Towards comprehensive foundations of computational intelligence.

Włodzisław Duch

Department of Informatics, Nicolaus Copernicus University, Grudziądzka 5, Toruń, Poland,
and School of Computer Engineering, Nanyang Technological University, Singapore.

Abstract. Although computational intelligence (CI) covers a vast variety of different methods it still lacks an integrative theory. Several proposals for CI foundations are discussed: computing and cognition as compression, meta-learning as search in the space of data models, (dis)similarity based methods providing a framework for such meta-learning, and a more general approach based on chains of transformations. Many useful transformations that extract information from features are discussed. Heterogeneous adaptive systems are presented as particular example of transformation-based systems, and the goal of learning is redefined to facilitate creation of simpler data models. The need to understand data structures leads to techniques for logical and prototype-based rule extraction, and to generation of multiple alternative models, while the need to increase predictive power of adaptive models leads to committees of competent models. Learning from partial observations is a natural extension towards reasoning based on perceptions, and an approach to intuitive solving of such problems is presented. Throughout the paper neurocognitive inspirations are frequently used and are especially important in modeling of the higher cognitive functions. Promising directions such as liquid and laminar computing are identified and many open problems presented.

1 Introduction

Computational intelligence emerged from interactions of several research communities with overlapping interests, inspired by observations of natural information processing. H. Spencer in his "Principles of Psychology" published in 1855 drew neural network diagrams and made a conjecture that all intelligence may be interpreted in terms of successful associations between psychological states driven by the strength of connections between the internal states (see the early history of connectionism in [172]).

Research in artificial neural networks (ANNs) grew out from attempts to drastically simplify biophysical neural models, and for a long time was focused on logical and graded response (sigmoidal) neurons [5] used for classification, approximation, association and vector quantization approaches used for clusterization and self-organization [106]. Later any kind of basis function expansion, used since a long time in approximation theory [143] and quite common in pattern recognition, became "a neural network" [4], with radial basis function (RBF) networks [140] becoming a major alternative to multilayer perceptron networks (MLPs). However, as pointed out by Minsky and Papert [131] there are some problems that such networks cannot solve. Although these

authors were wrong about the XOR (or the parity) problem which is easily solved by adding hidden neurons to the network, they were right about the topological invariants of patterns, in particular about the problem of connectedness (determining if the pattern is connected or disconnected). Such problems can only be solved with a different type of neurons that include at least one phase-sensitive parameter [111], or with spiking neurons [173]. Computational neuroscience is based on spiking neurons [68], and although mathematical characterization of their power has been described [119] their practical applications are still limited. On the other hand feedforward artificial neural networks found wide applications in data analysis [144] and knowledge extraction [62]. Better understanding of mathematical foundations brought extensions of neural techniques towards statistical pattern recognition models, such as the Support Vector Machines (SVMs) [155] for supervised learning and Independent Component Analysis [90] and similar techniques for unsupervised learning.

Most networks are composed of elements that perform very simple functions, such as squashed weighted summation of their inputs, or some distance-based function [57, 58]. Connectionist modeling in psychology [151] introduced nodes representing whole concepts, or states of network subconfigurations, although their exact relations to neural processes were never elucidated. It was natural from this point of view to extend connectionist networks to all kinds of graphical models [99], including Bayesian belief networks and abstract network models for parallel distributed processing. Concept nodes, representing information derived from perception or from some measurements, are not too precise and thus may be represented in terms of fuzzy membership functions. Fuzzy models may be formally derived as generalization of multivalued logics, but the field has also rather obvious cognitive inspirations related to models of intelligent behavior at a higher, psychological level, rather than elementary neural level. Sets of fuzzy rules have a natural graphical representation [116], and are deeply connected to neural networks [37]. Fuzzy rules organized in a network form may be tuned by adaptive techniques used in neural networks, therefore they are called neurofuzzy systems [133, 136]. Thus fuzzy and neural systems are at the core of computational intelligence.

Brains and sensory organs have been structured and optimized by the evolutionary processes to perform specific functions. This inspiration led to introduction of evolutionary programming Fogel66,Goldberg89, and later also other biologically-inspired optimization approaches, such as ant, swarm, and immunological system algorithms [16, 105, 30], that can be used for optimization of adaptive parameters in neural and neurofuzzy systems. Although the algorithms based on these diverse biological inspirations are used for similar applications the time-scales of evolutionary, behavioral and immunological processes involved are very different, and the type of intelligence they are related to is also quite different.

The three main branches of computational intelligence are thus inspired by evolutionary processes that structured brains and intelligence, low-level brain processes that enable perception and sensorimotor reactions (primary sensory and motor cortices), and intermediate level fuzzy concepts and associative reasoning (higher sensory areas in temporal and parietal lobes). To cover all phenomena related to intelligence in a computational framework representation and reasoning based on complex knowledge structures is needed. Traditionally artificial intelligence (AI) has been concerned with

high level cognition, using the symbolic knowledge modeling to solve problems that require sequential reasoning, planning and understanding of language, but ignoring learning and associative memories. High-level cognition requires different approach than perception-action sequences at the lower cognitive level, where artificial neural networks, pattern recognition and control techniques are used. Knowledge used in reasoning and understanding language is based on a large number of concepts with complex structure, huge amount of diverse information that may be combined in an infinite number of ways. Making inferences from partial observations requires systematic search techniques and may draw inspirations from decision-making processes in which prefrontal cortex of the brain is involved. One of the big challenges facing CI community is integration of the good-old fashioned artificial intelligence (GOFAI). Although some attempts in this direction have been made [88] typical textbooks on artificial intelligence [179, 152] include very little information on neural networks or fuzzy systems, with learning reduced to probabilistic, Bayesian models, maximum likelihood approaches, and symbolic machine learning methods. Overlap between the two communities in terms of conference topics, or journal publications has always been minimal. Each branch of CI has its natural areas of application requiring methods that may not overlap significantly. Even neural networks and pattern recognition communities, despite a considerable overlap in applications, tend to be separated.

This situation may change in near future. Development of artificial pets or other autonomous systems that should survive in hostile environment is a great challenge for signal analysis to model perception, control systems for behavioral modeling, and perception-based reasoning including attention. Autonomous agents, such as robots, need to reason using both abstract knowledge and information based on perceptions, information coming from sensors, categorized into information granules that are easier to handle. Efforts in cognitive robotics require combination of high behavioral competence with human-level higher cognitive competencies. Autonomous agents should be based on cognitive architectures that integrate low and high-level cognitive functions. This area is slowly gaining popularity and will be a natural meeting ground for all branches of computational intelligence. Development of chatterbots that involve people in interesting conversation is based now on natural language processing and knowledge representation techniques, but it remains to be seen how far one can go in this direction without including real perception. Another driving force that should encourage the use of search techniques that form the basis for AI problem solving is the "crises of the richness" that afflicts computational intelligence. Recent component-based data mining packages [1] contain hundreds of learning methods, input transformations, pre- and post-processing components that may be combined in thousands of ways. Some form of meta-learning that should automatically discover interesting models is urgently needed and it has to be based on search in the model space. Heuristic search techniques have been developed to solve complex combinatorial problems and CI has reached the stage now where they should be used.

In this paper an attempt is made to outline foundations for a large part of computational intelligence research, identify open problems and promising directions, show how to solve this "crises of the richness" and how to go beyond pattern recognition, towards

[1] See: http://en.wikipedia.org/wiki/Data_mining

problems that are of interest in artificial intelligence, such as learning from partial observations or perceptions, and systematic reasoning based on perceptions. The emphasis here is on architectures and capabilities of models, rather then train techniques, therefore evolutionary and other optimization approaches are not discussed. In the second section foundations of computational intelligence are discussed. Computing and cognition seen from the perspective of compression of information is analyzed, introducing a new measure of syntactic and semantic information content, heterogeneous systems that may discover specific bias in the data, and meta-learning scheme to discover good data models in an automatic way. In the third section a general CI theory based on composition of transformations is outlined, showing how new information is generated, extracted from the data, how to use heterogeneous learning systems, redefine the goal of learning in case of difficult problems, understand data in the similarity-based framework and use many individual models in meta-learning schemes. Section four shows how to go beyond pattern recognition using intuitive computing and correlation machines. Neurocognitive inspirations are very important at every step and are discussed in section five. A summary of open problems closes this paper.

2 Searching for computational intelligence foundations

A quick glance on some books with "computational intelligence" title [107, 132, 136] shows that the field still lacks a coherent framework. Many branches of CI are presented one after another, with distant biological inspirations as a common root, although in case of such statistical techniques as kernel methods or SVMs such inspirations cannot be provided. Different experts define computational intelligence as a collection of computational techniques that are glued together only for historical or even personal reasons. Modern pattern recognition textbooks [63, 174, 164] start from the Bayesian probabilistic foundations that may be used to justify both discriminat as well as the nearest neighbor type of learning methods. Supervised and unsupervised pre-processing techniques, classification, rule extraction, approximation and data modeling methods, cost functions and adaptive parameter optimization algorithms are used as components that may be combined in thousands of ways to create adaptive systems.

Can there be a common foundation for most computational intelligence methods guiding the creation of adaptive systems? Computational learning theory [102] is a rigorous mathematical approach to learning, but it covers only the theoretical aspects of learning and is rarely used in practice. In brain science it has been commonly assumed (although only recently tested [160]) that sensory systems and neurons in the primary sensory cortex are well adapted through the evolutionary and developmental processes to the statistical properties of the visual, auditory and other types of signals they process. Neurons in the higher sensory areas react to progressively more complex aspects of signal structure. Difficult, ill-determined problems (including perception and natural language processing) may be solved only by using extensive background knowledge. Despite relatively rigid architecture of primary sensory areas neurons can quickly adapt to changes in the sensory signal statistics, for example variance, contrast, orientation and spatial scale. Thus a series of transformations is made before sufficient amount of information is derived from the signal to perform object identification, and then infor-

mation about the object is used for associations and further reasoning. Inspirations from brain sciences serve below to discuss some challenges facing CI, and will be explored in more details later in this paper.

2.1 Some challenges for CI

Intelligent systems should have goals, select appropriate data, extract information from data, create percepts and reason with them to find new knowledge. Goal setting may be a hierarchical process, with many subgoals forming a plan of action or solution to a problem. Humans are very flexible in finding alternative ways to solve a given problem, and a single-objective solutions are rarely sufficient. Brains have sufficient resources to search for alternative solutions to the problem, recruiting many specialized modules in this process. An important challenge for computational intelligence is thus to create flexible systems that can use their modules to explore various ways to solve the same problem, proposing multiple solutions that may have different advantages. This idea will be explored below using a meta-learning search process in the space of all possible models that may be composed from available transformations. The great advantage of Lisp programming is that the program may modify itself. There are no examples of CI programs that could adjust themselves in a deeper way, beyond parameter optimization, to the problem analyzed. This idea has been partially implemented in the similarity-based meta-learning scheme [35, 53], and is also in accord with evolving programs and connectionist systems [101] that to a limited degree change their structure.

Most CI algorithms have very limited goals, such as prediction (using approximation method) or diagnosis (classification) based on data with some fixed structure. Such algorithms are essential building blocks of general intelligent systems, although their goals and information flow is determined by the user who tries to find a method that works for a given data. For example, running neural network software the user has to make many decisions, designing the network, selecting the training method, setting parameters, preparing the data, evaluating results and repeating the whole cycle. In effect the user acts as an external controller for the system, while the brain has parts controlling other parts [149, 150]. With sufficiently large library of different procedures for data preprocessing, feature selection and transformation, creation of data models, optimization of these models and postprocessing of results (already hundreds of components are available in such packages as Weka [180], Yale [129] and others) the control problem becomes quite difficult and the number of possible variants of such approach guarantees a constant supply of conference papers for many years to come.

Most efforts in the computational intelligence field goes into the improvement of individual algorithms. For example, model selection [164] in neural networks is usually restricted to architectures (number of nodes, each performing the same type of functions) and improvements of the training schemes; in the decision tree the focus is on the node splitting criteria and pruning strategies. The current focus on accuracy improvements of individual models, dominating in the academic journal and conference papers, is hard to justify both from the practical and theoretical point of view. The ‘no free lunch’ theorem [63, 174] shows that there is no single learning algorithm that is inherently superior to all other algorithms. In real world applications there may be many

additional considerations, different methods may offer different advantages, for example presenting results in comprehensible way or using features of the problem that can be obtained with lower costs or effort. These considerations are almost never addressed in the literature on learning systems. In practice "Experience with a broad range of techniques is the best insurance for solving arbitrary new classification problems (Chapter 9.2.1, [63]). Moreover, one should find all algorithms that work sufficiently well for a given data, but offer different advantages. Although data mining packages include now many algorithms still some of the best algorithms used in the *StatLog* project [128] are not implemented in research or commercial software. It is doubtful that new algorithms are going to be always significantly better. Most programs have many parameters and it is impossible to master them all.

The first challenge for CI is thus to create flexible systems that can configure themselves finding several interesting solutions for a given tasks. Instead of a single learning algorithm priorities may be set to define what will be an interesting solution, and a system that automatically creates algorithms on demand should search for configurations of computational modules in the space of all models restricted by the user priorities. For example, if the goal is to understand or make a comprehensible model of the data, methods that extract rules from data or that find interesting prototypes in the data should be preferred, although methods that provide interesting visualizations may also be considered. A lot of knowledge about reliability of data samples, possible outliers, suspected cases, relative costs of features or their redundancies is usually ignored as there is no way to pass it to and to use it in CI programs. Models that are not working well on all data may work fine on some subsets of data and be still useful. In practical applications validation and verification of solutions may be of great importance.

These challenges have been only partially addressed so far by CI community. Systematic generation of interesting models has been the topic of meta-learning research. In the simplest version meta-learning may be reduced to a search for good models among those available in the data mining packages, a search in the model/parameter space. In the machine learning field the multistrategy learning has been introduced by Michalski [64]. Learning of a single model may be sufficiently difficult, therefore to be feasible search in the space of many possible models should be heuristically guided. The Metal project [71] tried to collect information about data characteristics and correlate it with the methods that performed well on a given data. A system recommending classification methods for a given data has been built using this principle, but it works well only in a rather simple cases. Not much is known about the use of heuristic knowledge to guide the search for interesting models. One problem with most meta-learning approaches is that the granularity of the existing models is too large and thus only a small subspace of all possible models is explored.

Although computational intelligence covers a vast variety of different methods it lacks integrative theory. Bayesian approaches to classification and approximation in pattern recognition [63, 174, 164] covers mostly statistical methods, leaving many neural and fuzzy approaches in CI largely outside of their scope. There is an abundance of specialized, ad hoc algorithms for information selection, clusterization, classification and approximation tasks. An integrative theory is needed, providing a good view of interrelations of various approaches, and a good start for meta-learning, that could

automatically find an appropriate tool for a given task. Creating such a theory is a great challenge. Some guiding principles that address it are described below.

2.2 Computing and cognition as compression

Neural information processing in perception and cognition is based on the principles of economy, or information compression [22]. In computing these ideas have been captured by such concepts as the minimum (message) length encoding, minimum description length, or general algorithmic complexity [117]. An approach to information compression by multiple alignment, unification and search has been proposed as a unifying principle in computing, analysis and production of natural language, fuzzy pattern recognition, probabilistic reasoning and unsupervised inductive learning [181, 182], but so far only models for sequential data have been considered. The difficulty in applying such principles to real problems are due to the importance of underlying structures for handling information and knowledge representation. Multiple alignment is sufficient for sequential information that is stored in string but not for structured information that is stored in the brain subnetworks in a way that is not yet fully understood.

Information compression and encoding of new information in terms of old has been used to define the measure of syntactic and semantic information introduced in [56]. This information is based on the size of the minimal graph representing a given data structure or knowledge–base specification, thus it goes beyond alignment of sequences. A chunk of information has different value for someone who does not have any associations with it and treats it as a fact to be ignored or remembered, and a very different value for an expert who may have to restructure many existing associations to accommodate it. Semantic information measure, introduced in [56], is proportional to the change of algorithmic (Chaitin-Kolmogorov) information [20] that is needed to describe the whole system, and therefore measures relative complexity, depending on the knowledge already accumulated. Algorithmic information or the relative complexity of an object y in respect to a given object x is defined as the minimal length of the program p for obtaining y from x. Algorithmic information captures some intuitive features of information: a binary string obtained by truly random process cannot be compressed and carries the amount of information equal to the number of its digits. An apparently random string may, however, be easily computable from some short algorithm, for example a square root or the value of π. Because of the Gödel and related theorems it is in general not possible to decide whether the string is random or simple to compute. Although algorithmic information has correct intuitive features it is usually very hard to compute, relying on a concept of universal computer.

Suppose that information is encoded in n-element binary strings. There are 2^n possible strings and if they are chosen in a completely random way only a little compression will be possible. Each new string will contribute about n bits of information and strings will be represented in form of a binary tree. Encoding new information in terms of the old is possible in various ways if some parts of the new bit strings are already present in the previously analyzed old bit strings. For example, if the new string B_n differs from an old string B_o only by the last bit b_n the whole B_n string contributes only one bit of information relatively to B_o. If many bit strings are received the whole information contained in them may be presented in a binary tree and folded into a minimal graph. If

all 2^n strings are present the minimal graph may be represented in a very compact form (Fig. 1, left). However, if all but the last string 11111 are received the minimal graph (created by folding binary tree with just one edge removed) is rather complicated (Fig. 1, right). Adding the last string carries – from the point of view of the whole system that builds internal representation of the structure of the incoming information – a large amount of information, reducing the 5-bit graph by 7 edges and 4 vertices. The number of edges plus vertices changed in the minimal graph representing all data is thus a useful measure of the structural information that is gained by receiving new information. If the strings repeat themselves no new structural information is gained, although frequency of repeated information chunks may be useful for other purposes.

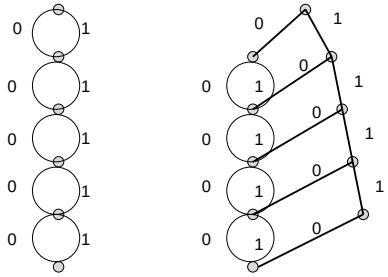

Fig. 1. Left: minimal graph representing a set of all 32 binary 5-bit strings. Right: minimal graph for a set of all 5-bit binary strings, without 11111 string. Such graphs are created by folding binary trees to minimal graphs.

Semantic contents or meaning of information may only be defined in a context of cognitive system, for example an expert system based on a set of rules stored in a knowledge base, or a semantic network. Knowledge base, together with the rules of inference, defines a universum of facts, representing knowledge or some model of the world. Information gain relatively to the knowledge base is defined as the change of the size of the minimal knowledge based that can accommodate this fact and its consequences. When a new fact is learned by a human it may take from several seconds to days or even years before this fact is truly accommodated and the meaning is fully understood. Years of repetition of basic facts in mathematics and natural sciences are required to really digest this information: once it has been integrated into the "knowledge base" the information contained in the school book can be quickly reviewed and new information encoded in terms of the old. Adding new rule to the knowledge base requires accommodation of this knowledge, and thus modification of existing knowledge. For example, learning that some fruits are edible may require a short sequence of symbols to transmit, but will have large influence on the knowledge based, creating new goals and modifying behavior of humans and animals.

Perfect minimal encoding is probably never accomplished in real brains, but even an imperfect approximation of this process gives a useful measure of semantic information. For example, cyclomatic complexity [125] of a software module is calculated from a connected graph showing topology of control flow within the program as $CC = E - N + p$, where E is the number of edges of the graph, N the number of nodes of the graph and p the number of connected components. Software with cyclomatic complexity over 50 is considered untestable and very high risk. Adding new module to such software leads to a large change in the amount of information that the system gains, making it hard to predict all the consequences.

The use of algorithmic information measures in computational intelligence is still rare. However, CI systems that encode new information in terms of the known information are certainly not new. They include constructive neural networks that add new nodes only if the current approximation is not sufficient [85, 50], similarity-based systems that accept new reference vector checking first if it is not redundant, decision trees that are pruned to increase their generalization and ignore data that are already correctly handled, information selection methods that increases the pool of features or their combinations only when new data justifies it, and many other approaches.

2.3 Meta-learning via search in the model space

Meta-learning requires detailed characterization of the space of possible models, the ability to create and modify CI models in a systematic way. This should not be done in a random way, as the space of all possible models is too large to explore. Some framework that covers large number of different methods is needed. For feedforward neural networks such framework involves possible architectures, from simplest to more complex, and a taxonomy of different types of transfer functions [57, 58], allowing for systematic exploration of different network models. Although the focus of neural network community has been on learning algorithms and network architectures, it is clear that selection of transfer functions is decisive for the speed of convergence in approximation and classification problems. As already shown in [57] (see also [11, 12]) some problems may require $O(n^2)$ parameters using localized functions and only $O(n)$ parameters when non-local functions are used. The n-parity problem may be trivially solved using a periodic function with a single parameter [40] while the multilayer perceptron (MLP) networks need $O(n^2)$ parameters and learn it only with great difficulty. It is hard to create basis functions expansion that will not have the universal approximator property, yet the fact that MLPs and radial basis function (RBF) networks are universal approximators has somehow blinded the researches who ignored quite obvious fact that it is the speed of convergence (especially for multidimensional data) that is most important. In principle neural networks may learn any mappings, but the ability to learn quickly and accurately requires flexible "brain modules", or transfer functions that are appropriate for the problem to be solved. This problem will not disappear thanks to better learning procedures or architectures, the main research topics in neural computing.

Meta-learning methods should help to build the final model from components performing different transformations on available data. Almost all adaptive systems are homogenous, i.e. they are built from many processing elements of the same type. MLP

neural networks and decision trees use nodes that partition the input space by hyperplanes. Networks based on localized functions (RBF, Gaussian classifiers) frequently use nodes that provide spherical or ellipsoidal decision borders. This cannot be the best inductive bias for all data, frequently requiring large number of processing elements even in cases when simple solutions exist. Neurocognitive inspirations that go beyond simple neurons may point the way here. A single cortical column in the brain provides many types of microcircuits that respond in a qualitatively different way to the incoming signals [121]. Other cortical columns may combine these responses in a perceptron-like fashion to enable complex discriminations. At the level of higher cognition brains do not recognize all objects in the same feature space. Even within the same sensory modality a small subset of complex features is selected, allowing to distinguish one class of objects from another (for example, in case of vision a simple sketch is sufficient). Object recognition or category assignment by the brain is probably based on evaluation of similarity to memorized prototypes of objects using a few characteristic features.

In contrast to human categorization most pattern recognition systems implicitly assume that classification is done using the same features in all regions of the input space. Memory-based techniques use single distance (or similarity) function to distinguish all objects, statistical or neural methods provide hyperplanes (MLPs) or Gaussian functions (RBF networks) for discrimination, but rarely both. Decision trees are usually univariate, employing a decision rule for the threshold value of a single feature, partitioning the input space into hyperrectangles. Multivariate decision trees provide several hyperplanes at high computational cost. Support Vector Machines use one kernel globally optimized for a given dataset [27]. All these systems may be called "homogenous" since they search for a solution providing the same type of elements, the same type of decision borders in the whole feature space. Committees of the homogenous systems are frequently used to improve and stabilize results [17]. Combining systems of different types in a committee is a step towards heterogeneous systems that use different types of decision borders, but such models may become quite complex and difficult to understand.

A rather obvious extension of traditional approach is to use class-specific features that may be quite distinct for different classes. This approach is used in an implicit way in feedforward neural networks with strong regularization that leads to creation of hidden nodes strongly linked only to selected features and used for specific classes. This type of solutions may also emerge in hierarchical systems, such as decision trees, where each class may end in a different branch using different features, although at least one feature is always shared among all classes. Using feature selection for separate classifiers $C_k(\mathbf{X})$ that distinguish a single class from the rest may lead to completely distinct sets of features. In the K-class problem a set of $i = 1..K$ selector transformations $\mathbf{Z}_i = \mathcal{T}_i(\mathbf{X}_i)$ may be defined by feature selection techniques, and these classifiers are restricted to their own $C_k(\mathbf{Z}_k)$ subspaces. The final decision depends on the relative confidence of individual classifiers that may assign the same vector to the class they specialize in. Even though individual classifiers are trained on different feature subsets their confidence may be scaled by a second transformation, such as additional scaling or a linear mixture of their predictions. Alternatively, new adaptive transformations may, trained on the K-dimensional vectors ${}^1\mathbf{X} = C_k(\mathbf{X})$ obtained as predictions for all

training data. In this case the classifiers are used as data transformations and their number may be larger than K. Binary classifiers (such as decision trees) usually give only one answer $C_k(\mathbf{X}) = 1$, but some other classifiers may actually create K probabilities, so the final dimensionality after this transformation may reach at least K^2.

A more sophisticated approach to class-specific use of features has been presented by Baggenstoss [8]. It is based on estimation of probability density functions (PDFs) in the reduced low-dimensional feature space selected separately for each class, and mapping these PDFs back to the original input space, where Bayesian classification is performed. To constrain the inverse mapping (it is obviously not unique) a reference hypothesis is used for which $\mathcal{P}(\mathbf{X}|H_0)$ and $\mathcal{P}(\mathbf{Z}|H_0)$ are both known, and likelihood rations are preserved, that is:

$$\mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{Z})\mathcal{P}(\mathbf{X}|H_0)/\mathcal{P}(\mathbf{Z}|H_0) \tag{1}$$

The "correction factor" to the PDF calculated in $\mathbf{Z}$ space is simply the ratio of the two reference hypothesis PDFs. The PDF projection theorem open may possibilities worth exploration, although the choice of the reference hypothesis may sometimes be non-trivial.

2.4 Similarity-based framework for meta-learning

Similarity (or dissimilarity, measured by some distance) is a very fundamental concept that can be used as a basis for computational intelligence methods [138]. For additive similarity measures models based on similarity to prototypes are equivalent to models based on fuzzy rules and membership functions [48]. Similarity functions may be related to distance functions by many transformations, for example exponential transformation $S(\mathbf{X},\mathbf{Y}) = \exp(-D(\mathbf{X},\mathbf{Y}))$. Additive distance functions $D(\mathbf{X},\mathbf{Y})$ are then converted to the multiplicative similarity factors (membership functions). For example, Euclidean distance function $D_2(\mathbf{X},\mathbf{Y})^2 = \sum_i W_i(X_i - Y_i)^2$ is equivalent to a multivariate Gaussian similarity function $S_2(\mathbf{X},\mathbf{Y}) = \exp(-D_2(\mathbf{X},\mathbf{Y})^2)$ centered at $\mathbf{Y}$ with ellipsoidal contours of constant values $D_2(\mathbf{X},\mathbf{Y})$ =const, equal to the product of univariate Gaussian membership functions $S_2(\mathbf{X},\mathbf{Y}) = \prod_i G(X_i,Y_i) = \prod_i \exp[-W_i(X_i-Y_i)^2]$. Using such transformations fuzzy rules (F-rules) with product norms may always be replaced by prototype-based rules (P-rules) with appropriate similarity functions. On the other hand all additive distance functions may be replaced by product T-norms with membership functions given by exponential one-dimensional distance factors. For example, the Manhattan distance function $D_1(\mathbf{X},\mathbf{P}) = \sum_{i=1} |X_i - P_i|$ leads to a product of $\exp(-|X_i-P_i|)$ membership functions. However, non-additive distance functions (for example the Mahalanobis distance) are difficult to approximate using products or combinations of one-dimensional fuzzy membership functions, unless explicit correlations between fuzzy features are taken into account.

Prototype-based rules (P-rules), although rarely used, are in many cases easier to comprehend than fuzzy rules (F-rules) and create new possibilities for data understanding. Relations between these two types of rules have so far not been explored in details. Two types of prototype-based rules may be distinguished: minimum distance rules (discriminative approach), and the threshold rules (similarity-based approach). Minimum

distance rule selects the prototype that is closer:
IF $P = \arg\min_{\mathbf{P}'} D(\mathbf{X}, \mathbf{P}')$ THAN Class($\mathbf{X}$) = Class($\mathbf{P}$),
while threshold rules select the prototype that is sufficiently similar, or closer than some threshold distance:
IF $D(\mathbf{X}, \mathbf{P}) \leq d_P$ THEN C.

The minimum distance rule for two prototypes defines a hyperplane bisecting the line connecting these prototypes. There are many methods to find optimal hyperplanes [63, 85] and they can be used to select optimal position of prototypes. On the other hand variants of LVQ methods [106] lead to prototypes and thus hyperplanes that do not seem to correspond to any discriminant analysis algorithms. In particular the idea of maximizing margins, not only minimizing errors, used in SVM algorithms based on solutions to regularized least square problem in the primial or dual space [21] , has not yet been used for prototype optimization or selection [95, 79]. Any hyperplane defined by its bias and normal vector $(W_0, \mathbf{W})$ is equivalent to a minimal distance rule for two prototypes $\mathbf{P}, \mathbf{P}'$ such that $\mathbf{W}/||\mathbf{W}|| = (\mathbf{P} - \mathbf{P}')/||\mathbf{P} - \mathbf{P}'||$, and $W_0 = \frac{1}{2}||\mathbf{P} - \mathbf{P}'||$. Thus discrimination hyperplanes do not specify by themselves interesting prototypes, they can move in the subspace parallel to the $\mathbf{W}$ plane, and be placed in important positions, for example close to cluster centers.

Decision boundaries of the threshold rules depend on the type of the distance function $D(\mathbf{X}, \mathbf{P})$. They frequently provide localized decision regions and may not be able to label vectors in some areas of feature space, unless a default (ELSE) rule is used. Distance function $D_{\mathbf{W}}(\mathbf{X}, \mathbf{P})$ between prototype $\mathbf{P}$ and point $\mathbf{X}$ that has constant value for all points lying on a plane perpendicular to $\mathbf{W}$ is calculated by:

$$D_{\mathbf{W}}(\mathbf{X}, \mathbf{P}) = |\sum_{i}^{N} s_i(\mathbf{X}_i - \mathbf{P}_i)|; \quad s_i = \mathbf{W}_i/||\mathbf{W}|| \tag{2}$$

makes the threshold rule IF $D(\mathbf{X}, \mathbf{P}) \leq d_P$ THEN C, with $d_P = \frac{1}{2}||\mathbf{P} - \mathbf{P}'||$ equivalent to the minimum distance rule for propotypes $\mathbf{P}, \mathbf{P}'$. Relations between various discriminant analysis techniques on the one hand, and optimization of prototypes on the other hand, have just started to be explored. Relations between similarity based methods and fuzzy rule-based systems have also not yet been analyzed in depth. More attention has been devoted to relations between RBF networks and fuzzy logic models [103].

RBF and MLP networks may be viewed as a particular implementation of hierarchical sets of fuzzy threshold logic rules based on sigmoidal membership functions, equivalent to crisp logic networks applied to the input data with uncertainty [37]. Leaving uncertainty (fuzziness) on the input side makes the networks or the logical rule-based systems easier to understand, and achieves similar results as the Type-2 fuzzy systems [127]. Moreover, it shows deep connections between neural and fuzzy systems, with different types of input uncertainty equivalent to crisp input values with specific transfer functions. Many natural assumptions about uncertainty of input variable x lead to probability that rule $\Pr(x > \theta)$ is true given by the membership functions of sigmoidal shape, for example semi-linear functions for uncertainties that are constant in some interval or to erf functions (almost identical to logistic functions) for Gaussian uncertainties.

The radial basis functions became a synonym for all basis function expansions, although in approximation theory already in the classical book of Achieser published in 1956 [2] many such expansions were considered. RBF networks are equivalent to the fuzzy systems only in special cases, for example when the Gaussian membership functions are used [103], but in the literature RBF is frequently used as a synonym for Gaussian node networks, although any functions that depends only on Euclidean distance, or a radial variable $\phi(r) = \phi(||\mathbf{X} - \mathbf{R}||)$, is suitable for RBF expansion. However, it is not clear that a radial dependence is always the best assumption for a given data. Another useful category of basis set expansion methods uses separable basis functions (SBF), where each node implements a product of one-dimensional functions $\phi(\mathbf{X}) = \prod_i f_i(X_i)$. Approximation abilities of SBF networks are similar to those of RBF networks, and the separable function realized by their nodes may be interpreted in a natural way as the product of fuzzy membership functions. They may also form a natural generalization of the Naive Bayes (NB) approach, with each network node implementing a local NB model, and the whole network functioning as a committee of such models, aggregating evidence for each class and using some voting procedure or final decision [110]. It is not clear why so much research has been devoted to the RBF networks while neural networks based on separable functions are virtually unknown: the Feature Space Mapping (FSM) network [50, 60, 3, 44] seems to be the only existing implementation of the Separable Basis Function networks so far.

A general framework for similarity-based methods (SBMs) has been formulated using the concept of similarity [43, 35]. This framework includes typical feedforward neural network models (MLP, RBF, SBF) , some novel networks (Distance-Based Multilayer Perceptrons (D-MLPs, [41]) and the nearest neighbor or minimum-distance networks [33, 43]), as well as many variants of the nearest neighbor methods, improving upon the traditional approach by providing more flexible decision borders. This framework has been designed to enable meta-learning based on a search in the space of all possible models that may be systematically constructed. New algorithms are generated by applying admissible extensions to the existing algorithms and the most promising are retained and extended further. Training is performed using parameter optimization techniques [52, 53]. Symbolic values used with probabilistic distance functions allow to avoid ad hoc procedure to replace them with numerical values. To understand the structure of the data prototype-based interpretation of the results is used, simplifying predictive models and providing prototype rules (P-rules) that may be converted to fuzzy rules (F-rules) [48].

In the SBM approach objects (samples, cases, states) $\{\mathbf{O}^i\}$, $i = 1 \ldots n$ may be represented by a set of numerical or symbolic features $X_j^i = X_j(\mathbf{O}^i), j = 1...N$ characterizing these objects, or by a procedure that evaluates directly similarity $D(\mathbf{O}^i, \mathbf{O}^k)$ between objects, allowing for comparison of objects with complex structure. For classification problems a function or a procedure to estimate $p(C_i|\mathbf{X}; M), i = 1..K$, the posterior probability of assigning vector $\mathbf{X}$ to a class C_i, is defined, while for approximation problems a function $Y(\mathbf{X}; M)$ is defined. In both cases the model M includes various procedures, parameters and optimization methods. A general similarity-based model for classification problems is constructed from the following components:

– input pre-processing transformation, either providing directly dissimilarities of objects $D(\mathbf{O}^i, \mathbf{O}^k)$ or mapping them to symbolic/numerical descriptions $\mathbf{X}(\mathbf{O})$ that define the feature space; in both cases a data matrix is obtained;
– a procedure to select relevant features or their combinations;
– function $d_j(X_j; Y_j)$ to calculate similarity of $X_j,\ Y_j$ feature values, $j = 1..N$;
– function $D(\mathbf{X}, \mathbf{Y}) = D(\{d_j(X_j; Y_j)\})$ that combines similarities defined for each attribute to compute similarities of vectors;
– specification of the neighborhoods, such as the number of reference vectors k taken into account around of $\mathbf{X}$, or the number of hidden nodes in feedforward neural networks based on functions with localized support in the feature space;
– the weighting function $G(D) = G(D(\mathbf{X}, \mathbf{R}))$ estimating contribution of each reference vector $\mathbf{R}$;
– a set of prototype (reference) vectors $\{\mathbf{R}\}$ selected from the set of training vectors $\{\mathbf{X}^i\}$ and optimized using some procedures;
– a set of class probabilities or membership values $p_i(\mathbf{R}), i = 1 \ldots K$ for each reference vector;
– a misclassification risk matrix $\mathcal{R}(C_i, C_j), i, j = 1 \ldots K$;
– a scaling function $K(\cdot)$ estimating the influence of the error, for a given training example, on the total cost function;
– a function (or a matrix) $\mathcal{S}(\cdot, \cdot)$ evaluating similarity (or more frequently dissimilarity) of the classes; if class labels are soft, or if they are given by a vector of probabilities $p_i(\mathbf{X})$, classification task is in fact a mapping;
– a total cost function $E[\mathcal{D}_T; M]$ that is minimized at the training stage directly, in a crossvalidation, bootstrap or other procedure;
– procedures to optimize parameters of the model at each stage.

This framework may be used to automatically generate a sequence of models with growing complexity. It has been used in a meta-learning scheme to solve classification problems [53], starting from the simplest k-NN model parameterized by $M = \{k, D(\cdot, \cdot), \{\mathbf{X}\}\})$, i.e. the whole training dataset used as the reference set, k nearest prototypes included with the same weight, using a typical distance function, such as the Euclidean or the Manhattan distance. Probabilities are computed as $p(C_i|\mathbf{X}; M) = N_i/k$, where N_i is the number of nearest vectors that belong to class C_i. The initially model has thus only one parameter k for optimization. If such model is not sufficiently accurate new procedures/parameters are added to create a more complex model. The search tree in the space of all models is generated by extending current model in the simplest way, using a measure that penalizes for increased model complexity and rewards for increased accuracy. Various model selection criteria may be applied to control this process [174]. Several good models are maintained in a beam search procedure, and the search stops when additional complexity of possible extensions of existing models does not justify increased accuracy.

Scalar products or cosine distances are a particular way to measure similarity, quite useful especially when vectors have significantly different norms (as in the evaluation of text similarity [123]). Similarity may be evaluated in a more sophisticated way by learning [112] and designing various kernels that evaluate similarity between complex objects [27, 155]. Although kernels are usually designed for SVM methods, for example in text classification [118] or bioinformatics [171, 7, 169] applications, they may be

directly used in the SBM framework, because kernels are specific (dis)similarity functions. In particular positive semidefinitive kernels used in SVM approach correspond to some Euclidean dissimilarity matrices [138]. SVM is just one particular method of a single hyperplane optimization in a space where kernel $K(\mathbf{X}, \mathbf{Y})$ serves as a scalar product. However, for data models requiring different resolutions in different areas of feature spaces SVM may not be the best approach, and a combination of kernels for feature space expansion, easily accommodated in the SBM framework, should be used. Meta-learning is not a single method, but an approach that leads to a tailor-made methods created on demand, therefore it may to a large degree avoid the "no-free-lunch" theorem restrictions [174]. SVM models are certainly an important family among a large number of models that may be explored during metalearning.

3 Transformation-based CI theory

Similarity-based methods using only direct similarity comparisons of some vector representations are restricted to a single transformation (not counting pre-processing), $\mathbf{Y} = \mathcal{T}(\mathbf{X}; \mathbf{R})$. Although in principle this is sufficient for universal approximation [83] in practice it may slow down the convergence and make a discovery of simple data models very difficult. SBM framework is generalized here to include multiple transformation steps. In the stacking approach [183, 161, 157] one classifier produces outputs that are used as inputs for another classifier. Wolpert showed [183] that biases of stacked classifiers may be deducted step by step, improving generalization of the final system. The same is true if a series of transformations is composed together to produce a data model. Transformation-based approach fits very well to modern component-based data mining and may be presented in from of graphical models, although quite different than probabilistic models presented in [99].

General similarity transformations may act on objects and produce vectors, either by analysis of object properties or object similarity to some reference objects. In both cases feature-based vector description of the object is produced $\mathbf{X}$, although the size of this vector may be different. In the first case the number of features N is independent of the number of objects, while in the second case all training objects may be used as a reference and $\mathbf{X}_i = K(\mathbf{O}, \mathbf{O}_i)$ feature values calculated using kernel $K(\cdot, \cdot)$ function (or procedure) to provide n-dimensional vector.

CI calculations may be presented as a series of transformations, divided into several stages. Starting from the raw input data ${}^0\mathbf{X} = \mathbf{X}$ that defines initial feature space, first transformation $\mathcal{T}_1$ scales individual features, filters them, combining pairs or small subsets of features. This leads to a new dataset ${}^1\mathbf{X} = \mathcal{T}_1({}^0\mathbf{X})$ with vectors based on a new set of features that may have different dimensionality than the original data. The second transformation ${}^2\mathbf{X} = \mathcal{T}_2({}^1\mathbf{X})$ extracts multidimensional information from pre-processed features ${}^1\mathbf{X}$. This is further analyzed by subsequent transformations that either aim at separation of the data or at mapping to a specific structures that can be easily recognized by the final transformation. The final transformation provides desired information. These transformations can in most cases be presented in a layered, graphical form.

Chains of transformations created on demand should lead to optimal CI methods for a given data. This requires characterization of elementary transformations. The goal here is to describe all CI methods as a series of transformations but at a higher level than the psuedocode. Several types of vector transformations should be considered: component, selector, linear combinations and non-linear functions. Component transformations work on each vector component separately, for example shifting and scaling component values when calculating distances or scalar products. Selector transformations define subsets of vectors or subsets of features using various criteria for information selection, or similarity to the known cases (nearest neighbors), or distribution of feature values and class labels. Non-linear functions may serve as kernels or as neural transfer functions [57]. Transformations composed from these elementary types may always be presented in a network form.

3.1 Extracting information from single or small subsets of features

New information from available features may be extracted using various types of network nodes in several ways. First, by providing diverse basis functions or receptive fields for sampling the data separately in each dimension, although two or higher-dimensional receptive fields may also be used in some applications (as is the case for image or signal processing filters, such as wavelets). Fuzzy and neurofuzzy systems usually include a "fuzzification step", defining for each feature several membership functions $\mu_k(X_i)$ that act as large receptive fields. Projecting each feature value X_i on these receptive fields μ_k increases the dimensionality of the original data. This may lead to some improvement of results as even a linear transformation in the extended space has a greater chance to separate data better.

Second, information may be extracted by scaling the features using logarithmic, sigmoidal, exponential, polynomial and other simple functions; such transformations help to make the density of points in one dimension more uniform, circumventing some problems that would require multiresolution algorithms. Although they are rarely mentioned as a part of the learning algorithms adaptive preprocessing at this stage may have a critical influence on the final data model.

Third, information is extracted by creating new features using linear combinations, tensor products, ratios, periodic functions or using similarity measures on subsets of input variables. Non-linear feature transformations, such as tensor products of features, are particularly useful, as Pao has already noted introducing functional link networks [137, 1]. Rational function neural networks [85] in signal processing [115] and other applications use ratios of polynomial combinations of features; a linear dependence on a ratio $y = x_1/x_2$ is not easy to approximate if the two features x_1, x_2 are used directly. Groups of several strongly correlated features may be replaced by a single combination performing principal component analysis (PCA) restricted to small subspaces. To decide which groups should be combined standardized Pearson's linear correlation is calculated:

$$r_{ij} = 1 - \frac{|C_{ij}|}{\sigma_i \sigma_j} \in [-1, +1] \tag{3}$$

where the covariance matrix is:

$$C_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} \left(X_i^{(k)} - \bar{X}_i \right) \left(X_j^{(k)} - \bar{X}_j \right); \quad i,j = 1 \cdots d \tag{4}$$

These coefficient may be clustered, for example by using dendrogram techniques. Depending on the clustering thresholds they provide reduced number of features, but also features at different scales, from a combination of a few features to a global PCA combinations of all features. This approach may help to discover hierarchical sets of features that are useful in problems requiring multiscale analysis. Another way to obtain features for multiscale problems is to do clusterization in the data space and make local PCA within the clusters to find features that are most useful in various areas of space.

Linear combinations derived from interesting projection directions may provide low number of interesting features, but in some applications non-linear processing is essential. The number of possible transformations at this stage is very large. Feature selection techniques [82], and in particular filter methods wrapped around algorithms that search for interesting feature transformations (called "filtrappers" in [39]), may be used to quickly evaluate the usefulness of proposed transformations. The challenge is to provide a single framework for systematic selection and creation of interesting transformations in a meta-learning scheme. Evolutionary techniques may prove to be quite useful in this area.

3.2 Extracting information from all features

After transforming individual features or small groups of features to create ${}^1\mathbf{X}$ space additional transformations that involve all features are considered. Frequently these transformations are used to reduce dimensionality of the data. Srivastava and Liu [162] point out that the choice of optimal transformation depends on the application and the data set. They have presented an elegant geometrical formulation using Stiefel and Grassmann manifolds, providing a family of algorithms for finding orthogonal linear transformations of features that are optimal for specific tasks and specific datasets. They find PCA to be optimal transformation for image reconstruction under mean-squared error, Fisher discriminant for classification using linear discrimination, ICA for signal extraction from a mixture using independence, optimal linear transformation of distances for the nearest neighbor rule in appearance-based recognition of objects, transformations for optimal generalization (maximization of margin), sparse representations of natural images and retrieval of images from a large database. In all these applications optimal transformations are different and may be found by defining appropriate cost functions and optimizing them using stochastic gradient techniques. Some of their cost functions may be difficult to optimize and it is not yet clear that sophisticated techniques based on differential geometry, advocated in [162], offer significant practical advantages, although they certainly provide an interesting insight into the problem. Simpler learning algorithms based on numerical gradient techniques and systematic search algorithms give surprisingly good results and can be applied to optimization of difficult functions [108], but a detailed comparison of such methods has not yet been made.

In some cases instead of reduction of dimensionality expansion of the feature space may be useful. Random linear projection of input vectors into a high-dimensional space ${}^2\mathbf{X} = \mathbf{L}({}^1\mathbf{X})$ is the simplest expansion, with the random matrix $\mathbf{L}$ that has more rows than columns (see neurobiological justification of such projections in [121]). If highly nonlinear low-dimensional decision borders are needed large number of neurons should be used in the hidden layer, providing linear projection into high-dimensional space followed by filtering through neural transfer functions to normalize the output from this transformation. Enlarging the data dimensionality increases the chance to make the data separable, and this is frequently the goal of this transformation, ${}^2\mathbf{X} = \mathcal{T}_2({}^1\mathbf{X}; {}^1\mathbf{W})$. If (near) separability can be achieved this way the final transformation may be linear $\mathbf{Y} = \mathcal{T}_3({}^2\mathbf{X}; {}^2\mathbf{W}) = \mathbf{W}_2 \cdot {}^2\mathbf{X}$. A combination of random projection and linear model may work well [89] – this is basically achieved by random initialization of feedforward neural networks and a linear discriminant (LDA) solution for the output model, a method used to start a two-phase RBF learning [156]. However, one should always check whether such transformation is really justified from the point of view of model complexity, because linear discrimination may work also quite well for many datasets in the original feature space, and many non-random ways to create interesting features may give better results. It may also be worthwhile to add pre-processed ${}^1\mathbf{X} = \mathcal{T}_1(\mathbf{X})$ features to the new features generated by the second transformation ${}^2\mathbf{X} = ({}^1\mathbf{X}, \mathcal{T}_2({}^1\mathbf{X}; {}^1\mathbf{W}))$, because they are easier to interpret and frequently contain useful information – in case of linear transformations this simply adds a diagonal part to the matrix.

In general the higher the dimensionality of the transformed space the greater the chance that the data may be separated by a hyperplane [85]. One popular way of creating highly-dimensional representations without increasing computational costs is by using the kernel trick [155]. Although this problem is usually presented in the dual space the solution in the primal space is conceptually simpler [113, 21]. Regularized linear discriminant (LDA) solution is found in the new feature space ${}^2X = \mathbf{K}(\mathbf{X}) = K({}^1\mathbf{X}, \mathbf{X})$, mapping $\mathbf{X}$ using kernel functions for each training vector. Feature selection techniques may be used to leave only components corresponding to "support vectors' that provide essential support for classification, for example only those close to the decision borders or those close to the centers of cluster, depending on the type of the problem. Any CI method may be used in the kernel-based feature space $K(\mathbf{X})$, although if the dimensionality is large data overfitting is a big danger, therefore only the simplest and most robust models should be used. SVM solution to use LDA with margin maximization is certainly a good strategy.

Consider for example a two-class case. In m-dimensional space the expected maximum number of separable vectors randomly assigned to one of the classes is $2m$ [24, 85]. For k-bit strings there are $n = 2^k$ vectors and 2^n Boolean functions that may be separated in the space with $n/2$ dimensions with high probability. In case of k-bit Boolean problems localized kernels are not useful as the number of vectors n grows exponentially fast with k, but separation of all vectors is possible in the space generated by polynomial kernel of degree k, providing new features based on $n-1$ monomials $x_a, x_a x_b, x_a x_b x_c, \ldots, x_1 \ldots x_k$. Indeed SVM algorithms with such kernel are capable of learning all Boolean functions although they do not generalize well, treating each

vector as a separate case. For example, SVM with polynomial kernel of degree k (or with a Gaussian kernel) may solve the k-bit parity problem. However, removing a single string from the training set in the leave-one-out test will lead to perfect learning of the training sets, but always wrong predictions of the test vector, thus achieving 0% accuracy! Unfortunately only for parity the answer will always be wrong (each vector is surrounded by the nearest neighbors from the opposite class), for other Boolean functions one cannot count on it. This problem may be solved in a simple way in the original feature space if the goal of learning is redefined (see k-separability section below).

If the final transformation is linear $\mathbf{Y} = {}^3\mathbf{X} = \mathcal{T}_3({}^2\mathbf{X}; {}^2\mathbf{W})$ parameters ${}^2\mathbf{W}$ are either determined in an iterative procedure simultaneously with parameters ${}^1\mathbf{W}$ from previous transformations (as in the backpropagation algorithms [85]), or they may be sequentially determined by calculating the pseudoinverse transformation, as is frequently practiced in the two-phase RBF learning [156], although in experiments on more demanding data simultaneous adaptation of all parameters (in RBF networks they include centers, scaling parameters, and output layer weights) gives better results. The initial random transformation may use arbitrary basis functions, although for localized functions simple clustering instead of a random projection is recommended. Most basis function networks provide receptive fields in the subspace of the original feature space or on the pre-processed input data. Transformations of this kind may be presented as a layer of network nodes that perform vector mapping $\mathcal{T}_2({}^1\mathbf{X}; {}^1\mathbf{W})$ based on some specific criterion. Many interesting mappings are linear and define transformations equivalent to those provided by the Exploratory Projection Pursuit Networks (EPPNs) [98, 66]. Quadratic cost functions used for optimization of linear transformations may lead to formulation of the problem in terms of linear equations, but most cost functions or optimization criteria are non-linear even for linear transformations. A few such transformations are listed below:

- Statistical tests for dependency between class and feature value distributions, such has Pearson's correlation coefficient, χ^2 and other measures that may be used to determine best orthogonal combination of features in each node.
- Principal Component Analysis (PCA) in its many variants, with each node computing principal component [63, 174, 164]).
- Linear Discriminatory Analysis (LDA), with each node computing LDA direction (using one of the numerous LDA algorithms [63, 174, 164]).
- Fisher Discriminatory Analysis (FDA), with each node computing canonical component using one of many FDA algorithms [174, 165].
- Independent Component Analysis, with each node computing one independent component [90, 23].
- Linear factor analysis, computing common and unique factors from data [75].
- Canonical correlation analysis [70].
- KL, or Kullback-Leibler networks with orthogonal or non-orthogonal components; networks maximizing mutual information [168] are a special case here, with product vs. joint distribution of classes/feature values.
- Classical scaling, or linear transformation embedding input vectors in a space where distances are preserved [138].
- Linear approximations to multidimensional scaling [138].

– Separability criterion used on orthogonalized data [76].

Non-linearities may be introduced in transformations in several ways: either by adding non-linear functions to linear combinations of features, or using distance functions, or transforming components and combining results [57, 58]. Linear transformations in kernel space are equivalent to non-linear transformations in the original feature space. A few non-linear transformations are listed below:

– Kernel versions of linear transformations, including radial and other basis set expansion methods [155].
– Weighted distance-based transformations, a special case of general kernel transformations, that use (optimized) reference vectors [43].
– Perceptron nodes based on sigmoidal functions with scalar product or distance-based activations [42, 41], as in layers of MLP networks, but with targets specified by some criterion (any criterion used for linear transformations is sufficient).
– Heterogeneous transformations using several types of kernels to capture details at different resolution [57].
– Heterogeneous nodes based or several type of non-linear functions to achieve multiresolution transformations [57].
– KL, or Kullback-Leibler networks with orthogonal or non-orthogonal components; networks maximizing mutual information [168] are a special case here, with product vs. joint distribution of classes/feature values.
– χ^2 and other statistical tests for dependency to determine best combination of features.
– Factor analysis, computing common and unique factors, reducing noise in the data.
– Nodes implementing fuzzy separable functions, or other fuzzy functions [50].

Many other transformations of this sort are known and may be used at this stage in transformation-based systems. A necessary step for meta-learning is to create taxonomy, similarities and relations among such transformations to enable systematic search in the space of possible models, but this has not yet been done. An obvious division is between fixed transformations that are based on local criterion, with well-defined targets, and adaptive transformations that are based on criterions optimizing several steps simultaneously (as in backpropagation), where the targets are defined only for composition of transformations. Fixed $\mathcal{T}_2$ transformations have coefficients calculated directly from the input data or data after $\mathcal{T}_1$ transformation. Activity of the network nodes has then clear interpretation, and the number of nodes may be determined from estimation of such criteria as the information gain compared to the increased complexity. A general way to calculate fixed transformation coefficients is to create a single component (for example, one LDA hyperplane), and then orthogonalize the input vectors to this component, repeating the process in an iterative way. General projection pursuit transformations [98, 66] may provide a framework for various criteria used in fixed transformations.

Transformations may also have adaptive coefficients, determined either by an optimization procedure for the whole system (for example, global optimization or backpropagation of errors), or by certain targets set for this mapping (see the k-separability

section). The interpretation of node functions is not so clear as for the fixed targets for individual transformations, but the final results may be better. Fixed transformations may be very useful for initialization of adaptive transformations or may be useful to find better solutions of more complex fixed transformations. For example, multidimensional scaling requires very difficult minimization and seems most of the time to converge to a better solution is PCA transformations is performed first. Nonlinear Mapping Pursuit Networks (MPN), similar to EPPNs, may be defined and used as a fixed transformation layer, followed by linear model in the same way as it is done in the functional link networks [137].

Adding more transformation layers with distance-based conditions, that is using similarity in the space created by evaluation of similarity in the original input space, leads to higher-order nearest neighbor methods, rather unexplored area. There are many other possibilities. For example, consider a parity-like problem with vectors from the same class that are spread far apart and surrounded by vectors from other classes [40]. The number of nodes covering such data using localized functions will be proportional to the number of vectors. Such transformations will not be able to generalize. Kernel methods based on localized or polynomial kernels will also not be able to generalize. MLP networks may solve the problem but need architectures specially designed for each problem of this type and are hard to train. Linear projections may provide interesting views on such data, but the number of directions $\mathbf{W}$ that should be considered to find good projections grows exponentially fast with the number of input dimensions (bits). In case of n-bit parity projection $Y = \mathbf{W} \cdot \mathbf{X}$ counts the number of 1 bits, producing odd and even numbers for the two parity classes. A periodic functions (such as cosine) is sufficient to solve the parity problem, but is not useful to handle other logical problems. Interesting transformations should find directions $\mathbf{W}$ that project a large number of training vectors $\mathbf{X}$ into localized groups. A window function $G(||\mathbf{W} \cdot \mathbf{X} - Y||)$ may capture a region where a delocalized large cluster of vectors from a single class is projected. A constructive network that adds new nodes to capture all interesting projections should be able to solve the problem. The linear output transformations will simply add outputs of nodes from all clusters that belong to a given class.

This type of geometric thinking leads to transformations that will be very useful in metalearning systems, facilitating learning of arbitrary Boole'an problems.

3.3 Heterogeneous adaptive systems

Many transformations may lead to the same goal because transformations with non-polynomial transfer functions are usually universal approximators [114]. The speed of convergence and the complexity of networks needed to solve a given problem is more interesting. Approximations of complex decision borders or approximations of multidimensional mappings by neural networks require flexibility that may be provided only by networks with sufficiently large number of parameters. This leads to the bias-variance dilemma [14, 63, 174], since large number of parameters contribute to a large variance of neural models and small number of parameters increase their bias.

Regularization techniques may help to avoid overparameterization and reduce variance, but training large networks has to be expensive. In MLP models regularization methods decrease the weights, forcing the network function to be more smooth. This is

a good bias for approximation of continuous functions, but it is not appropriate for data generated using logical rules, where sharp, rectangular decision borders are needed. Two orthogonal hyperplanes cannot be represented accurately with soft sigmoidal functions used in MLPs or with Gaussian functions used by RBF networks. The inadequacy of fundamental model limits the accuracy of neural solutions, and no improvement of the learning rule or network architecture will change this. Transformations based on scalar products (hyperplanes, delocalized decision borders) solve some problems with $O(N)$ parameters, while the use of localized functions (for example Gaussians) requires $O(N^2)$ parameters, while on other problems this situation is reversed [57]. Therefore discovering the proper bias for a given data is very important. Some real world examples showing the differences between RBF and MLP networks that are mainly due to the transfer functions used were presented in [57] and [46, 47].

The simplest transformation that has the chance to discover appropriate bias for complex data may require several different types of elementary functions. Heterogeneous adaptive systems (HAS) introduced in [51] provide different types of decision borders at each stage of building data model, enabling discovery of the most appropriate bias for the data. Neural [59, 94, 45], decision tree [51, 78] and similarity-based systems [53, 177, 178] of this sort have been described, finding for some data simplest and most accurate models known so far.

Heterogeneous neural algorithms that use several transfer functions within one network may be introduced in several ways. A constructive method that selects the most promising function from a pool of candidates adding new node to the transformation has been introduced in [45, 94, 59]. Other constructive algorithms, such as the cascade correlation [65], may also be used for this purpose. Each candidate node using different transfer function should be trained and the most useful candidate added to the network.

The second approach starts from transformation that uses many types of functions using information selection or regularization techniques to reduce the number of functions [94]. Initially the network may be too complex but at the end only the functions that are best suited to solve the problem are left. In the ontogenic approach neurons are removed and added during the learning process [94].

The third approach starts from flexible transfer functions that are parameterized in a way that makes them universal. Linear activation based on a scalar product $\mathbf{W} \cdot \mathbf{X}$ is combined with higher order terms used in distance measures to create functions that for some parameters are localized, and for other parameters non-localized. Several functions of such kind have been proposed in [57]. In particular bicentral functions are very useful and flexible, with decision regions of convex shapes, suitable for classification. These functions are product of N pairs of sigmoidal functions (Fig. 2):

$$\begin{aligned} Bi2s(\mathbf{X};\mathbf{t},\mathbf{B},\mathbf{s}) &= \prod_{i=1}^{N} \sigma(A2_i^+)\,(1-\sigma(A2_i^-)) \\ &= \prod_{i=1}^{N} \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i}))\,(1-\sigma(e^{s'_i}\cdot(x_i - t_i - e^{b_i}))) \end{aligned} \tag{5}$$

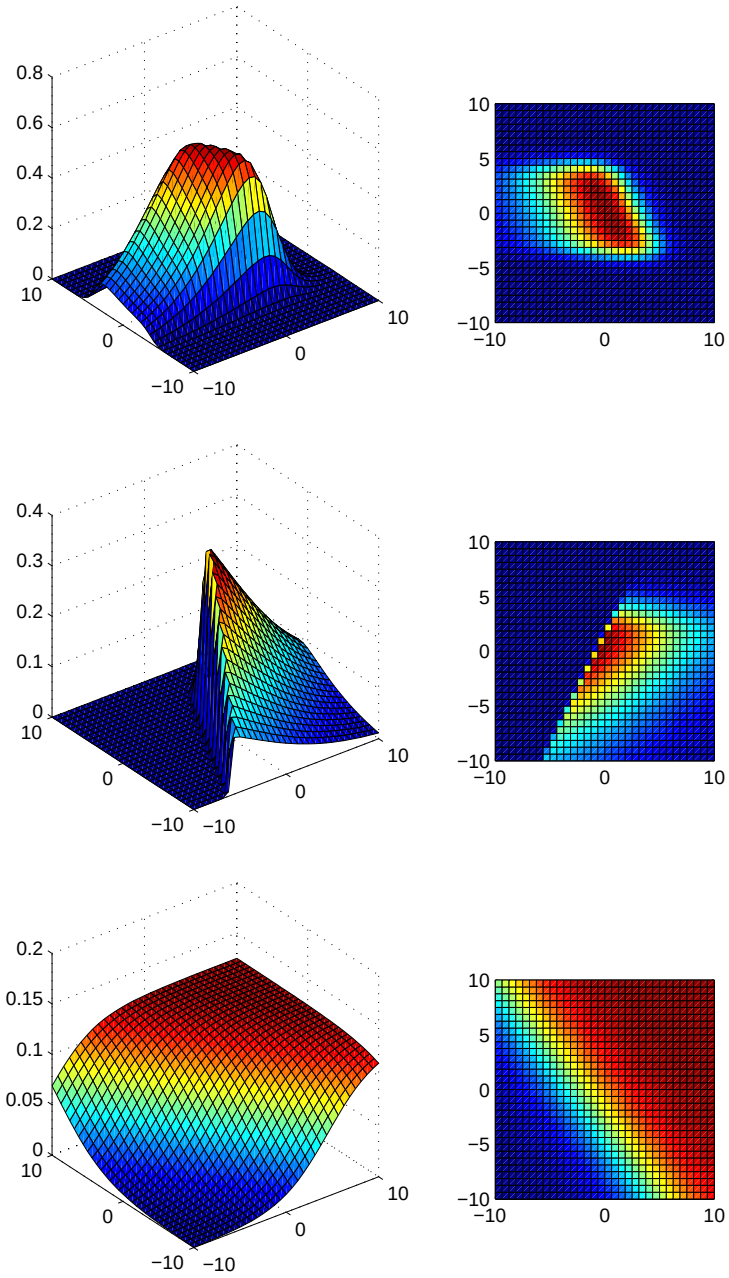

Fig. 2. A few shapes of general bicentral functions (Eq. 5).

The first sigmoidal factor in the product is growing for increasing input x_i while the second is decreasing, localizing the function around t_i. Shape adaptation of the density $Bi2s(\mathbf{X}; \mathbf{t}, \mathbf{B}, \mathbf{s})$ is possible by shifting centers $\mathbf{t}$, rescaling $\mathbf{B}$ and $\mathbf{s}$. Product form leads to well-localized convex contours of bicentral functions. Exponentials e^{s_i} and e^{b_i} are used instead of s_i and b_i parameters to prevent oscillations during the learning procedure (learning becomes more stable). Using small slope s_i and/or s_i' the bicentral function may delocalize or stretch to *left* and/or *right* in any dimension. This allows creation of such contours of transfer functions as half-infinite chanel, half-hyper ellipsoidal, soft triangular, etc.

Although the costs of using this function is a bit higher than of the bicentral function (each function requires $4N$ parameters) more flexible decision borders are produced. Rotations of these contours require additional N parameters. An important advantage of the bicentral functions comes from their separability, enabling analysis of each dimension or a subspace of the input data independently: one can forget some of the input features and work in the remaining subspace. This is very important in classification when some of the features are missing and allows to implement associative memories using feedforward networks [50, 3]. Bicentral functions with rotations (as well as multivariate Gaussian functions with rotation) have been implemented so far only in two neural network models, the Feature Space Mapping [50, 3] and the IncNet [100, 97, 96].

Very little experience with optimization of transfer functions in heterogenous systems has been gathered so far. Neural networks using different transfer functions should use lower number of nodes, and thus the function performed by the network may be more transparent. For example, one hyperplane may be used to divide the input space into two classes and one additional Gaussian function to account for local anomaly. Analysis of the mapping performed by an MLP network trained on the same data will not be so simple. More algorithms to create such models are needed.

3.4 Geometrical perspective

Composition of transformations may also be seen from geometrical perspective. Informational geometry [91] is aimed at characterization of the space of all possible probability distributions, and thus works in the space of model parameters. Geometry of heteroassociative vector transformations, from the input feature space to the output space, is also interesting. It is clear that different sensory signals are recognized in different feature subspaces, but even in a single sensory modality different objects are recognized paying attention to different features. This shows that vector space model for characterization of objects is too simple to account for object recognition in perception.

At each point of the input space relative importance of features may change. One way to implement this idea [35] is to create local non-symmetric similarity function $D(\mathbf{X} - \mathbf{Y}; \mathbf{X})$, smoothly changing between different regions of the input space. For example this may be a Minkovsky function $D(\mathbf{X} - \mathbf{Y}; \mathbf{X}) = \sum_i s_i(\mathbf{X})|X_j - Y_i)|$ with the scaling factor that depend on the point $\mathbf{X}$ of the input space, in particular many of them may be zero. Such scaling factors may be calculated for each training vector using local PCA, and interpolated between the vectors. Local Linear Embedding (LLE) is a popular method of this sort [148] and many other manifold learning methods have been

developed. Alternatively a smooth mapping may be generated training MLP or other neural networks to approximate desired scaling factors.

Prototype rules for data understanding and transformation may be created using geometrical learning techniques that construct a convex hull encompassing the data, for example an enclosing polytope, cylinder, a set of ellipsoids or some other surface enclosing the data points. Although geometrical algorithms may be different than neural or SVM algorithms, the decision surfaces they provide are similar to those offered by feedforward networks. A covering may be generated by a set of balls or ellipsoids following principal curve, for example using the piecewise linear skeletonization approximation to principal curves [104]. An algorithm of this type creating a "hypersausage" decision regions has been published recently [159]. More algorithms of this type should be developed, and their relations with neural algorithms investigated.

From geometrical perspective kernel transformations are capable of smoothing or flatting decision borders. Using the vectors $\mathbf{R}^{(i)}$ that are close to the decision border as support vectors for kernel (distance) calculation creates new features, placing support vectors on a hyperplane (distance for all $\mathbf{R}^{(i)}$ is zero). Therefore a single hyperplane after such transformation is frequently sufficient to achieve good separation of data. However, if the data has complex structure, disjoint clusters from the same class, or requires special transformation for extraction of information this may not be an optimal approach.

After the second transformation (or a series of transformations) all data is converted to the second internal representation ${}^2\mathbf{X}$, and the final transformation is added to extract simple structure from multidimensional data.

3.5 Redefining the goal of learning

The first two transformations should discover interesting structures in data or increase the chance of data separability, as in the case of kernel transformations. More transformations may be applied, either at the pre-processing stage (normalization, whitening, calculation of Fourier, Hadamard or wavelet coefficients etc), or at the information extraction stage. The role of the final transformations is to compress this information, find interesting views on the data from the point of view of certain goals. These transformations usually involves a drastic reduction of dimensionality. The number of outputs in the approximation problems is equal to the number of approximated components, or the problem is broken into several single-output functions. In the K-class classification problems the number of outputs is usually $K-1$, with zero output for the default class. In the Error Correcting Output Codes (ECOC) approach [32] learning targets that are easier to distinguish are defined, setting a number of binary targets that define a prototype "class signature" vectors. The final transformation compares then the distance from the actual output to these class prototypes.

The learning targets used in most CI methods for classification are aimed at linear separability. The final linear transformation provides a hyperplane that divides the data, transformed by a series of mappings $\mathcal{T}_k(..\mathcal{T}_2(\mathcal{T}_1(\mathbf{X})...))$, into two halfspaces. Linear transformation is the simplest and quite natural if the kernel transformation increases the dimensionality and flattens the decision borders, making the data linearly separable. However, for difficult problems, such as learning of Boolean functions, this will not

work. Instead of thinking about the decision hyperplane it is better to focus on interesting projections or more general transformations of data. For linearly separable data $\mathbf{W} \cdot \mathbf{X}$ projection creates two distinct clusters. For non-separable data an easier target is to obtain several well separated clusters. For example, in k-bit parity problem projection on $\mathbf{W} = [1, 1..1]$ shows $k + 1$ clusters clearly providing a satisfactory solution to the problem. Thus instead of aiming at linear separation using the final transformation based on hyperplane the goal of learning may be redefined by assuming another well-defined transformation. In particular using the interval-based transformation as the final step easily "disarms" the remaining non-linearity in data, and greatly simplifies the task of all previous transformations.

The dataset $\mathbf{X}_i$ of points belonging to two classes is called k-separability if a direction $\mathbf{W}$ exist such that points $y_i = \mathbf{W} \cdot \mathbf{X}_i$ are clustered in k intervals, each containing vectors from a single class only. A dataset that is k-separability may also be $k + m$ separable, until $k + m = n$ is reached. Although usually the minimal k is of interest sometimes higher k's may be preferred if the margins between projected clusters are larger, or if among k clusters some have very small number of elements. Problems that may be solved by linear projection on no less than k-clusters belonging to alternating classes are called k-separability problems [40]. This concept allows for a better characterization of the space of all non-separable problems. The difficulty of learning grows quickly with the minimum k required to solve a given problem. Linear (2-separable0 problems are quite simple and may be solved with linear SVM or any other variant of LDA model. Kernel transformations may convert some problems into linearly separable in higher dimensional space. Problems requiring $k = 3$, for example the XOR problem, are already slightly more difficult for non-local transformations (for example MLPs), and problems with high k quickly become intractable for general classification algorithms.

Among all Boolean classification problems linear separability is quite rare. For 3 bits there are 8 vertices in the cube and $2^8 = 256$ possible Boolean functions. Two functions are constant (always 0 or 1), 102 are linearly separable, 126 are 3-separable and 26 are 4-separable functions. For more than half of the 3-bit Boolean functions there is no linear projection that will separate the data. Almost half (126) of all the functions give at least 3 alternating clusters. For the 4-bit problem there are 16 hypercube vertices, with Boolean functions corresponding to 16-bit numbers, from 0 to 65535 (64K functions). The number of linearly separable functions is 1880, or less than 3% of all functions, with about 22%, 45% and 29% being 3 to 5-separable. About 188 functions were found that seem to be either 4 or 5-separable, but in fact contain projection of at least two hypercube vertices with different labels on the same point. Although the percentage of linearly separated functions rapidly decreases relatively low k-separability indices resolve most of the Boolean functions.

Changing the goal of learning may thus help to discover much simpler data models than those provided by kernel methods. The final transformation separating the classes on a line with k-intervals has only $k - 1$ parameters. Periodic or quasi-periodic separability in one dimension is worth considering to avoid high number of intervals. Other simplified transformations that handle different types of non-linearities may be defined in two or more dimensions. Mapping on the chessboard targets, or on localized Voronoi

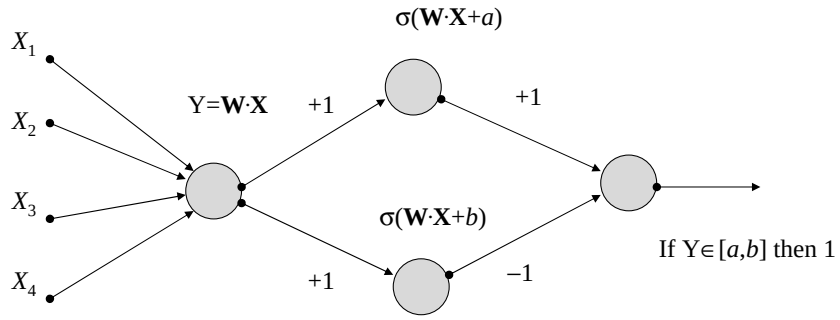

Fig. 3. MLP solution to the 3-separable case.

cells defined by prototypes localized on a regular multidimensional grid, may handle directly quite difficult non-linearities.

New targets require a different approach to learning because the vector labels Y_X do not specify to which interval a given vector $\mathbf{X}$ belongs. Gradient training is still possible if soft windows based on combinations of sigmoidal functions are used. Thus for 3-separable problems the final transformation is from 3 intervals: $[-\infty, a], [a, b], [b, +\infty]$ to $-1, +1, -1$ values. For the middle interval a soft window functions may be set $S(x; a, b) = \tanh(x - a) - \tanh(x - b) - 1 \in [-1, +1]$. Quadratic error function suitable for learning is:

$$E(a, b, \mathbf{W}) = \sum_{\mathbf{X}} (S(\mathbf{W} \cdot \mathbf{X}; a, b) - Y_X)^2 \tag{6}$$

Starting from small random weights the center y_0 of projected data $y = \mathbf{W} \cdot \mathbf{X}$, the range $[y_{\min}, y_{\max}]$ is estimated, and a, b values are set to $y_0 \pm |y_{\max} - y_{\min}|/4$. The weights and the $[a, b]$ interval are trained using gradient method. It is also possible to implement 3-separable backpropagation learning in purely neural architecture based on a single linear neuron or perceptron for projection plus a combination of two neurons creating a "soft trapezoidal window" function $S(x; a, b)$ that passes only the output in the $[a, b]$ interval [47]. The two additional neurons (Fig. 3) have fixed weights ($+1$ and -1) and biases a, b, adding only two adaptive parameters. An additional parameter determining the slope of the window shoulders may be introduced to scale the $\mathbf{W} \cdot \mathbf{X}$ values as the weights grow. The input layer may of course be replaced by hidden layers that implement additional mappings, for example kernel mappings, thus making this at least as powerful as SVM methods.

This network architecture has $n + 2$ parameters and is able to separate a single class bordered by vectors from other classes. For n-dimensional 3-separable problems standard MLP architecture requires at least two hidden neurons connected to an output neuron with $2(n + 1) + 3$ parameters. For k-separability case this architecture will simply add one additional neuron for each new interval, with one bias parameter. n-bit parity problems require only n neurons (one linear perceptron and $n - 1$ neurons

with adaptive biases for intervals), while in the standard approach $O(n^2)$ parameters are needed [92]. Tests of such architectures showed (W. Duch, R. Adamczak, M. Grochowski, in preparation) that indeed one may learn dificult Boolean functions this way. In fact we are training here a single bi-central function with rotations (Eq. 5), creating simplest possible model of the data.

Algorithms of this type, projecting data on many disjoined pure clusters, may have biological justification. Neurons in association cortex form strongly connected microcircuits found in cortical columns, resonating with different frequencies when an incoming signal $X(t)$ appears. This essentially projects the signal into high-dimensional space. A perceptron neuron observing the activity of a column containing many microcircuits learns to react to signals in an interval around particular frequency in a supervised way based on Hebbian principles. It is sufficient to combine outputs from selected microcircuits correlated with the category that is being learned. In case of signals microcircuits may be treated as resonators specializing in discovering interesting signal structures, for example Gabor filters in vision. A parallel array of one-bit threshold quantizers with sums of inputs is a crude approximation to such model. It achieves not only optimal signal detection, but even for suprathreshold input signals it improves its performance when additional noise is added, a phenomenon called "suprathreshold stochastic resonance" [147]. In case of abstract reasoning combination of disjoint projections on the $\mathbf{W} \cdot \mathbf{X}$ line is more useful than simple quantizers.

3.6 Prototype-based rules for data understanding

Most attempts to understand the structure of data in machine learning is focussed on extraction of logical rules [62, 47]. Relations between fuzzy logic systems and basis set networks are fairly obvious and have been described in details [103, 93]. The use of Gaussian functions in the Radial Basis Function (RBF) networks is equivalent to the use of sets of fuzzy rules with Gaussian membership functions. Although it is an interesting fact in itself, it has not lead to any new development, in most applications simple Gaussian classifiers are created. To optimize fuzzy systems neural adaptation techniques may be used, leading to neurofuzzy systems [50, 133, 136].

Fuzzy set $\mathcal{F}$ is defined by the universe $\mathcal{X}$ and the membership functions $\chi_F(X)$, specifying the degree to which elements of this universe belong to the set $\mathcal{F}$. This degree should not be interpreted as probability [109] and in fact at least four major interpretations of the meaning of membership functions may be distinguished [13]. One natural interpretation is based on the degree to which all elements $X \in \mathcal{X}$ are similar to the typical elements (that is those with $\chi_F(X) \approx 1$) of $\mathcal{F}$. From this point of view fuzzy modeling seems to be a special case of similarity modeling, field that have not yet been fully developed. On the other hand fuzzy models are quite successful and may contribute to new similarity measures. Relations between fuzzy rules and similarity to prototypes are worth more detailed exploration.

An analogy with human object recognition is quite fruitful. Perceiving and recognizing an object requires attention to be paid to its most characteristic features. First feature values X_i are measured by our sensory systems with relatively high precision (deriving, for example, physical parameters of sound), and then primary and secondary

sensory cortex transforms these input values using higher-order receptive fields that integrate spatio-temporal patterns facilitating recognition of complex patterns (for example, phonemes). In fuzzy modeling each feature X_i of an object $\mathbf{X}$ is filtered through a large receptive field $\mathcal{F}_{ij}$, defined by a membership function $\mu_{F_j}(X_i)$. Simple MFs, such as triangular, trapezoidal or Gaussian, are used to model the degree to which some value X_i belongs to the receptive field $\mathcal{F}_{ij}$. Comparing to the sophisticated processing of sensory signals by the brain this is a very crude approach in which larger receptive fields are obtained directly from individual features using membership functions, instead of non-linear combinations of several features. Brain-like information processing may of course be more accurately modeled using hierarchical fuzzy systems.

Selection of prototypes and features together with similarity measures offers new, so far unexplored alternative to neurofuzzy methods [49, 178, 15]. Duality between similarity measures and membership functions allows for generation of propositional rules based on individual membership functions, but there are significant differences. Fuzzy rules first apply large receptive fields (membership functions) to these individual features, combining them in conditions of rules later. P-rules in their natural form first create a combination of features (via similarity functions) and then apply various membership functions to this combination. Neurofuzzy systems generate fuzzy rules and optimize membership functions [133, 136] using input vectors defined in fixed-dimensional feature spaces. Similarity may be evaluated between objects with complex structures that are not easy to describe using a common sets of features. In particular the use of probabilistic, data dependent distance functions allows for definition of membership functions for symbolic data (such as the sequential DNA or protein strings) that may be difficult to derive in other way.

Experiments in cognitive psychology show that logical rules are rarely used to define natural categories, human categorization is based on memorization of exemplars and prototypes [146]. Similarity functions may be used to model the importance of different features in evaluating similarity between the new case in relation to stored prototypes. Multiplicative similarity factors may easily be converted to additive distance factors and vice versa. Rule-based classifiers are useful only if the rules they use are reliable, accurate, stable and sufficiently simple to be understood [47]. Prototype-based rules are useful addition to the traditional ways of data exploration based on crisp or fuzzy logical rules. They may be helpful in cases when logical rules are too complex or difficult to obtain. A small number of prototype-based rules with specific similarity functions associated with each prototype may provide complex decision borders that are hard to approximate using logical systems, but are still sufficiently simple to understand them. For example, such simple rules have been generated for some medical datasets using heterogeneous decision tree [78]. A single P-rule for breast cancer data classifying as malignant cancer all cases that are closer to prototype case (taken as one of the training cases) than a certain threshold achieves 97.4% accuracy (sensitivity 98.8% and specificity 96.8%). The accuracy in this case is at least as good as that of any alternative system tried on this data.

Combining various feature selection and prototype selection methods with similarity functions leads to many interesting algorithms. An interesting possibility is to use the prototype-based rules to describe exceptions in the crisp or fuzzy logic systems.

Systematic investigation of various membership functions, T-norms and co-norms, and their relation to distance functions is certainly worth pursuing. The algorithms for generation of P-rules should be competitive to the existing neurofuzzy algorithms and will become an important addition to the methods of computational intelligence. Although similarity measures provide great flexibility in creating various decision borders this may turn to be a disadvantage if the primary goal is to understand the data rather than make most accurate predictions (neurofuzzy approaches have of course the same problem). Optimized similarity measures may not agree with human intuition and in some cases larger number of prototypes with simpler similarity measures may offer more acceptable solution.

3.7 Multiple models for meta-learning

Multi-objective optimization problems do not have a single best solution. Usually data mining systems return just a single best model but finding a set of Pareto optimal models if several criteria are optimized is much more ambitious goal. For example, accuracy should be maximized, but variance should be minimized, or sensitivity should be maximized why the false alarm rate should be minimal. The search process for optimal models in meta-learning should explore many different models. Models that are close to the Pareto front [130] should be retained and evaluated by domain experts.

A forest of decision trees [77] and heterogeneous trees [78] is an example of a simple meta-search in a model space restricted to decision trees. Heterogeneous trees use different types of rule premises, splitting the branches not only using individual features, but also using tests based on distances from the training data vectors. These trees work in fact in a kernel space, but the optimization process is quite different than in the SVM case. In case when linear discrimination works well standard decision trees may give poor results, but adding distance-based conditions with optimal support vectors far from decision borders provides flat spherical borders that work as well as hyperplanes. The beam search maintains at each stage k decision trees (search states), ordering them by their accuracy estimated using cross-validation on the training data [78]. This algorithm has found some of the simplest and most accurate decision rules that gave different tradeoffs between sensitivity and specificity.

The metalearning search procedure creates many individual models and it would be wasteful not to use them, unless only models of specific types are of interest (for example, models that are easily comprehensible). Metalearning usually leads to several interesting models, as different types of optimization channels are enabled by the search procedure. If a committee of models is desired diversification of individual models that should perform well in different regions of input space may be necessary, especially for learning of difficult tasks. The mixture of models allows to approximate complicated probability distributions quite accurately improving stability of individual models. Individual models are frequently unstable [17], i.e. quite different models are created as a result of repeated training (if learning algorithms contains stochastic elements) or if the training set is slightly perturbed [6].

Although brains are massively parallel computing devices attention mechanisms are used to inhibit parts of the neocortex that are not competent in analysis of a given type of signal. All sensory inputs (except olfactory) travel through the thalamus where their

importance and rough category is estimated. Thalamic nuclei activate only those brain areas that may contribute useful information to the analysis of a given type of signals [166]. This observation may serve as an inspiration for construction of better algorithms for data analysis. In the metasearch process all models that handle sufficiently many cases mistreated by other models should be maintained.

A committee based on competent models, with various factors determining regions of competence (or incompetence) may be used to integrate decisions of individual models [54, 55]. The competence factor should reach $F(\mathbf{X}; M_l) \approx 1$ in all areas where the model M_l has worked well and $F(\mathbf{X}; M_l) \approx 0$ near the training vectors where errors were made. A number of functions may be used for that purpose: a Gaussian function $F(||\mathbf{X}-\mathbf{R}_i||; M_l) = 1 - G(||\mathbf{X}-\mathbf{R}_i||^a; \sigma_i)$, where $a \geq 1$ coefficient is used to flatten the function, a simpler $1/\left(1 + ||\mathbf{X} - \mathbf{R}_i||^{-a}\right)$ inverse function, or a logistic function $1 - \sigma(a(||\mathbf{X} - \mathbf{R}_i|| - b))$, where a defines its steepness and b the radius where the value drops to 1/2. Because many factors are multiplied in the incompetence function of the model each factor should quickly reach 1 outside of the incompetence area. This is achieved by using steep functions or defining a threshold values above which exactly 1 is taken.

Results of $l = 1 \ldots m$ models providing estimation of probabilities $\mathcal{P}(C_i|\mathbf{X}; M_l)$ for $i = 1 \ldots K$ classes may be combined in many different ways [110]: using majority voting, averaging results of all models, selecting a single model that shows highest confidence (i.e. gives the largest probability), selecting a subset of models with confidence above some threshold, or using simple linear combination. For class C_i coefficients of linear combination are determined from the least-mean square solution of:

$$\mathcal{P}(C_i|\mathbf{X}; M) = \sum_{l=1}^{m} \sum_{m} W_{i,l} F(\mathbf{X}; M_l) \mathcal{P}(C_i|\mathbf{X}; M_l) \tag{7}$$

The incompetence factors simply modify probabilities $F(\mathbf{X}; M_l)\mathcal{P}(C_i|\mathbf{X}; M_l)$ that are used to set linear equations for all training vectors $\mathbf{X}$, therefore the solution is done in the same way as before. After renormalization $\mathcal{P}(C_i|\mathbf{X}; M) / \sum_j \mathcal{P}(C_j|\mathbf{X}; M)$ give final probability of classification. In contrast to AdaBoost and similar procedures [10] explicit information about competence, or quality of classifier performance in different feature space areas, is used here. Many variants of committee or boosting algorithms with competence are possible [110], focusing on generation of diversified models, Bayesian framework for dynamic selection of most competent classifier [69], regional boosting [122], confidence-rated boosting predictions [154], task clustering and gating approach [9], or stacked generalization [183].

A committee may be build as a network of networks, or a network where each element has been replaced by a very complex processing element made from individual network. This idea fits well to the transformation-based learning. Incompetence factors may be used to create virtual subnetworks, with different effective path of information flow. Modulation of the activity of modules is effective only if the information about the current state is distributed to all modules simultaneously. In the brain this role may be played by the working memory. Here it can be replaced by a networks of modules adjusting their internal states (local knowledge that each module has learned) and

their interactions (modulations of weights) to the requirements of the information flow through this system.

4 Beyond pattern recognition

The transformation-based approach described here is quite general and may be used for all kinds of pattern recognition problems, classification, approximation, pattern completion, association and unsupervised learning, extending what has already been done in the similarity-based approach [35, 43]. Computational intelligence should go beyond that, using partial observations (perceptions) to reason and learn from them.

Biological systems may be viewed as associative machines, but associative memories do not capture essential features of this process. Real brains constantly learn to pay attention to relevant features and use correlations between selected feature values and correct decisions. People may learn to act appropriately without explicitly realizing the rules behind their actions, showing that this process may proceed intuitively, without conscious control. Associative machines should learn from observations correlation of subsets of features and apply many such correlations in decision making process. Problems solved by symbolic artificial intelligence are of this sort. Bits and pieces of knowledge should be combined in a search for a final solution, and this leads in all interesting cases to a combinatorial explosion.

Not much progress has been made along this line of research in the last decades, although already in the PDP Books [151] several articles addressed combinatorial problems that are beyond pattern recognition. Boltzmann machines and harmony theory have been used to answer questions about complex systems from partial observations [151], but they proved to be very inefficient because the stochastic training algorithm needs time that grows exponentially with the size of the problem. Helmholtz machines [28], and recently introduced multi-layer restricted Boltzmann machines and deep belief networks [87] have been used only for pattern recognition problems so far. These models are based on stochastic algorithms and binary representations and thus are rather restricted.

Inferences about complex behavior from partial observations require systematic search. Suppose that a number of relations between small subsets of all variables characterizing complex system or situation are known *a priori* or from observations. For example, representing discrete changes of 3 variables ($\Delta A = +$ for increase, $\Delta A = -$ for decrease and $\Delta A = 0$ for no change) $3^3 = 27$ possibilities are distinguished, from all three variables decreasing, $(\Delta A, \Delta B, \Delta C) = (-,-,-)$ to all three increasing $(\Delta A, \Delta B, \Delta C) = (+,+,+)$. If these variables are constrained by additive $A = B+C$, multiplicative $A = B \cdot C$ or inverse additive $A^{-1} = B^{-1}+C^{-1}$ relations $A = f(B,C)$ then for all these relations 14 of the 27 possibilities cannot occur, for example $\Delta A = 0$ is impossible if both $\Delta B = \Delta C = -$ or both are $+$). If many such relations are applicable for N variables out of 3^N possible solutions only a few will be in agreement with all constraints. Assuming specific relations: $f(A_1, A_2) = A_3; f(A_2, A_3) = A_4; ...f(A_{N-2}, A_{N-1}) = A_N$ leaves only $4N + 1$ possible solutions. For a large N is is a negligible fraction of all 3^N possibilities. Relations among 3 variables – representing partial observations – are stored in "knowledge atoms", or nodes arranged

in one-dimensional array, connected to relevant input features. If the values of any two variables A_i, A_{i+1} or A_i, A_{i+2} are known then one of such nodes will provide the value of the third variable. In at most $N-2$ steps, in each step selecting nodes that have only one unknown input, all values of variables are determined. Suppose now that only a single variable in two nodes has specific value, for example A_1 and A_4. An assumption about the value of A_2 should be made, starting 3 branches of a search tree with $A_2 = -, 0, +$. In the first step this leads to an inference of A_3 value, and in the second step $f(A_2, A_3) = A_4$ is checked, leaving only those branches for which both relations are true.

This is obviously a very fortuitous set of relations, but in most real situations a very small search trees, sometimes reduced to a single branch, are still sufficient. An application to the analysis of a simple electric circuit (7 variables, currents I, voltages V and resistances R) [151] using network that keeps in its nodes relations between I, V, R (Ohm's law) and Kirchoff laws ($V = V_1 + V_2$ and $R = R_1 + R_2$) has been reported [50], showing how *a priori* knowledge enables solutions to problems involving qualitative changes of currents, voltages and resistances. The feature space representation works as a powerful heuristics in the reasoning process. In the seven variable problem considered here there are $3^7 = 2187$ different strings of 7 variables, but only 111 may actually be realized. 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 situations that fulfills all the constraints is vanishingly small. Network nodes may also implement functions that determine numerical values of unknown variables (such as $V = I \cdot R$).

Such network of knowledge atoms may solve mathematical problems without explicit transformation of equations, and when qualitative reasoning is sufficient it serves as a model of intuitive computing. For more complex situations hierarchical decomposition of the problem is necessary, depending on the questions asked. For example, changes of parameters of a single or a few elements of a complex electrical circuit may be decomposed into blocks, and there is no need to assign values to all variables. People in such cases analyze graphical structure of connections and nodes representing the problem, starting from elements mentioned in the problem statement.

Network nodes may also estimate probabilities of different observations, and then the algorithm may use them in sequential Markov-chain reasoning or in other variants of such algorithms approximating joint distribution of variables in various ways. The confabulation architecture [86] is an example of such algorithm that uses products of a few (usually 4) conditional probabilities in a specific way
$\arg\max_j(\min[p(i_1|j)p(i_2|j)p(i_3|j)p(i_4|j)]$,
where $p(i_1|j)$ is the probability that word i_1 precedes word j in a sequence. This algorithm trained on a large corpus of English stories produces plausible words j, although it cannot capture the meaning of the story or learn a strategy of games played, due to the lack of long-term dependencies and structural representation of the concepts.

Similar approaches may be useful in many other fields. In intelligent control random actions may be correlated with probability distributions of different results, creating several scenarios [167]. Problems of this type are somewhere between pattern recognition and typical artificial intelligence problems. Neural networks (a core CI technology)

may be used as heuristics to constrain search (a core AI technology) in problem solving. Robots, including autonomous vehicles, need to combine reasoning with pattern recogntion in a real time. It would be very worthwhile to collect data for real problems of this kind, encouraging the development of algorithms for their solutions.

The inference process in this approach resembles human reasoning that either proceeds sequentially, or temporarily assumes one of the possibilities, checking if it is consistent with all knowledge assumed to be true at a given stage. The interplay between left and right hemisphere representations leads to generalization of constraints that help to reason at the meta-level [38]. These ideas may form a basis for an associative machine that could reason using both perceptions (observations) and *a priori* knowledge. In this way pattern recognition (lower level cognitive functions) may be combined in a natural way with reasoning (higher level cognitive functions).

A very interesting approach to representation of objects as evolving structural entities/processes has been developed by Goldfarb and his collaborators [74, 72, 73]. Structure of objects is a result of temporal evolution, a series of transformations describing the formative history of these objects. This is more ambitious than the syntactic approach in pattern recognition [67], where objects are composed of atoms, or basic structures, using specific rules that belong to some grammar. In the evolving transformation system (ETS) object structure is a temporal recording of structured events, making syntax and semantics inseparable. ETS formalism leads to a new concept of class which is represented by similar structural processes. Classes defined by decision borders do not capture the meaning of objects. Inductive learning process should discover class structures as a series of transformations that change the primitive elements to their observed structure. In this way a generative model is produced that may generate an infinite set of examples of objects from a given class.

This line of thinking is in agreement with the modified goal of learning presented above. Various initial transformations should discover interesting representations of different aspects of objects, learn how to measure their (dis)similarities introducing kernel functions. These transformations may be viewed as receptive fields of sensors observing the data, or as selection of operations that compare objects in some specific way. For comparison of strings, to take the simplest example, various substring operations may be used. ETS is also in agreement with the idea of computing and learning as compressions, as evolving transformations compress the information. Application of ETS to structural representation of molecules has been presented [73], and structural representation of spoken language has been analyzed from this point of view [81].

5 Neurocognitive inspirations, computing, cognition and compression

How to scale up our present models to perform more interesting cognitive functions? Neurocognitive inspirations lead to modular networks that should process information in a hierarchical way that roughly should correspond to functions of various brain areas, and these networks become modules that are used to build next-level supernetworks, functional equivalents of larger brain areas. The principles on which models should be based at each level are similar [61]: networks of interacting modules should adjust to

the flow of information (learn) changing their internal knowledge and their interactions with other modules. Efficient algorithms for learning are known only at the lowest level, when very simple interactions and local knowledge of processing elements are assumed. The process of learning leads to emergence of novel, complex behaviors and competencies. Maximization of system information processing capacity may be one guiding principle in building such systems: if the supernetwork is not able to model all relations in the environment then it should recruit additional members that will specialize in learning facts, relations or behaviors that have been missing.

Very complex supernetworks, such as the individual brains, may be further treated as units that cooperate to create higher-level structures, such as groups of experts, institutions, think-tanks or universities, commanding huge amounts of knowledge that is required to solve the problems facing the whole society. Brain-storming is an example of interaction that may bring ideas up that are further evaluated and analyzed in a logical way by groups of experts. The difficult part is to create ideas. Creativity requires novel combination, generalization of knowledge that each unit has, applying it in novel ways. This process may not fundamentally differ from generalization in neural networks, although it takes place at much higher level of complexity. The difficult part is to create a system that has sufficiently rich, dense representation of useful knowledge to be able to solve the problem by combining or adding new concepts/elements [38].

The brain has much more computing power than our current machinery and thus may solve problems in a different way. Nevertheless brain resources are limited and the mechanism of encoding the new information using old memory structures is quite natural, leading to a great conservation of resources and enabling associative recall. This is also a source of serious difficulty in defining the meaning of symbols, encoded by activity of neural microcircuits that is constantly changing, spreading activation to other concepts. As a result relations between concepts change depending on the context, making the invariant meaning of concepts only a rough approximation. In experimental psychology this process, known as semantic priming, is one of the most popular subjects of investigation [126].

How can this priming process be approximated? An attractor network model has been created to explain results of psychological experiments [25]. However, such dynamical models are rather complex and do not scale well with the size of the network. Processing sequential information by simpler mechanisms, such as spreading activation in appropriately structured networks, is more suitable for information retrieval [26]. The challenge here is to create large semantic networks with overall structure and weighted links that facilitate associations and reproduce priming effects. Wordnet (http://wordnet.princeton.edu) and many other dictionaries provide some useful relations that may be used in network construction, although these relations capture a small fraction of knowledge and associations that humans have about each concept. Medical applications of spreading activation networks are easier to create because huge lexical resources are available [124].

Activation of semantic networks may be seen as dynamical construction of a relevant feature space, or non-zero subspace for vector models in information retrieval. New information is projected into this space, expanding it by adding new dimensions with non-zero components. Although the use of semantic networks is quite natural geometri-

cal description of this process is still interesting. In geometrical model activation of the network concept node corresponds to a change of a metric around this concept (active concept associates with other active concepts), and this changes similarity relations at a given moment. Concepts in some specific meanings attract other concepts that become closer, while other concepts increase their distance, facilitating disambiguation. Static vector space models do not take that into account such dynamical changes, therefore spreading activation models should have an important advantages here.

Projecting new information on the semantic network creates strong associations with existing network nodes, encoding partially new knowledge in terms of the old one. Basic perceptual and conceptual objects of mind are created early in the developmental process, therefore perceptual information will be predominantly encoded in terms of the old knowledge. Seeing or hearing new objects will be remembered by adding new nodes that bind together (symbolize) specific configuration of node activations, responsible for recognition of natural categories. These nodes in turn are linked to new nodes coding abstract concepts that do not activate directly any perceptual nodes, allowing for higher and higher levels of abstraction.

Computing and cognition may be seen as information compression, although there are clearly exceptions. In the brain signals may be projected into higher-dimensional space, for example information from retina is projected on visual cortex with order of magnitude expansion in the number of cells involved. In computing we have seen that kernel methods make implicit projection into a highly-dimensional space to achieve separability before final reduction of information is made. Although the idea of cognition as compression is worth exploring it has been so far developed only for sequential, one-dimensional systems [182], and to some extent in our approach to concept disambiguation in medical domain[124]. Multiple alignment may be applied to sequential data, while computational intelligence methods work with many other types of data, including signals, vectors, spatial structures (like chemical molecules), or multimedia data. To account for preliminary processing of sensory and motor signals a more general approach to data transformation is needed. Instead of a sequence alignment calculus based on "small world" active subnetworks in the huge semantic network encoding relations between mind objects is required. New knowledge activates old similar structures, extends them in new directions, and thus is encoded in a compressed way. This process may be approximated using graphs representing active network nodes that represent current mind state. Associations between different network configurations are determined by transition probabilities between network states.

Several interesting developments emerged from neurocognitive inspirations, the two most important theories being the liquid computing [121] and laminar computing [80, 145]. Liquid computing concentrates on microcircuits and columnar architecture, and laminar computing on the layered architecture of neocortex. The neocortex microcircuits composed of densely connected neurons within a diameter of 500μm are heterogeneous and differ across brain regions. Many properties of these microcircuits are stereotypical, therefore a concept of a generic microcircuit template may be a useful abstraction allowing for understanding of dynamical properties of the cortex. Maass and Markram [120] argue that online computing is indistinguishable from learning, because temporal integration of information in a stream of data constantly changes the

system. Learning is thus an integral part of microcircuits dynamics. Boolean functions and approximations to Boolean functions may be computed by feedforward networks with stereotypical basis functions, including sigmoidal or any other nonlinear functions. Computing such functions is thus relatively easy, there are many biophysical mechanisms that can influence neuronal states and thus can be interpreted as performing computations and learning. A useful approximation to microcircuit dynamics may be provided by finite state automata [18, 29, 34]. The Liquid State Machine (LSM) model aims at better approximation at the microscopic level, based on "liquid" high dimensional states of neural microcircuits that change in real time. In [34] another approximation relating neurodynamical states to psychological space and mental events has been advocated to bridge neuroscience and psychology.

LSM treats complex dynamical systems as non-linear filters transforming input signals $x(t)$ into activations of microcircuit (hidden) neurons $h(t) = L(x(t))$. The output stream $y(t) = M(h(t))$ is then provided using "readout neurons" that learn to select and recombine activations of the hidden layer. In fact this is a simple transformation model that projects signals into high-dimensional spaces $h(t)$ created by microcircuits in form of temporal filters or basis functions, where separation is relatively straightforward. Oscillators based on neurons with diverse time constants are needed to create good projections. Linear or simple perceptron readout neurons are sufficient to approximate any time-varying signal. A single LSM may be used for various tasks, depending on the training of these output neurons. It has a fading memory for signals, depending on the time constants of the neurons. In fact any larger reservoir of neurons with some memory will be sufficient to create projections of signals to high-dimensional space. The Echo State Networks, or more general Reservoir Computing, use untrained recurrent neural networks as "reservoirs of activity" to implement this projection. These approaches are equivalent to the Liquid State Machines [84] and are being applied to time series prediction, dynamical system identification and speech recognition with great success [84].

The question how to use neural elements to compute Boolean functions has many solutions [120]. A complementary question that is also worth asking is: how can neural circuits discover approximate but rather complicated logic (Boolean function) in data, going beyond trivial associations? How are different activations of the hidden layer recombined to enable this? This is what humans and animals evidently do all the time. LSM has so far been used on in problems where simple associations are sufficient. k-separability and other more general goals of learning may be responsible for the ability of cortex to learn complex approximate logic using liquid states. One good algorithm for k-separability combines projections with discovery of local clusters. This is essentially what is needed for object-centered representations in vision [31] and has been used to model the outputs of parietal cortex neurons [141, 142]. Any continuous sensory-motor transformation may be approximated in this way [153]. Although precise neural implementation of such basis functions is not clear they may result from the activity of microcircuits. Generalization of k-separability to time-dependent transformations done by liquid state machines is rather straightforward. Therefore it is quite likely that k-separability is an interesting abstraction of a basic biological mechanism.

Laminar computing is based on inspirations derived from organization of neocortex into six layers with specific architecture, and information flow in the brain on macroscopic scales. The bottom-up interactions provide signals from the senses, while the top-down interactions provide expectations or prior knowledge, that helps to solve ill-defined problems. Finally the horizontal interactions enable competition and specialization. Laminar computing has been studied initially in the visual cortex [80] but seems to have captured more general principles of cortical computations [145]. It explains how distributed data is grouped into coherent object representations, how attention selects important events and how cortex develops and learns to express environmental constraints. The laminar computing models have been used to explain many neurophysiological and psychophysical results about visual system, but it has also been used to develop new algorithms for image processing. In practical applications a version of Adaptive Resonant Theory [19] called LAMINART [145], has been used. So far this is the most advanced approach to perception that will certainly play a very important role in the growing field of autonomous mental development and cognitive robotics.

6 Summary of open problems

Many open problems have been identified in the preceding chapters. Below is a summary of some of them:

– Solid foundations of computational intelligence that go beyond probabilistic pattern recognition approaches are needed.

The current state is clearly not satisfactory. Bayesian probabilistic framework forms the foundation of many pattern recognition methods [63, 174, 164] and graphical networks [99], but it does not cover most branches of computational intelligence. Good foundations should help to create methods that adjust themselves to the data, finding the simplest possible model that accurately describes the data. The no-free-lunch theorem shows that a single approach is not sufficient [63] and automatic selection of methods included in larger data mining packages is still too difficult as these methods are not presented from the point of view of common foundations.

Earlier suggestions to base CI foundations on similarity concept [35, 43, 139] led to a framework in which meta-learning could be implemented [53] using search techniques in the model space. It has been found that similarity-based rule systems are equivalent to fuzzy systems [49], providing an alternative way to understand data, and that neural algorithms can also be presented in this framework. Heterogeneous constructive systems of this type are especially useful and have already discovered some of the simplest descriptions of data [78, 51]. A framework based on transformations, presented in this paper for the first time, is even more general, as it includes all kinds of pre-processing and unsupervised methods for initial data transformations, and looks at learning as a series of data transformations, defining new goals of learning. The work on hyperkernels also goes in similar direction [135].

Although this framework is still in the initial stage of its development it has a chance to provide foundations for computational intelligence models. Adaptation of

these models requires some optimization techniques, processes that operate on admissible transformations. A new generation of data mining software, capable of implementing arbitrary sequences of transformations for meta-learning, is in development (K. Grąbczewski and N. Jankowski, in preparation).

– Existing methods cannot learn difficult problems.

CI algorithms are rarely addressing real, difficult problems, and many researchers are convinced that universal approximators, such as neural networks, are good tools to learn the structure of any data. In fact off-the-shelf systems work well only for problems that are "not-too-far" from the linearly separable problems. Fortunately many interesting datasets in machine learning domain are of this kind. Problems with inherent approximate Boolean logic become quickly intractable with growing complexity of the logical functions. Growing complexity of such problems may be characterized using the notion of k-separability [40], or even more general goals for learning. We are probably not aware how many problems in bioinformatics or text processing are intractable and therefore are ignoring them. Datasets for such difficult problems are needed to increase awareness of the need for such methods, but first we should be able to solve at least some of these problems.

– Problems requiring reasoning based on perceptions should be explored.

The basic function of simple brains is to link perception with action. Agents with more complex brains have internal states and goals, and need to perceive and reason before making actions. Agent-based approach in artificial intelligence [152] is usually based on symbolic reasoning and logical rules. There has been some work on hybrid neural-symbolic architectures [176], but not much effort devoted to neural architectures capable of representations of predicate relations. The connectionist model Shruti [158, 175] seems to be an exception, although it has not gained wider popularity. Although spreading activation networks allow for some reasoning (mostly disambiguation and coherent interpretation of concepts [124]) this is not sufficient to solve problems requiring combinatorial search, compositionality, problems arising in sentence parsing or sequential reasoning. How to control spreading activation to account for systematic thinking process? New mathematical techniques for representation of objects and relations by active subgraphs in large semantic networks seem to be required. Search processes may be constrained by "intuitive computing" using neural heuristics [34]. Reinforcement learning [163], reservoir learning [84], laminar computing [80, 145] and chunking [134] should also be used as general mechanisms for sequence learning and divide-and-conquer sub-problem parsing. No cognitive architectures have captured so far all these processes, and combining them together is an interesting challenge.

Neurocognitive inspirations for understanding language and general cognitive processes lead to distributed connectionist systems that can be approximated by networks with local activations, and that in turn may be partially understood in terms of symbolic processes and probabilistic or finite automata [176]. Investigation of relations between approximations of brain functions at different level of complexity is quite fruitful, leading to new models of mental processes based on psychological spaces [34, 61]. At the

simplest level simple perceptrons are found, with a single internal parameter and synaptic interactions based on fixed weight connections. Enhanced perceptrons sensitive to phase synchronization [111] are able to solve the famous connectedness and other problems posed by Minsky and Papert [131]. Networks with spiking neurons also have this capability [173], but it is not clear if they have additional powers – characterization of different complexity classes seems to be incomplete here. At a higher level complex processing elements modeling whole microcircuits are found, and even higher networks of networks and societies of minds. Although new biological mechanisms at the level of synaptic learning are certainly very important, simple abstractions of complex functions such as self-organization proved to be quite interesting. At the level of approximating microcuircuit and minicolumn functions simpler mechanisms, implementing for example some form of k-separability learning, may also be useful. Perhaps correlation-based learning, as in the Alopex algorithm [170], would be sufficient for biological implementation? Such approximations may also be viewed as information compression [182].

– Methodology of evaluation and development of CI methods is urgently needed.

Every year hundreds of methods are introduced, some of them rediscovered many times, others being minor variations on the known themes. It is well known that a data on which a given method works well may always be found [63]. There is no simple way to evaluate *a priori* how a new method will perform on a real data, but at least it is possible to understand what type of decision borders it is using and thus what type of problems it may solve. Efforts to provide standard benchmarks have largely failed. For example, the Data for Evaluating Learning in Valid Experiments (Delve) project [2] has been quite promising, presenting problems of growing complexity, but the project has unfortunately been abandoned.

Without standard methodology of evaluating new approaches no progress will be possible. There is a tendency to create algorithms and test them on new datasets, ignoring good reference results that are hard to compete with. Even though quite interesting datasets are available from some competitions many papers end with one or two trivial examples. Benchmarks that go beyond pattern recognition are particularly hard to find. It is up to the editors and reviewers of journals to enforce comparison with the simplest methods that may solve similar problems, and require solutions to problems of increasing complexity. For example, in extraction of logical rules comparison with rules extracted by popular decision trees should be required [62], or in classification problems comparison with linear discrimination and the nearest neighbor model. Collecting raw data, results of data analysis and software implementing different methods should be encouraged by data repositories and large conferences.

Comparison of accuracy does not address problems in real applications. There may be many performance measures, different costs involved, tradeoffs between model simplicity and accuracy, rejection rate and confidence, or costs of obtaining different features and making different types of errors. Verification and validation of CI models is of great importance in industrial applications where software should always perform up to the intended specifications. Testing the system on data that is similar to the training data

[2] http://www.cs.utoronto.ca/ delve/

may not be sufficient. The standard neural network testing is not able to validate applications used to assess safety of nuclear reactors, chemical plants, military equipment or medical life support systems. These issues are largely ignored by the CI academic community and mostly explored by researchers working in NASA, IBM and other research oriented companies. Visualization of data transformations performed by CI systems, including analysis of perturbation data, are very useful tools [36], although still rarely used.

Cognitive robotics may be an ultimate challenge for computational intelligence. Various robotic platforms that could be used for testing new ideas in semi-realistic situations would be very useful. They have to combine perception, object recognition, reasoning, planning and control in real-time environment. However, computational intelligence has even more general ambitions, looking for solutions to all hard problems for which effective algorithms are not known.

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

References

1. F. Corbacho A. Sierra, J.A. Macias. Evolution of functional link networks. *IEEE Transactions on Evolutionary Computation*, 5:54–65, 2001.
2. N.I. Achieser. *Theory of Approximation*. Frederick Ungar, New York, 1956. Reprinted: Dover Publications, New York 1992.
3. R. Adamczak, W. Duch, and N. Jankowski. New developments in the feature space mapping model. In *Third Conference on Neural Networks and Their Applications*, pages 65–70, Kule, Poland, Oct 1997.
4. J. A. Anderson, A. Pellionisz, and E. Rosenfeld. *Neurocomputing 2*. MIT Press, Cambridge, MA, 1990.
5. J. A. Anderson and E. Rosenfeld. *Neurocomputing - foundations of research*. MIT Press, Cambridge, MA, 1988.
6. R. Avnimelech and N. Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11:483–497, 1999.
7. F.R. Bach and M.I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.
8. P.M. Baggenstoss. The pdf projection theorem and the class-specific method. *IEEE Transactions on Signal Processing*, 51:672–668, 2003.
9. B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.
10. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine learning*, 36:105–142, 1999.
11. Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. *Advances in Neural Information Processing Systems*, 18:107–114, 2006.
12. Y. Bengio, M. Monperrus, and H. Larochelle. Non-local estimation of manifold structure. *Neural Computation*, 18:2509–2528, 2006.
13. T. Bilgiç and I.B. Türkşen. Measurements of membership functions: Theoretical and empirical work. In D. Dubois and H. Prade, editors, *Fundamentals of Fuzzy Sets, Vol. 1*, pages 195–232. Kluver, Boston, 2000.
14. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

15. M. Blachnik, W. Duch, and T. Wieczorek. Selection of prototypes rules context searching via clustering. *Lecture Notes in Artificial Intelligence*, 4029:573–582, 2006.
16. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.
17. L. Breiman. Bias-variance, regularization, instability and stabilization. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 27–56. Springer-Verlag, 1998.
18. Y. Burnod. *An Adaptive Neural Network. The Cerebral Cortex*. Prentice-Hall, London, 1990.
19. G.A. Carpenter and S. Grossberg. Adaptive resonance theory. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks, 2nd ed*, pages 87–90. MIT Press, Cambridge, MA, 2003.
20. G. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, 1987.
21. O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, in print, 2006.
22. N. Chater. The search for simplicity: A fundamental cognitive principle? *Quarterly Journal of Experimental Psychology*, 52A:273–302, 1999.
23. A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing. Learning Algorithms and Applications*. J. Wiley & Sons, New York, 2002.
24. T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.
25. G.S. Cree, K. McRae, and C. McNorgan. An attractor model of lexical conceptual processing: Simulating semantic priming. *Cognitive Science*, 23(3):371–414, 1999.
26. F. Crestani. Application of spreading activation techniques in information retrieval. *Artifical Intelligence Review*, 11(6):453–482, 1997.
27. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
28. P. Dayan and G.E. Hinton. Varieties of helmholtz machines. *Neural Networks*, 9:1385–1403, 1996.
29. A.M. de Callataÿ. *Natural and Artificial Intelligence. Misconceptions about Brains and Neural Networks*. Elsevier, Amsterdam, 1992.
30. L.N. de Castro and J.I. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.
31. S. Deneve and A. Pouget. Basis functions for object-centered representations. *Neuron*, 37:347–359, 2003.
32. T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal Of Artificial Intelligence Research*, 2:263–286, 1995.
33. W. Duch. Neural minimal distance methods. In *Proceedings 3-rd Conference on Neural Networks and Their Applications*, pages 183–188, Kule, Poland, Oct 1997.
34. W. Duch. Platonic model of mind as an approximation to neurodynamics. In S i. Amari and N. Kasabov, editors, *Brain-like computing and intelligent information systems*, pages 491–512. Springer, 1997.
35. W. Duch. Similarity based methods: a general framework for classification, approximation and association. *Control and Cybernetics*, 29:937–968, 2000.
36. W. Duch. Coloring black boxes: visualization of neural network decisions. In *Int. Joint Conf. on Neural Networks, Portland, Oregon*, volume I, pages 1735–1740. IEEE Press, 2003.
37. W. Duch. Uncertainty of data, fuzzy membership functions, and multi-layer perceptrons. *IEEE Transactions on Neural Networks*, 16:10–23, 2005.
38. W. Duch. Computational creativity. In *World Congres on Computational Intelligence, Vancouver, Canada*, pages 1162–1169. IEEE Press, 2006.

39. W. Duch. Filter methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, foundations and applications*, pages 89–118. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2006.
40. W. Duch. k-separability. *Lecture Notes in Computer Science*, 4131:188–197, 2006.
41. W. Duch, R. Adamczak, and G. H. F. Diercksen. Distance-based multilayer perceptrons. In M. Mohammadian, editor, *International Conference on Computational Intelligence for Modelling Control and Automation*, pages 75–80, Amsterdam, The Netherlands, 1999. IOS Press.
42. W. Duch, R. Adamczak, and G. H. F. Diercksen. Neural networks in non-euclidean spaces. *Neural Processing Letters*, 10:201–210, 1999.
43. W. Duch, R. Adamczak, and G.H.F. Diercksen. Classification, association and pattern completion using neural similarity based methods. *Applied Mathemathics and Computer Science*, 10:101–120, 2000.
44. W. Duch, R. Adamczak, and G.H.F. Diercksen. Feature space mapping neural network applied to structure-activity relationship problems. In Soo-Young Lee, editor, *7th International Conference on Neural Information Processing (ICONIP'2000)*, pages 270–274, Dae-jong, Korea, 2000.
45. W. Duch, R. Adamczak, and G.H.F. Diercksen. Constructive density estimation network based on several different separable transfer functions. In *9th European Symposium on Artificial Neural Networks*, Bruges, Belgium, Apr 2001.
46. W. Duch, R. Adamczak, and K. Grąbczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:1–9, 1998.
47. W. Duch, R. Adamczak, and K. Grąbczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.
48. W. Duch and M. Blachnik. Fuzzy rule-based systems derived from similarity to prototypes. *Lecture Notes in Computer Science*, 3316:912–917, 2004.
49. W. Duch and M. Blachnik. Fuzzy rule-based systems derived from similarity to prototypes. In N.R. Pal, N. Kasabov, R.K. Mudi, S. Pal, and S.K. Parui, editors, *Lecture Notes in Computer Science*, volume 3316, pages 912–917. Physica Verlag, Springer, New York, 2004.
50. W. Duch and G. H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.
51. W. Duch and K. Grąbczewski. Heterogeneous adaptive systems. In *IEEE World Congress on Computational Intelligence*, pages 524–529. IEEE Press, Honolulu, May 2002.
52. W. Duch and K. Grudziński. Search and global minimization in similarity-based methods. In *International Joint Conference on Neural Networks*, page Paper 742, Washington D.C., 1999. IEEE Press.
53. W. Duch and K. Grudziński. Meta-learning via search combined with parameter optimization. In L. Rutkowski and J. Kacprzyk, editors, *Advances in Soft Computing*, pages 13–22. Physica Verlag, Springer, New York, 2002.
54. W. Duch and L. Itert. Competent undemocratic committees. In L. Rutkowski and J. Kacprzyk, editors, *Neural Networks and Soft Computing*, pages 412–417. Physica Verlag, Springer, 2002.
55. W. Duch and L. Itert. Committees of undemocratic competent models. In L. Rutkowski and J. Kacprzyk, editors, *Proc. of Int. Conf. on Artificial Neural Networks (ICANN), Istanbul*, pages 33–36, 2003.
56. W. Duch and N. Jankowski. Complex systems, information theory and neural networks. In *First Polish Conference on Neural Networks and Their Applications*, pages 224–230, Kule, Poland, Apr 1994.

57. W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 1999.
58. W. Duch and N. Jankowski. Taxonomy of neural transfer functions. In *International Joint Conference on Neural Networks*, volume III, pages 477–484, Como, Italy, 2000. IEEE Press.
59. W. Duch and N. Jankowski. Transfer functions: hidden possibilities for better neural networks. In *9th European Symposium on Artificial Neural Networks*, pages 81–94, Brusells, Belgium, 2001. De-facto publications.
60. W. Duch, N. Jankowski, A. Naud, and R. Adamczak. Feature space mapping: a neurofuzzy network for system identification. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 221–224, Helsinki, Aug 1995.
61. W. Duch and J. Mandziuk. Quo vadis computational intelligence? In P. Sincak, J. Vascak, and K. Hirota, editors, *Machine Intelligence: Quo Vadis?*, volume 21, pages 3–28. World Scientific, Advances in Fuzzy Systems – Applications and Theory, 2004.
62. W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for understanding of data. *Proceedings of the IEEE*, 92(5):771–805, 2004.
63. R. O. Duda, P. E. Hart, and D.G. Stork. *Patter Classification*. J. Wiley & Sons, New York, 2001.
64. R.S. Michalski (ed). *Multistrategy Learning*. Kluwer Academic Publishers, 1993.
65. S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufmann, 1990.
66. J.H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.
67. K.S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, New York, 1982.
68. W. Gerstner and W.M. Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.
69. G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34:179–181, 2001.
70. A. Gifi. *Nonlinear Multivariate Analysis*. Wiley, Boston, 1990.
71. Ch. Giraud-Carrier, R. Vilalta, and P. Brazdil. Introduction to the special issue on meta-learning. *Machine Learning*, 54:197–194, 2004.
72. L. Goldfarb and D. Gay. What is a structural representation? fifth variation. Technical Report Technical Report TR05-175, Faculty of Computer Science, University of New Brunswick, Canada, 2005.
73. L. Goldfarb, D. Gay, O. Golubitsky, and D. Korkin. What is a structural representation? a proposal for a representational formalism. *Pattern Recognition*, (submitted), 2006.
74. L. Goldfarb and S. Nigam. The unified learning paradigm: A foundation for ai. In V. Honovar and L. Uhr, editors, *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, pages 533–559. Academic Press, Boston, 1994.
75. R.L. Gorsuch. *Factor Analysis*. Erlbaum, Hillsdale, NJ, 1983.
76. K. Grąbczewski and W. Duch. The separability of split value criterion. In *Proceedings of the 5th Conf. on Neural Networks and Soft Computing*, pages 201–208, Zakopane, Poland, 2000. Polish Neural Network Society.
77. K. Grąbczewski and W. Duch. Forests of decision trees. *Neural Networks and Soft Computing, Advances in Soft Computing*, pages 602–607, 2002.
78. K. Grąbczewski and W. Duch. Heterogenous forests of decision trees. *Springer Lecture Notes in Computer Science*, 2415:504–509, 2002.
79. M. Grochowski and N. Jankowski. Comparison of instance selection algorithms. ii. results and comments. *Lecture Notes in Computer Science*, 3070:580–585, 2004.

80. S. Grossberg. How does the cerebral cortex work? development, learning, attention, and 3d vision by laminar circuits of visual cortex. *Behavioral and Cognitive Neuroscience Reviews*, 2:47–76, 2003.
81. A. Gutkin. Towards formal structural representation of spoken language: An evolving transformation system (ets) approach. Technical Report PhD Thesis, School of Informatics, University of Edinburgh, UK, 2005.
82. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature extraction, foundations and applications*. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2006.
83. L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics, Springer-Verlag, New York, 2002.
84. H. Haas H. Jaeger. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.
85. S. Haykin. *Neural Networks - A Comprehensive Foundation*. Maxwell MacMillian Int., New York, 1994.
86. R. Hecht-Nielsen. Cogent confabulation. *Neural Networks*, 18:111–115, 2005.
87. G.E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:381–414, 2006.
88. V. Honavar and L. Uhr, editors. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. Academic Press, Boston, 1994.
89. G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17:879–892, 2006.
90. A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley & Sons, New York, NY, 2001.
91. S i. Amari and H. Nagaoka. *Methods of information geometry*. American Mathematical Society, 2000.
92. E.M Iyoda, H. Nobuhara, and K. Hirota. A solution for the n-bit parity problem using a single translated multiplicative neuron. *Neural Processing Letters*, 18(3):233–238, 2003.
93. J-S. R. Jang and C.T. Sun. Functional equivalence between radial basis function neural networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4:156–158, 1993.
94. N. Jankowski and W. Duch. Optimal transfer function neural networks. In *9th European Symposium on Artificial Neural Networks*, pages 101–106, Bruges, Belgium, 2001. Defacto publications.
95. N. Jankowski and M. Grochowski. Comparison of instance selection algorithms. i. algorithms survey. *Lecture Notes in Computer Science*, 3070:598–603, 2004.
96. N. Jankowski and V. Kadirkamanathan. Statistical control of growing and pruning in RBF-like neural networks. In *Third Conference on Neural Networks and Their Applications*, pages 663–670, Kule, Poland, October 1997.
97. N. Jankowski and V. Kadirkamanathan. Statistical control of RBF-like networks for classification. In *7th International Conference on Artificial Neural Networks*, pages 385–390, Lausanne, Switzerland, October 1997. Springer-Verlag.
98. C. Jones and R. Sibson. What is projection pursuit. *Journal of the Royal Statistical Society A*, 150:1–36, 1987.
99. M. Jordan and Eds. T.J. Sejnowski. *Graphical Models. Foundations of Neural Computation*. MIT Press, 2001.
100. V. Kadirkamanathan. A statistical inference based growth criterion for the RBF networks. In Vlontzos, editor, *Proceedings of the IEEE. Workshop on Neural Networks for Signal Processing*, pages 12–21, New York, 1994.
101. N. Kasabov. *Evolving Connectionist Systems - Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. Springer, Perspectives in Neurocomputing, 2002.

102. M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.
103. V. Kecman. *Learning and Soft Computing*. MIT Press, Cambridge, MA, 2001.
104. B. Kégl and A. Krzyzak. Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:59–74, 2002.
105. J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.
106. T. Kohonen. *Self-organizing maps*. Springer-Verlag, Heidelberg Berlin, 1995.
107. A. Konar. *Computational Intelligence. Principles, Techniques and Applications*. Springer, New York, 2005.
108. M. Kordos and W. Duch. Variable step search mlp training method. *International Journal of Information Technology and Intelligent Computing*, 1:45–56, 2006.
109. B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice Hall International, 1992.
110. L.I. Kuncheva. *Combining Pattern Classifiers. Methods and Algorithms*. J. Wiley & Sons, New York, 2004.
111. N. Kunstman, C. Hillermeier, B. Rabus, and P. Tavan. An associative memory that can form hypotheses: a phase-coded neural network. *Biological Cybernetics*, 72:119–132, 1994.
112. G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
113. Y.J. Lee and O. L. Mangasarian. Ssvm: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20:5–22, 2001.
114. M. Leshno, V.Y. Lin, Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.
115. H. Leung and S. Haykin. Detection and estimation using an adaptive rational function filters. *IEEE Transactions on Signal Processing*, 12:3365–3376, 1994.
116. H. Li, C.L.P. Chen, and H-P. Huang. *Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering*. CRC Press, 2000.
117. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1997 (2nd ed).
118. H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.
119. W. Maass and Eds. C. M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, Cambridge, MA, 1998.
120. W. Maass and H. Markram. Theory of the computational function of microcircuit dynamics. In S. Grillner and A. M. Graybiel, editors, *Microcircuits. The Interface between Neurons and Global Brain Function*, pages 371–392. MIT Press, 2006.
121. W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14:2531–2560, 2002.
122. R. Maclin. Boosting classifiers regionally. In *Proc. 15th National Conference on Artificial Intelligence, Madison, WI.*, pages 700–705, 1998.
123. C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.
124. P. Matykiewicz, W. Duch, and J. Pestian. Nonambiguous concept mapping in medical domain. *Lecture Notes in Artificial Intelligence*, 4029:941–950, 2006.
125. T.J. McCabe and C.W. Butler. Design complexity measurement and testing. *Communications of the ACM*, 32:1415–1425, 1989.
126. T.P. McNamara. *Semantic Priming. Perspectives from Memory and Word Recognition*. Psychology Press, 2005.

127. J.M. Mendel. *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice-Hall, 2000.
128. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*. Elis Horwood, London, 1994.
129. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2006)*, 2006.
130. K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.
131. M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.
132. S. Mitra and T. Acharya. *Data Mining: Multimedia, Soft Computing, and Bioinformatics*. J. Wiley & Sons, New York, 2003.
133. D. Nauck, F. Klawonn, R. Kruse, and F. Klawonn. *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons, New York, 1997.
134. A. Newell. *Unified theories of cognition*. Harvard Univ. Press, Cambridge, MA, 1990.
135. C.S. Ong, A. Smola, and B. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1045–1071, 2005.
136. S.K. Pal and S. Mitra. *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing Paradigm*. J. Wiley & Sons, New York, 1999.
137. Y.H. Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, Reading, MA, 1989.
138. E. Pękalska and R.P.W. Duin. *The dissimilarity representation for pattern recognition: foundations and applications*. New Jersey; London: World Scientific, 2005.
139. E. Pękalska, P. Paclik, and R.P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2:175–211, 2001.
140. T. Poggio and F. Girosi. Network for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497, 1990.
141. A. Pouget and T.J. Sejnowski. Spatial transformation in the parietal cortex using basis functions. *Journal of Cognitive Neuroscience*, 9:222–237, 1997.
142. A. Pouget and T.J. Sejnowski. Simulating a lesion in a basis function model of spatial representations: comparison with hemineglect. *Psychological Review*, 108:653–673, 2001.
143. M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation of Functions and Data*, pages 143–167, Oxford, 1987. Oxford University Press.
144. J.R. Rabunal and J. Dorado, editors. *Artificial Neural Networks in Real-life Applications*. Idea Group Pub, 2005.
145. R. Raizada and S. Grossberg. Towards a theory of the laminar architecture of cerebral cortex: Computational clues from the visual system. *Cerebral Cortex*, 13:100–113, 2003.
146. I. Roth and V. Bruce. *Perception and Representation*. Open University Press, 1995. 2nd ed.
147. D. Rousseau and F. Chapeau-Blondeau. Constructive role of noise in signal detection from parallel arrays of quantizers. *Signal Processing*, 85:571–580, 2005.
148. S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
149. Asim Roy. Artificial neural networks - a science in trouble. *SIGKDD Explorations*, 1:33–38, 2000.
150. Asim Roy. A theory of the brain: There are parts of the brain that control other parts. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN 2000)*, volume 2, pages 81–86. IEEE Computer Society Press, 2000.
151. D.E. Rumelhart and J.L. McClelland (eds). *Parallel Distributed Processing, Vol. 1: Foundations*. MIT Press, Cambridge, MA, 1986.

152. S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
153. E. Salinas and T.J. Sejnowski. Gain modulation in the central nervous system: where behavior, neurophysiology, and computation meet. *Neuroscientist*, 7:430–440, 2001.
154. R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.
155. B. Schölkopf and A.J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.
156. F. Schwenker, H.A. Kestler, and G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14:439–458, 2001.
157. A.K. Seewald. Exploring the parameter state space of stacking. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 685–688, 2002.
158. L. Shastri. Advances in shruti - a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11:79–108, 1999.
159. Wang Shoujue and Lai Jiangliang. Geometrical learning, descriptive geometry, and biomimetic pattern recognition. *Neurocomputing*, 67:9–28, 2005.
160. E. Simoncelli and B.A. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216, 2001.
161. P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36:59–83, 1999.
162. Anuj Srivastava and Xiuwen Liu. Tools for application-driven linear dimension reduction. *Neurocomputing*, 67:136–160, 2005.
163. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
164. R. Tibshirani T. Hastie and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.
165. J.D. Tebbens and P. Schlesinger. Improving implementation of linear discriminant analysis for the small sample size problem. *Preprint, submitted to Elsevier Science*, 2006.
166. R.F. Thompson. *The Brain. The Neuroscience Primer*. W.H. Freeman and Co, New York, 1993.
167. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
168. K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.
169. K. Tsuda and W.S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20:i326–i333, 2004.
170. K.P. Unnikrishnan and K.P. Venugopal. Alopex: a correlation-based learning algorithm for feeedforward and recurrent neural networks. *Neural Computation*, 6:469–490, 1994.
171. J.-P. Vert. A tree kernel to analyze phylogenetic profiles. *Bioinformatics*, 18:S276–S284, 2002.
172. S.F. Walker. A brief history of connectionism and its psychological implications. In A. Clark and R. Lutz, editors, *Connectionism in Context*, pages 123–144. Springer-Verlag, Berlin, 1992.
173. D.L. Wang. On connectedness: a solution based on oscillatory correlation. *Neural Computation*, 12:131–139, 2000.
174. A.R. Webb. *Statistical Pattern Recognition*. J. Wiley & Sons, 2002.
175. C. Wendelken and L. Shastri. Multiple instantiation and rule mediation in shruti. *Connection Science*, 16:211–217, 2004.
176. S. Wermter and R. Sun. *Hybrid Neural Systems*. Springer, 2000.

177. T. Wieczorek, M. Blachnik, and W. Duch. Influence of probability estimation parameters on stability of accuracy in prototype rules using heterogeneous distance functions. *Artificial Intelligence Studies*, 2:71–78, 2005.
178. T. Wieczorek, M. Blachnik, and W. Duch. Heterogeneous distance functions for prototype rules: influence of parameters on probability estimation. *International Journal of Artificial Intelligence Studies*, 1:xxx–yyy, 2006.
179. P.H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, third edition, 1992.
180. I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd Ed, 2005.
181. J.G. Wolff. Information compression by multiple alignment, unification and search as a unifying principle in computing and cognition. *Artificial Intelligence Review*, 19:193–230, 2003.
182. J.G. Wolff. *Unifying Computing and Cognition. The SP Theory and its Applications*. CognitionResearch.org.uk (Ebook edition), 2006. http://www.cognitionresearch.org.uk.
183. D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.