input signals known at the start of the iteration through the network and receiving from the atoms-of-knowledge nodes values for the unknown variables that are consistent with all known facts. If a unique solution exists it will be found in a few iterations. If there are many solutions stochastic elements are introduced into the network to find them all. For the electrical circuit problem of Fig. 7 network iterations bring the following results:

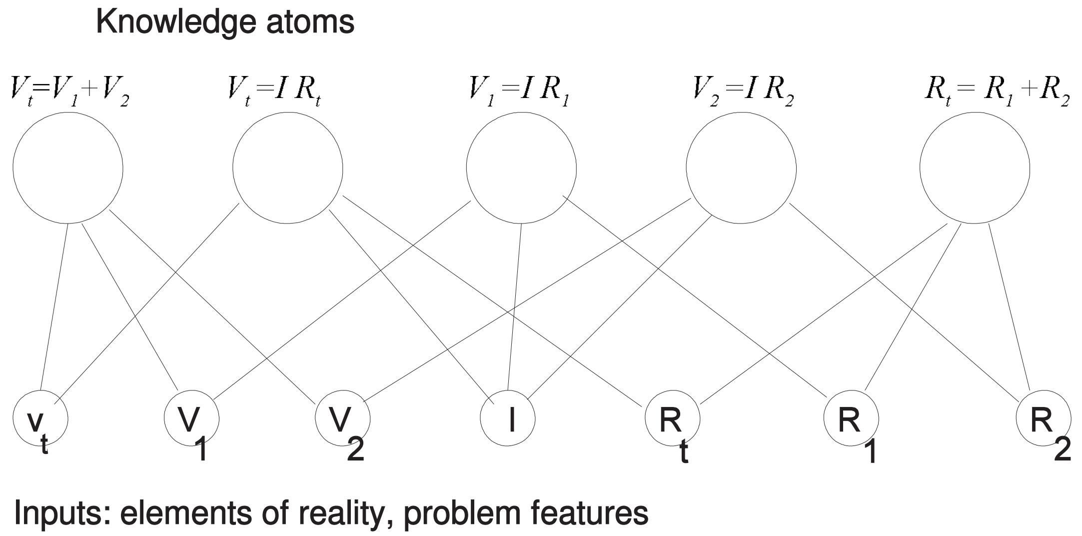

Fig. 9. Feature Space Mapping network with input-output nodes in lower layer, iteratively determining the values of the unspecified inputs consistent with the knowledge stored in the atoms-of-knowledge internal nodes .

Iteration	V_t	V_1	V_2	I	R_t	R_1	R_2
0	0					0	+
1	0				+	0	+
2	0			–	+	0	+
3	0	–	+/0/–	–	+	0	+
4	0	–	+	–	+	0	+

In the first iteration only the R_t value was determined, in the second also the value of I while in the third all other variables are uniquely defined, except for V_2 that require an additional iteration in which the node $V_t=V_1+V_2$ determines its value uniquely.

The feature space representation works as a powerful heuristics in the reasoning process. In the seven variable problem considered here there are $3^7 = 2187$ possible facts but only about 5% of them (precisely 111) are true even though for 3 variables almost 50% of facts are true. In more complicated cases, when the number of variables involved is larger and the number of values these variable may take is also larger, the percentage of facts that are true is vanishingly small, making the Feature Space Mapping very effective in recognizing only “sensible” patterns of the values of variables.

Many other search strategies may be introduced in the Feature Space Mapping systems. In a more complex system nodes and meta-nodes that are most frequently used are visited first, simulating the easiest recall of the most useful knowledge. Reasoning frequently starts from a rather general or vague ideas and concentrates on details. Multilevel approach to learning in the Feature Space Mapping system is natural: finding relevant facts by making all facts very fuzzy at the beginning of the search period and than focusing on a more localized regions in feature space using precise facts stored there. This is analogous to transition from general intuitive to logical point-wise reasoning.

4.4 Learning large number of data and self-organization

Fuzzy character of facts stored in the Feature Space Mapping allows for representation of many data items using a modest number of facts or network nodes. In fact some of the models of associative memory, like

Cerebellar Model Arithmetic Computer (CMAC), derived from data on cerebellar function, or Sparse Distributed Memory (SDM) [4] consists of nothing else but mapping from the input space to the "feature space" that blurs the data. In CMAC each data point becomes a circle and points that are close in the input space have overlapping circles, enabling generalization. The feature space nodes are connected via links with adjustable weights to the output units. It is interesting how such a simple model based on „blurring" of the input data may find many applications in robot control, pattern and signal recognition [67] and other fields. Feature Space Mapping model may be regarded as generalization of such models to the arbitrary fuzzy decision regions. The nodes of the Feature Space Mapping network compute functions that define the nonzero regions of feature space. In the CMAC model hashing functions are used to map these nonzero regions into the physical memory and to retrieve them. In the Feature Space Mapping one-dimensional searching procedures and bumptree hierarchical organization of the network are used to find facts in the feature space.

Adaptive system should not change itself into a lookup table but for a large amount of data achieve good data compression. To avoid excessive number of facts in the feature space and to preserve good generalization the training data is represented in the Feature Space Mapping system as fuzzy facts. Learning large amount of data is done in the same way as in the Learning Vector Quantization (LVQ) methods [8]. Instead of the codebook vectors $\mathbf{m}_c$ separable functions realized by the network nodes are used and instead of the distance of the codebook vectors to the new input data $\mathbf{X}$ fast searching procedures based on the bumptree hierarchical network organization are used to determine nodes nearest to the point $\mathbf{X}$. The Feature Space Mapping adaptive system tries to minimize a local error function

$$E[F_W] = \sum_{i=1}^{N} \sum_{j \in O(Ci)} K_i(X_j - C_i)(Y_j - F_W(X_j))^2 \tag{4.3}$$

where the kernel functions K_i and the neighborhood definitions $O(C_i)$ depend on the problem. Minimization is done by adapting the internal parameters of the node functions, trying to cover rather large part of the feature space. In some applications where facts are sharply defined less generalization is preferred in the inference procedure. Then facts are well localized and the feature space is almost empty. In other applications the feature space should be divided into decision regions that cover the whole space.

Learning large amounts of data in an unsupervised way may lead to self-organization of facts in the feature space. To achieve this self-organization in the Feature Space Mapping systems each new data point $\mathbf{X}$ is assigned to an existing fact or a new fact is created. If it is assigned to the existing fact $\mathbf{F}$ than fact $\mathbf{F}$ is moved in the direction of $\mathbf{X}$ and also all facts in the feature space in a certain radius from $\mathbf{X}$ are moved towards it. This algorithm finds in an unsupervised way interesting features in the data and is useful in cluster analysis of the data. Kohonen has described formation of very interesting self-organizing topological maps of patterns in a one and two-dimensional sheets of neurons using such algorithm. Unsupervised learning in Kohonen's Self-Organized Feature Mappings (SOFM)[9] is connected with the mapping from a high dimensional data or input space (represented in his case by the weight vectors and in the Feature Space Mapping model by the feature space) to a low-dimensional (usually one, two or three-dimensional) target space in which neurons are placed. Topological relations in the data space are to some degree reflected in relations among neurons in the target space. It is well known that much of the brain's activity is based on two-dimensional topographical maps (tonotopic, somatosensory, visual). Information in the input space is contained not only in the facts but also in the topography of the input space, in the metric relations between facts occupying the input space. There is no reason why this should be an Euclidean space. Those facts that tend to appear together, in the syntactic or in the semantic sense, are placed close to each other - this is possible if the coordinate system is curved. Since it is hard to draw or even imagine more than 3-dimensional feature space a method of visualization of high dimensional facts and topographical relations among them could be useful. In the Feature Space Mapping it is done via direct mapping of the feature space to some low-dimensional target space fulfilling conditions that minimize the measure of topography preservation. An example of such measure is [69]:

$$D_1(\mathbf{r}) = \sum_{i>j}^{k} (R_{ij} - r_{ij})^2 \tag{4.4}$$

Here R_{ij} are fixed numbers obtained from computing the distances between facts in the feature space and r_{ij} are unknown distances between their images in the n-dimensional target space. This function is positively defined and is equal to zero if the same distances between the objects in the target as in the input space are possible, but in general this is not the case. The minimum of this function corresponds to metric relations of k

objects mapped to the n-dimensional target space resembling as closely as possible the original relations in the input space. Should it happen that the data lies on a plane embedded in the input space such mapping to two dimensions should recognize this fact and restore perfectly metric relations among all objects. Minimization of the D_1 function leads to the same type of self-organization in the target space as the approach of Kohonen [7]. The advantage of the direct mapping is two-fold. First, feature space in the Feature Space Mapping model is more flexible than in the Kohonen's network and second, the maps obtained by the direct mapping are free from random distortions that are inherent in Kohonen's algorithm. Direct mapping with topographical constraints allows to reduce the complexity of the input data and to view the relations among data items in the feature space in one, two or three dimensions.

4.5 Time-series forecasting

Most accurate results in prediction of chaotic time series are obtained using a combination of expert system (to account for discontinuous events) and universal approximators such as neural networks. For example, prediction of power consumption based on such hybrid models work much better than other methods [68]. Simple linear autoregression leads to a very poor performance, as documented in a number of papers on time-series analysis. The trick in time-series is to analyze not the dependence of $y=f(t)$ but rather $y(t+1)=f(y(t),y(t-1)..,y(t-k))$.

Feature Space Mapping has not yet been applied to the time-series forecasting, but similar models give for such problems quite good results, highly competitive with other advanced approaches. Moody and Darken proposed an architecture using a single-layer of locally tuned processing units of the gaussian type [42]. They have used it in the time-series prediction obtaining much better results than with the backpropagation network based on sigmoidal functions. Resource Allocation Networks [44] and function estimation networks [45] were tested on the Mackey-Glass chaotic time series with excellent results. Therefore Feature Space Mapping should lead to similar accuracy.

5. Summary

The Feature Space Mapping approach has been introduced in this paper. The approach is based on direct modeling of the input data in the feature space. Facts and other type of data are represent by fuzzy objects in the feature space. These objects may have symbolic interpretation but they are multidimensional and cannot be decomposed into a symbolic label and other features. Many theoretical aspects are developed without reference to the final realization of the system. For example, classification, approximation, association and generalization can be discussed on purely theoretical grounds as processes in the feature space.

Different network realizations of this idea are possible and some of them were presented. Node processing functions of the network should be of the localized, separable type. A number of separable node functions suitable for modeling multidimensional densities of arbitrary shapes were discussed. These functions model fuzzy objects in the feature space. Nodes of the network act as filter devices around specific data points rather than as a threshold devices characteristic to the multilayer perceptrons. One way of looking at these processing elements is from the point of view of the fuzzy sets theory [65]. Each fact has a localized membership function defined by a separable function in a multidimensional feature space. Constructive algorithms are used to create the network of optimal complexity for the problem at hand, adding more nodes or removing existing nodes of the network if necessary. The network scales linearly with the number of fuzzy facts stored and is ideal for parallel processing.

The Feature Space Mapping combines qualities characteristic to neural networks and to expert systems. It may be classified as a neurofuzzy system. Neural qualities include unsupervised learning and self-organization of data; learning from examples (supervised learning); association, generalization, and formation of new categories; fine tuning of internal representations of data for high accuracy of classification and approximation. Learning is very fast and is done in one pass over the training data. Gradient techniques are applicable for forming associations. Multi-scale approach (focusing and defocusing) to exploration of relevant parts of the feature space is encouraged. Since the localized representation of the network node functions is used the Feature Space Mapping system may be analyzed in details and full control over all associations and generalizations is retained. Typical neural network based on distributed representation creates many problems: it is hard to analyze, learns slowly and from time to time exhibits unpredictable behavior, including catastrophic forgetting of all associations learned. The number of adjustable parameters in an adaptive system may be much smaller if approximating functions realized by the network nodes will include functions specific to the problem at hand. Feature Space Mapping is not restricted to gaussian functions, although for some applications gaussians are indeed an optimal choice.

Expert system qualities include explicit, localized representation of knowledge by multidimensional objects with symbolic and non-symbolic components; learning from general laws; representation of fuzzy facts; straightforward implementation of the fuzzy expert system production rules; reasoning mechanism based on searches using the knowledge contained in feature space, with the depth of search equal to the number of unknown features only; use of many search strategies; adding metaconcepts and metastrategies. Knowledge is represented by objects localized in the feature space and by the metric relations among these objects and therefore is easy to analyze. Knowledge represented in the distributed systems is delocalized and is hard to analyze [70]. Thanks to separability of the network node processing functions it is easy to focus on single unknown variable, one after another. Expert systems, including fuzzy expert systems, are not able to use "intuitive" knowledge, derived from examples while in the Feature Space Mapping system such knowledge, although hard to formalize in rules, is also used.

Among the issues not discussed here are applications of the Feature Space Mapping to model human conceptual space. Mathematical models of human behavior were based on the theory of catastrophes in the 60-ties but they failed due to the global character of the elementary catastrophes and low dimensionality of the parameter space (only 4 parameters were allowed, otherwise the number of elementary catastrophes becomes infinite). Semantic memory, based on the semantic feature space, was discussed by cognitive psychologists already in the 70-ties [66] and the topography of semantic spaces found in the psychological experiments is quite similar to the topography obtained by the self-organized semantic mapping [7]. Same results are obtained by the direct mapping from the feature space to the two-dimensional target space [69] that simulates feature detection and specialization in the cerebral cortex.

Feature Space Mapping may be derived as a special case from a general idea of a mind space and the language describing mind events [71]. Such models offer also an interesting perspective on the problems laying at the foundation of cognitive sciences, such as the famous mind-body problem. Many networks (brains or bodies) may realize the same function (mind), and many functions may be realized by the same network. The behavior of the system is determined by the knowledge contained in an abstract mind space (in this paper feature space). The objects in this space are nondecomposable, multidimensional mixtures of sensory, motoric, abstract and hidden, internal features. Both aspects, mind and body, or abstract mind space and hardware network realization, are thus just two aspects of the same system. The language of neuronal excitations (state of the network) is not useful for description of mental events, and vice versa, psychological concepts (or concepts related to the mind space) are useless to talk about neural events.

A number of possible applications of the Feature Space Mapping system in physics and chemistry were indicated, such as creating the intelligent knowledge bases or qualitative analysis of complex systems. Feature Space Mapping already now deserves the name of a universal adaptive system. Theoretical foundations and software implementations of such universal systems should become one of the most important research topics in the near future. It is not yet clear what are the limitations of such systems.

Acknowledgments

W.D. gratefully acknowledges support of the Max-Planck Institute during his visits in Garching. It is a pleasure to thank prof. M. Kłobukowski, Mr. Norbert Jankowski and Antoine Naud for interesting discussions on numerous subjects related to this paper.

References

[1] J.A. Anderson, and E. Rosenfeld, (Eds). Neurocomputing: Foundations of Research. (The MIT Press: Cambridge, MA.(1988)); J.A. Anderson, A. Pellionisz, and E. Rosenfeld, (Eds). Neurocomputing 2: Directions for Research. (The MIT Press: Cambridge, MA. 1990)

[2] J.L. McClelland and D.E Rumelhart, Explorations in Parallel Distributed Processing: Computational Models of Cognition and Perception (The MIT Press, Cambridge, MA 1986)

[3] M. Caudill and C. Butler, Naturally Intelligent Systems. (The MIT Press: Cambridge, Massachusetts, 1990); D.S. Levine, Introduction to Neural and Cognitive Modeling. (Lawrence Erlbaum: Hillsdale, N.J 1990); M. Zeidenberg, Neural Networks in Artificial Intelligence. (Ellis Horwood, Ltd., Chichester. 1990); R. Hecht-Nielsen, Neurocomputing. (Addison Wesley, 1990); J. Hertz, A. Krogh and R. Palmer, Introduction to the Theory of Neural Computation. (Addison-Wesley: Redwood City, California 1991)

[4] P.D. Wasserman, Advanced Methods in Neural Networks. (Van Nostrand Reinhold: New York, 1992)

[5] B. Widrow, M. A. Lehr, Proceedings of the IEEE 78 (1990) 1415-1442

[6] B. Humpert, Comput. Phys. Commun. 58-75 (1990) 223

[7] T. Kohonen, Neural Networks 1 (1988) 3-16

[8] T. Kohonen, Proceedings of the IEEE 78 (1990) 1464-1480

[9] T. Kohonen, Self-organization and Associative Memory. (Springer-Verlag, New York, 1984, 2nd Edition: 1988; 3rd edition: 1989).

[10] A. Cherubini and R. Odorico, Comput. Phys. Commun. 72 (1992) 249

[11] Communications of the ACM, March 1994

[12] W. Duch and G.H.F. Diercksen, Comp. Phys. Commun. (in print)

[13] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning representations by backpropagating errors. Nature, 323 (1986) 533-536; D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, (The MIT Press, Cambridge, MA 1986), Vol.1, pp 318-362; P.J. Werbos, Proc. of IEEE 78 (1990) 1550

[14] P. Peretto, An Introduction to the Modeling of Neural Networks (Cambridge Univ. Press, UK 1992)

[15] T.L.H. Watkin, A. Rau and M. Biehl, Rev. Mod. Phys. 65 (1993) 499

[16] W. Duch, Neural Network World (in print; also presented at the NeuroNet'93 conference); Towards Artificial Minds, in: Proc. of I National Conference on neural networks and applications, Kule, April 1994 (also UMK-KMK-TR 1/94 report)

[17] J.J. Hopfield, Proc. Nat. Acad. Sci. 79 (1982) 2554; ibid 81 (1984) 3088

[18] D.H. Ackley, G.E. Hinton and T.J. Sejnowski, Cognitive Science 9 (1985) 147

[19] E.B. Baum, Neural Comput. 1 (1989) 201-207

[20] R.O. Duda, P.E. Hart, Pattern classification and scene analysis (New York, NY, Wiley 1973)

[21] G.S. Sebestyen, Decision-making Processes in Pattern Recognition (Macmillian, New York 1962)

[22] C.M. Bachmann, L.N. Cooper, A. Dembo, O. Zeitouni, Proc. Nat. Acad. Sci. 21 (1987) 7529-7531

[23]B.G. Batchelor, Methods of pattern classification. In: “Practical Approach to Pattern Classification” (London, Plenum 1974); A.G. Arkadev, E.M. Braverman, Teaching computers to recognize patterns (Academic Press, London 1967)

[24] D.L. Reilly, L.N. Cooper, C. Elbaum, Biol. Cybern. 45 (1982) 35

[25] S.N. Kavuri, V. Venkatasubramanian, Comp. Chem. Engng 17 (1993) 765

[26] A. Nigrin, Neural Networks for Pattern Recognition. (Cambridge MA: The MIT Press 1993).

[27] G. Cybenko, Math Control Systems Signals 2 (1989) 303; K. Funahashi, Neural Networks 2 (1989) 183; K. Hornik, M. Stinchcombe and H. White, Neural Networks 2 (1989) 359; K.L. Jones, Proceedings of the IEEE 78 (1990) 1586

[28] E.J. Hartman, J.D. Keeler and J.M. Kowalski, Neural Computation 2 (1990) 210; J. Park and I.W. Sandberg, Neural Computations 3 (1991) 246

[29] V. Kůrková, K. Hlaváčková, Proceedings of the Neuronet'93, Prague (1993)

[30] H. Leung, S. Haykin, Neural Computation 5 (1993) 928

[31] C. Bishop, Neural Comput. 3 (1991) 579-588

[32] B. Ripley, Flexible non-linear approaches to classification. In: From Statistics to neural networks. Theory and Pattern Recognition Applications, eds. V. Cherkassky, J.H. Friedman, W. Wechsler (Springer Verlag 1994)

[33] D.F. Specht, Proc. of IEEE Intern. Conference on Neural Networks 1 (1988) 525-533

[34] D.F. Specht, IEEE Transactions on Neural Networks 2 (1991) 568; H. Schiøler, U. Hartmann, Neural Networks 5 (1992) 903-909

[35] R. R. Trippi, E. Turban E., Neural Networks in Finance and Investing, (Chicago, Ill, Probus 1992).

[36] J. D. Farmer, J. J. Sidorowich, Phys. Rev. Letters 59 (1987) 845-848; M. Casdagli, Physical Review A, 35 (1989) 335-356; J.B. Elsner, J. Phy. A 25 (1992) 843-50

[37] T.D. Sanger, Neural Comput. 3 (1991) 67-78

[38] R. Durbin, D.E. Rumelhart, Neur. Comput. 1 (1989) 133

[39] E. Hartman and J. D. Keeler, Neural Computation 3 (1991) 566

[40] IEEE Transactions on Circuits and Systems 40 (1993), No. 3, speciall issue on cellular neural networks.

[41] T. Poggio, F. Girosi, Proc. of the IEEE 78 (1990) 1481; M.J.D. Powell, “Radial basis functions for mulitvariable interpolation: a review”, in: J.C. Mason and M.G. Cox, eds, Algorithms for Approximation. (Clarendon Press, Oxford 1987); Dyn N, Interpolation and Approximation by Radial and Related Functions, in: Approximation Theory VI, Vol. 1, C.K. Chiu, L.L Schumaker and J.D. Watts (Eds) (Academic Press 1989).

[42] J. Moody, C.J. Darken, Neural Comput. 1 (1989) 281-294 ;

[43] E.Alpaydin, GAL: networks that grow when they learn and shrink when they forget. Int. Computer Sci. Inst., Berkley, CA Tech. Rep. 91-032 (1991)

[44] J. Platt, Neural Comput. 3 (1991) 213

[45] V. Kadirkamanathan, M. Niranjan, Neural Computation 5 (1993) 954

[46] B. Fritzke, Vector quantization with growing and splitting elastic net, in: ICANN '93: Proceedings of the International Conference on artificial neural networks, Amsterdam 1993

[47] S.E. Fahlman, C. Lebiere, Tech. Rep. CMU-CS-90-100, Carnegi-Mellon School of Comp. Sci. (1990)

[48] R. Penrose, The Emperor's new mind (Oxford Univ. Press 1989)

[49] A. Turing, Mind 59 (1950), reprinted in: D.R. Hofstadter, D.C. Dennett, The mind's I (Basic Books, New York 1981)

[50] L.M. Encarnação, M.H. Gross, An adaptive classification scheme tto approximate decision boundries using local Bayes criteria - the “Melting Octree” Network. Int. Computer Sci. Inst., Berkley, CA Tech. Rep. 91-047 (1992)

[51] L. Bottou, V. Vapnik, Neural Comput. 4 (1992) 888-901

[52] V. Vapnik, L. Bottou, Neural Comput. 5 (1993) 893-909

[53] R.A. Jacobs, M.I.Jordan, S.J. Nowlan, G.E. Hinton, Neural Comput. 3 (1991) 79-87

[54] J. Buhmann, H. Kühnel, Neural Computation 5 (1993) 75-88

[55] J.H. Bentley, Commun. ACM 18 (1975) 509-517

[56] L. Breiman, J. Friedman, R. Olshen, C.J. Stone, Classification and regression trees (Wadsworth, Belmont, CA 1984)

[57] J.H. Friedman Multivariate adaptive regression splines, Tech Rep. 102 (1988), Stanford Univ. Lab. for Computational Statistics.

[58] Y-f. Wong, Neural Comput. 5 (1993) 89

[59] G.J. Montgomery, K. C. Drake, Neurocomputing 2 (1991) 97

[60] J-S. R. Jang, C_T. Sun, IEEE Transactions on Neural Networks 4 (1993) 156-158

[61] R. Sun, A connectionist model for commonsense reasoning incorporating rules and similarities, in: Knowledge Acquisitions (Academi Press, Cambridge 1992).

[62] S. Omohundro, Complex Systems 1 (1987) 273-347; Physica D 42 (1990) 307-321; Bumptrees for efficient function, constraint and classification learning. Im: Advances in neural information systems 3, eds. Lippman, Moody and Touretzky (Morgan Kaufman Publishers, San Mateo, CA 1991)

[63] Y-H. Pao, Adaptive Pattern recognition and neural networks (Addison-Wesley, Readnig, MA 1989)

[64] P. Niyogi, F. Girosi, AI Memo 1467, CBCL Memo 88, MIT AI Laboratory, 1994 (available via ftp from publications.ai.mit.edu)

[65] G.J. Klir and T.A. Folger, Fuzzy Sets, Uncertainity and Information. (Prentice Hall, NJ 1988)

[66] R.E. Mayer, Thinking, problem solving, cognition (WH Freeman 1992); D.E. Meyer, cognitive Psychology 1 (1970) 242-300

[67] W.T. Miller III, F.H. Glanz and L.G. Kraft, Proc. of the IEEE 78 (1990) 1561

[68] K-H. Kim, D-Y. Park, J-K. Park, Expert System Application to Power Systems IV, Melbourne (1993), pp. 164-168

[69] W. Duch, Open Systems and Information Dynamics (1994, in print)

[70] S.I. Gallant, Neural network learning and expert systems (Bradfor Book, MIT Press 1993)

[71] W. Duch, Mind space - a solution to the mind-body problem (1994, submitted to PSYCOLOQUY)