Learning from Examples, Agent Teams and the Concept of Reflection

Uwe Beyer Frank Śmieja

June 15, 1993

Abstract

Learning from examples has a number of distinct algebraic forms, depending on what is to be learned from which available information. One of these forms is $x \xrightarrow{G} y$, where the input–output tuple (x, y) is the available information, and G represents the process determining the mapping from x to y. Various models, $y = f(x)$, of G can be constructed using the information from the (x, y) tuples. In general, and for real-world problems, it is not reasonable to expect the exact representation of G to be found (i.e. a formula that is correct for all possible (x, y)). The modeling procedure involves finding a satisfactory set of basis functions, their combination, a coding for (x, y) and then to adjust all free parameters in an approximation process, to construct a final model. The approximation process can bring the accuracy of the model to a certain level, after which it becomes increasingly expensive to improve further. Further improvement may be gained through constructing a number of agents $\{\alpha\}$, each of which develops its own model f_α. These may then be combined in a second modeling phase to synthesize a *team* model. If each agent has the ability of internal *reflection* the combination in a team framework becomes more profitable. We describe reflection and the generation of a *confidence* function: the agent's estimate of the correctness of each of its predictions. The presence of reflective information is shown to increase significantly the performance of a team.

Keywords: Agent accuracy, approximation rate, reflection, confidence, team.

1 Introduction

"Learning from examples" is the construction of a model of a process using information obtained from observed instances of the process [6, 1]. There are various types of learning from examples, depending on what is to be modeled, the information that is available and the modeling assumptions that are made. In this paper we are interested in the class represented by the relation

$$x \xrightarrow{G} y \tag{1}$$

where the tuples (x, y) are the available information, and G represents the mapping reproducing it and which is to be discovered by the modeling process. Given is an ordered set of tuples (x, y), known as the learning set $\mathcal{L}$. The task of the model is to reproduce these and another set of tuples $\mathcal{T}$, known as the test set, using only the information in $\mathcal{L}$.

We define *approximation rate* as the observed improvement in model accuracy per example seen. This is expected to fall to zero with increasing number of examples seen, indicating that the model's approximation can, after a certain level, not be further improved upon. Improvements in this limiting value of accuracy have been sought in the use of modularization, team methods and model stacking. Such methods use a number of *agents*, each implementing a slightly different form of model, and combine their suggestions in differing ways to produce an overall model. All these methods seek improvements through adaptation at the team level, without using any additional information about the constituent agents themselves. We describe the concept of *reflection* within an agent, whereby the agent additionally estimates the quality of each of its answers with a *confidence* value. Teams of reflective agents (*reagents*) are shown to produce better improvements in accuracy than teams of non-reflective agents.

The paper has the following structure. In section 2 the general process of learning from examples is described. The limitation of an approximation is discussed in section 3. In section 4 the idea of agent reflection is introduced. Teams and cooperation are described in section 5. Results on an OCR problem domain using teams of reflective and non-reflective agents are presented in section 6.

2 Learning from examples

Learning from examples is concerned with development of models through the observation of facts contained in an external world. Facts are observed events occurring in the world as a result of causes present. It is the task of the model, as with a scientific theory, not only to reproduce the cause–effect pairs that have been observed, but also to make predictions of effects from causes that have not yet been observed.

Constructing a successful model of an observed process is dependent upon several related aspects. Consider the problem of modeling the horizontal distance traveled by a ball tossed under a gravitational field (see Figure 1). The phenomenon wished to be modeled is defined by the effects to be predicted: here the horizontal displacement of the ball, s. There are five major and related aspects of the modeling process that can be identified, we term them: distillation, recoding, instrumentation, combination and approximation (Figure 2).

The distillation D is the task of choosing the data (the causes) considered important to the phenomenon. One can envisage various available data that are not important to the phenomenon but are nevertheless to be observed in the world: present time of day, color of ball and cloud cover for instance. The choice of postulated causes from available facts must be performed somewhere within the modeling process. It is most frequently performed externally by the experimenter, who uses his expert knowledge to determine that (in the example) the parameters θ and v are suitable ones to use. A stand-alone model, which one would expect a robot to employ, must determine for itself, through observation of the process, which are the relevant parameters. This is not necessarily a trivial process.

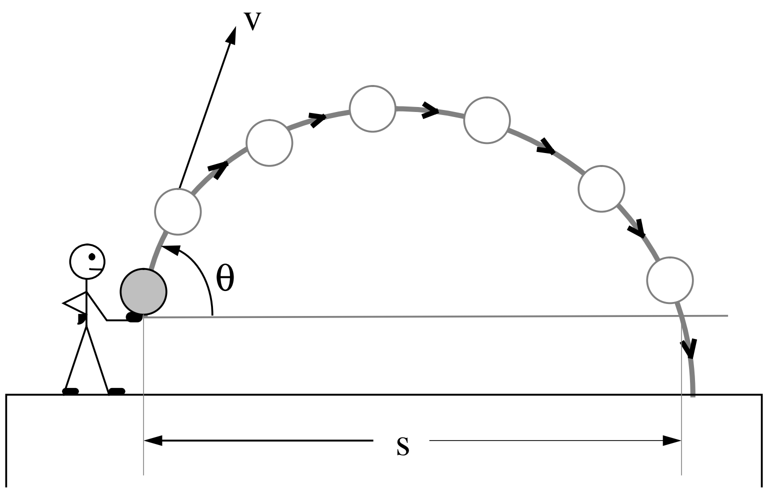

Figure 1: An example phenomenon observed in the world, which is to be modeled through learning from examples. The condition is the tossing of a ball, the phenomenon is the horizontal displacement s.

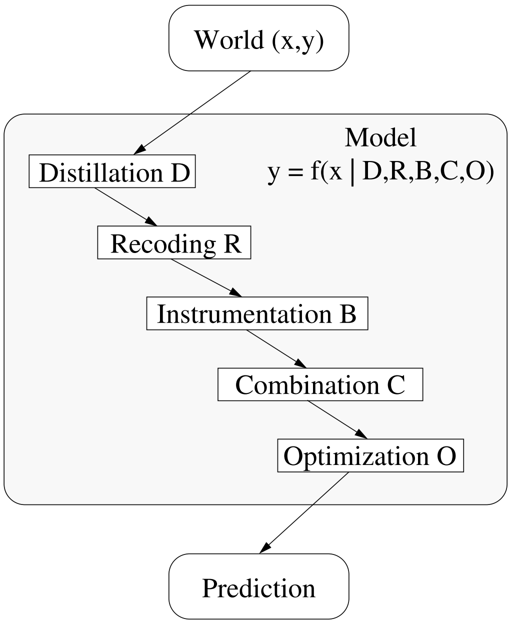

Figure 2: The five major considerations in constructing a model that learns from examples. The transformation from world to prediction consists of a number of stages, none of which is necessarily trivial.

Class name	General form	Tunable parameters $\{\lambda_i\}$
polynomial	$y = \sum_{k=0}^{n} a_k x^k$	a_k, n
hyperbolic	$y = \frac{a_0}{2} + \sum_{r=1}^{n}(a_r \cosh rx + b_r \sinh rx)$	a_r, b_r, n
harmonic	$y = \frac{a_0}{2} + \sum_{r=1}^{n}(a_r \cos rx + b_r \sin rx)$	a_r, b_r, n
probabilistic	$y = E(x) = \int_{-\infty}^{\infty} x\, g(x\|\sigma_1, \ldots, \sigma_n) dx$	$g, \sigma_1, \ldots \sigma_n$
rule-based	$y = g(x, P, \phi)$	P, ϕ

Table 1: In the form for probabilistic solutions, $g(x|\sigma_1, \ldots, \sigma_n)$ is the probability density function (p.d.f.), which includes the assumptions upon which the statistical model is based. For the rule-based solutions, P is the set of predicates chosen, and ϕ the set of production rules.

The recoding R involves transforming some or all of the chosen dependent variables into a form that enables easier modeling. It is related to distillation in that the recoding can be viewed as being part of a measuring instrument producing the observed data. In the example the use of $\sin\theta$ and/or $\cos\theta$ is to be recommended over θ. However, the modeling system must somehow recognize what is a reasonable recoding and what is not. For example, the coding $\exp(-\theta^2)$ is also possible, but is not helpful for this problem.

The instrumentation B describes the basis functions that are to be employed to fit the model to the observed data, and how they are to be combined. The possible functions can be neatly classified into five broad groups: polynomial, harmonic, hyperbolic, probabilistic [26] and rule-based (Table 1). The process B generates a number of free parameters $\{\lambda_i\}$ (general examples of which are indicated in the table).

The combination C represents the way in which the basis functions are combined to form the model's function f. We identify two standard mechanisms: multiplication and addition, or `AND` and `OR`, or sigma (Σ) and pi (Π). From these most expressions f can be constructed. The function f is then to be seen in general as a series of additions $\sum_i \mu_i g_i$ and multiplications $\Pi_i g_i$, which can in turn be combined in similar ways. The process C generates a number of free parameters $\{\mu_i\}$.

The resulting model's function f can be expressed as a function of x and the free parameters:

$$f = f(x; \{\lambda_i\}; \{\mu_i\}). \tag{2}$$

The parameters must be adjusted in order to fit the data $\mathcal{L}$. This represents the fifth consideration in the modeling process, the approximation or optimization process O (this includes the initialization of the free parameters). There are a range of methods that may be used for this optimization process, including simulated annealing, genetic algorithms [13] and neural networks. Neural networks normally represent methods employing harmonic or hyperbolic basis functions (a sigmoid is a hyperbolic construct), and generally Σ combinations. The parameters to optimize are the "weights", or $\{\mu_i\}$ above. The learning procedure is the method used to adjust these tunable parameters. The typical example is back propagation [19], although other methods are frequently employed, such as conjugate gradient, Newton–Raphson and second-order gradient descent.

The difficulty of the approximation task is dependent not only on the complexity of G, but also on the R, B, C and O used in the model. It seems therefore unreasonable to demand a universal definition of the "learnability" of an arbitrary G. If a constructive polynomial method is employed, a (for example) order 10 polynomial will be harder to approximate than a straight line, because the number of data points required to be seen to define the former is higher. However, if an order ≥ 10 polynomial is employed exclusively then the straight line will be harder to approximate. The question of model learnability should therefore be better expressed in terms of the progress of the actual accuracy of the model.

This section represents a sketch of the various factors that contribute to the actual modeling process. They lead one to the conclusion that as a result of the freedom present in the construction of a model, it is unreasonable to expect to find one that performs optimally. Instead, it is more likely that a combination of several models built in differing ways (varying any of D, R, B, C, and O) will produce a significantly better overall model. The problem that has to be solved is how to combine these models.

3 Modeling with a single agent

An **agent** is a general modeling process producing a y–value for each x–value, using the information in $\mathcal{L}$. When the tuples (x, y) are chosen by the agent itself (and not, for example, by a teacher) the process is called **exploration** [29, 11]. The following quantities can be defined for an agent.

Model accuracy. An agent is graded by the accuracy of its predictions. This may be estimated statistically by comparing a set of its predictions with events in the real world. The accuracy $A_\alpha(\mathcal{M})$ of an agent α tested on a set $\mathcal{M}$ of (x, y) tuples is defined:

$$A_\alpha(\mathcal{M}) := \frac{1}{|\mathcal{M}|} \sum_i^{|\mathcal{M}|} \delta(f_\alpha(x_i), y_i) \tag{3}$$

where $\delta(a, b)$ is defined: $\delta(a, b) = 1$ if $a = b$, and 0 otherwise. The learn accuracy $A_\alpha(\mathcal{L})$ is the degree to which an agent correctly reproduces the tuples it has already seen. The test accuracy ("generalization") $A_\alpha(\mathcal{T})$ is the degree to which the agent correctly models a new set of tuples $\mathcal{T}$.

Approximation rate. Let an agent's model after seeing t tuples be $f_\alpha(x, t)$, and the accuracy as a function of t be written $A_\alpha(t, \mathcal{M})$. Then the approximation rate $R(\alpha, t, \mathcal{M})$

is defined as:

$$R(\alpha, t, \mathcal{M}) := A_\alpha(t, \mathcal{M}) - A_\alpha(t-1, \mathcal{M}) = \frac{dA_\alpha(t, \mathcal{M})}{dt}. \tag{4}$$

A negative approximation rate indicates the agent's predictions are becoming worse, a positive rate indicates they are becoming better, and a steady near-zero rate indicates convergence of the approximation process to an asymptotic value of accuracy, $A_\alpha(\infty, \mathcal{M})$. In practice, the approximation rate may more usefully be expressed as a function of accuracy, and be estimated by:

$$R^\star(A) = \frac{A_\alpha(t_2, \mathcal{M}) - A_\alpha(t_1, \mathcal{M})}{t_2 - t_1} \tag{5}$$

which is permissible for small changes in accuracy over the interval (t_1, t_2).

Conjecture 1 *When the adaptation is stable the approximation rate of an agent becomes a generally decreasing function of the number of (x, y) tuples seen.*

This principle applies to many real-world problems, such as optical character recognition (OCR). Its says the following: *the better the performance of a modeling system, the more new tuples that must be seen in order to increase this performance further.*

The effect is readily seen in the competitive field of OCR. The most successful systems on the market have been developed from data-banks containing an order of 100,000 exemplars *per character*. Thus, a reasonably successful system recognizing hand-printed letters would typically have a database containing a couple of million exemplars. Figure 3a shows this effect for an upper-case letter recognition agent. The agent employed pixel coding (16×16 input patterns) with a sigmoid-based feed-forward neural network, and the approximation was carried out using the back-propagation procedure [19]. The test set used to measure the accuracy was held constant, and was disjunct from all the learn sets. The curve shows the best accuracy achieved as a function of the number of exemplars used in the approximation. In this plot it is not clear when the accuracy will reach one. The growth is clearly extremely slow after a (problem-dependent) threshold number of patterns has been seen. Plotted alongside in Figure 3b is the corresponding approximation rate $R^\star(A)$ as a function of accuracy.

The consequences of Conjecture 1 can be dealt with either by "brute-force" learning (which is with today's computer technology quite a reasonable alternative), or in an architectural manner. The professional OCR systems go about it in a brute-force way. With teams and reflection we put forward an alternative which allows significant improvement on single agent performance.

4 Reflection

Reflection is a method by which an agent observes its own behavior and produces additional information as a result (see Figure 4). This means using the information from the learn set and the agent's performance on the learn set to produce predictions as to the agent's competence on a further x. The process is *dynamic*: observation of behavior can

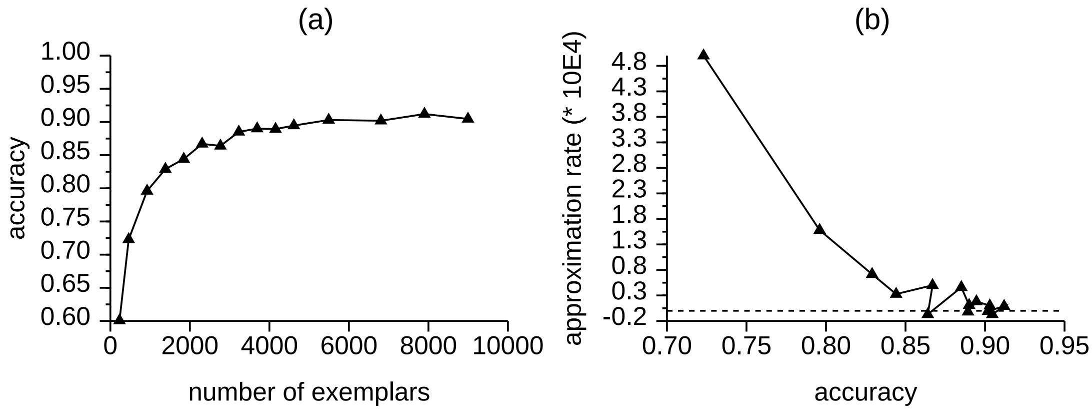

Figure 3: (a): Dependence of accuracy for a particular agent on the number of (upper-case letters) exemplars used in the adaptation. Each point is associated with a statistical error of around ± 0.005 to ± 0.01. (b): The corresponding plot of approximation rate as defined in eqn. (5) (the dotted line marks zero approximation rate).

be performed incrementally with each new tuple seen. The agent becomes, with this extra reflective element, a reflective agent, or *reagent*. The information produced by reflection we have denoted by c in Figure 4. It is called **confidence** and is defined in the following way.

Def 1 (confidence) *The confidence function $c_\alpha(x,t)$, $0 \leq c_\alpha(x,t) \leq 1$, is a model of the probability that the prediction $f_\alpha(x,t)$ for agent α is correct for an input x:*

$$c_\alpha(x,t) \models P(f_\alpha(x,t) = y). \tag{6}$$

The function $c_\alpha(x,t)$ is a function of x and thus attempts to establish the zones of competence of the agent α in the x–space. The function is adaptable and is fitted to the learn set data $\mathcal{L}$. The confidence can be seen as a generalization of a rejection function [24], which could be constructed from confidence by introducing a threshold confidence value above which the agent's prediciton is accepted and below which it is rejected. We will show in section **??** how such a continuous-valued rejection function such as confidence provides extra information about the agent's model, which can be used to compare and combine a number of agents. One may alternatively interpret confidence as an indication as to the amount of "belief" [20] that should be placed in the current approximation.

Eqn. (6) states that reflection is the generation of an inner model of the reliability of the modeling process itself. It differs from the modeling processes described before in that there is no tuple $(x, P(f_\alpha(x) = y))$ that is available from the world for fitting the function $c(x)$: every tuple (x, y) that is learned has by definition been perfectly learned, and consequently $P(f_\alpha(x) = y) = 1$! So what does it mean, the *probability* that the answer to a question x will be correct?

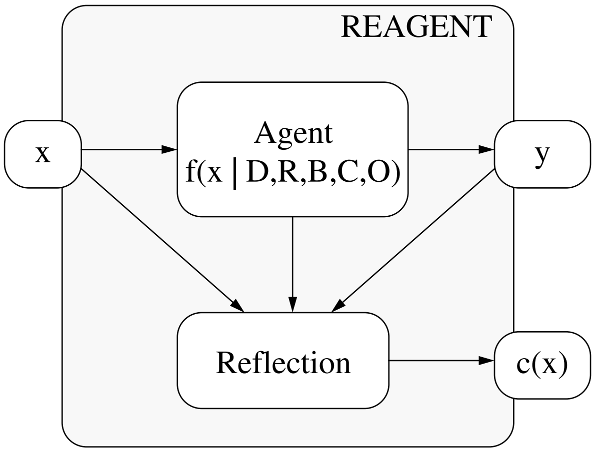

Figure 4: Reflection

The actual probability $P^\star$ that a prediction for x is correct is defined in the range 2ϵ about the point x for a particular test set $\mathcal{M}$:

$$P^\star(x,\epsilon) = \int_{x-\epsilon}^{x+\epsilon} \delta_d(f_\alpha(x,t), y)\, dx, \tag{7}$$

where all (x, y) are tuples in $\mathcal{M}$, and $\delta_d(\cdot)$ is the Dirac delta function. The value ϵ must be defined by the experimenter, and represents the size of the region about x in which the function f is considered to be smooth.

The difficulty with the calculation expressed in eqn. (7) is that the variable x (and the region ϵ) is a (high dimensional) vector, and is generally discrete. If we could calculate the true probability $P^\star$ we would have the desired confidence model. If we can achieve this then $P^\star$ should be used as the confidence function itself. It is generally extremely difficult to determine $P^\star$: that is why a model is sought.

We are left with the two questions:

1. How can a reasonable confidence be feasibly generated?
2. What criterion—to be tested quantitatively—must it fulfill in order that it be seen as a useful quantity?

We describe first a simple criterion for testing confidence, and then the ways in which it may be generated.

4.1 Confidence reliability

In order to measure the appropriateness of a confidence function, the only simple information available are facts that the approximation is correct or incorrect for each tuple in the given test set $\mathcal{M}$. One cannot independently work out a "true" probability that the given tuples will be correctly reproduced. One must, therefore, use the binary information per tuple, of whether it is correctly or incorrectly modeled by the agent, and compare this with the continuous confidence value, to produce an accuracy measure for the confidence function. The only objective way we see of doing this, which has relevance for teams, is to check whether the confidence, when used as a rejection indicator, produces reliable enough results. Thus, a standard threshold value of confidence should be chosen: we chose 0.5 as the value above which a confidence should guarantee mostly correct answers. The reliability ρ is then calculated as follows:

$$\rho_\alpha(\mathcal{M}) := \frac{1}{\mathcal{M}'} \sum_{i \in \mathcal{M}'} \delta(f_\alpha(x_i), y_i) \quad \text{where} \quad (x_i, y_i) \in \mathcal{M}', \quad \text{and} \tag{8}$$

$$\mathcal{M}' := \{(x, y) : (x, y) \in \mathcal{M} \wedge c_\alpha(x) > 0.5\} \tag{9}$$

Table 3 in section 6 below displays examples of confidence reliability for various reagents. It can be seen quite clearly that agent 8 is particularly bad, and should consequently not be used in a team (see later). All the other agents are about 0.9 reliable (even the one with accuracy 0.68).

4.2 Two classes of confidence

One may classify the confidence function in terms of the set of tuples used to generate it. If the set used is $\mathcal{L}$ (that used to generate f_α), the function c_α is called **learn confidence**. If the confidence training set is disjunct from $\mathcal{L}$, the function c_α is called **calibration confidence**. We consider in this paper only the learn confidence functions. The calibration functions require use of extra data, and it is questionable whether that extra data, which by definition may *not* now be used to improve f_α, would have been better used to do just that. It is an expensive and non-dynamic way of generating a confidence function. The idea itself is in a similar vogue to "bootstrap" methods used by statisticians [4] for estimating sampling distributions. However, here the information is only used to provide a global confidence in an estimation process, and not an individual confidence for every subsequent estimate produced by the process.

4.3 Construction of learn confidence functions

4.3.1 Example 1: Geometrical reasoning

Consider an agent using the k–nearest neighbor (kNN) [2] method to form a discrete function discriminating m classes. The confidence can be produced by geometric reasoning [15] in the following way. An input question x is compared with the $\{x_i\}$ stored during the learning process. Each exemplar i has associated with it an output y_i. In a classification process there are a finite number of different outputs. The classification of x is influenced only by the k exemplars that are nearest to it in input space, using a euclidean metric (see Figure 5). A number of possible classes are represented in these k nearest neighbors of x, and the

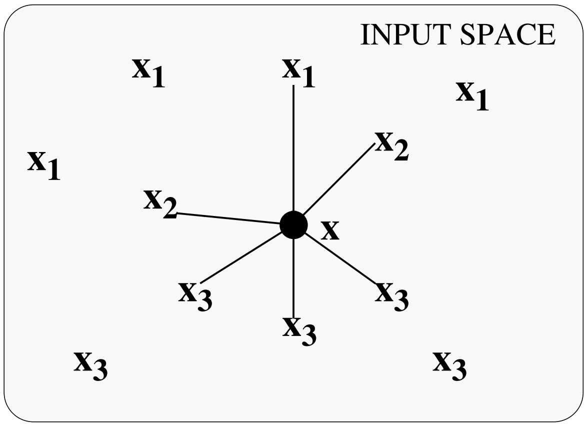

Figure 5: Choosing a winning class in the k–nearest neighbor procedure. Here the number of neighbors is $k = 6$. The subscript on the patterns in the picture indicates the class they represent.

confidence for the chosen class l is given by:

$$c_l(x) = \frac{\Phi_l(x)}{\sum_k \Phi_k(x)} \tag{10}$$

where the summation is over all classes in the neighborhood, and the "potential" of a class k given the input x, $\Phi_k(x)$, is defined

$$\Phi_k(x) = \sum_{\substack{i \in \mathcal{N} \\ y_i \equiv k}} \frac{1}{\|x - x_i\|}, \tag{11}$$

where (x_i, y_i) is a stored tuple. $y_i \equiv k$ indicates y_i must represent the class k. k. The summation is over the nearest neighborhood $\mathcal{N}$ of stored tuples.

When the problem is not discrete (like classification), but continuous, the process can be seen as 1-nearest neighbor, where each different stored tuple is considered a (former) class, and the summation in eqn. (11) is non-existent.

4.3.2 Example 2: Feed-forward neural network

A feed-forward neural network with real output nodes naturally produces confidences for a classification task [7, 16]. The confidence is simply formed by extending the discrete class answer to a set of relative probabilities between classes. Thus if a neural network agent produces an output vector $s(x)$ (see Figure 6), the confidence $c_k(x)$ for the class k is given by:

$$c_k(x) = \frac{s_k(x)}{\sum_i s_i(x)} \tag{12}$$

For a non-classification task, there is no obvious way in which confidence may be derived from a feed-forward network.

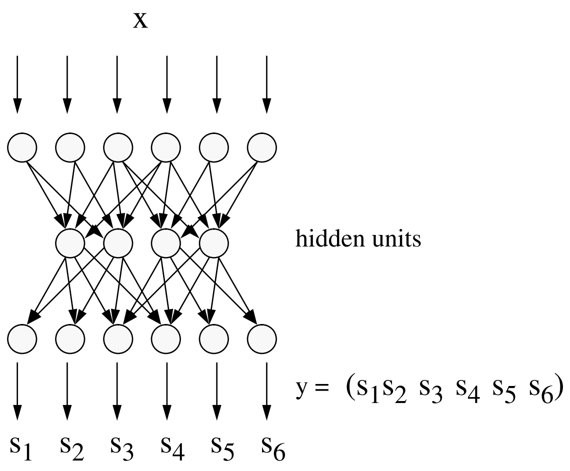

Figure 6: Operation of a feed-forward neural network.

4.3.3 Example 3: MINOS reagent

A MINOS reagent [21, 25] is constructed from two agent modules, called Worker and Monitor (see Figure 7). The Worker learns the mapping $f_\alpha(x)$ and the Monitor learns the confidence $c_\alpha(x)$. The Worker is generally taken to be a feed-forward neural network. The MINOS reagent forms part of a *Pandemonium* system of reagents, each of which gets allocated different (x, y) tuples to learn. The Monitor keeps tabs on the tuples that have been allocated to be learned by the Worker. If it is a feed-forward neural network, the Monitor learns incrementally to map an x to $\{0, 1\}$, depending on whether or not the tuple (x, y) is being learned by the Worker. Thus as each new (x, y) tuple is seen, the confidence is adapted. See [22] for further information about how the Pandemonium system performs on particular problem domains.

The advantages of such a MINOS reagent is that confidence can be produced whether the problem is discrete or continuous, and that the procedure is adaptive but not directly dependent on the actual process modeling f (the Worker). It is most useful for problems that have clear discontinuities, or that require "discrete modularization" (this term is explained in section 5.2). For problems such as OCR, where classes overlap quite significantly, the confidence modeling process is at a disadvantage when it is not strongly linked to the f modeling process, as in the previous two examples. In such cases a Pandemonium system will produce reliable rejection information (i.e., when it is confident an answer is correct, it will be correct with high accuracy), but require large rejection rates (since rejection tends to throw out those patterns lying on fuzzy boundaries). Unfortunately, a Pandemonium will have a larger "don't know" area mainly in those regions where the patterns overlap. We use such a system in section 6 as part of a team, and it can be seen that the information from the MINOS agents for the "don't know" areas, although not useful for a single agent, become useful in a team, increasing its performance.

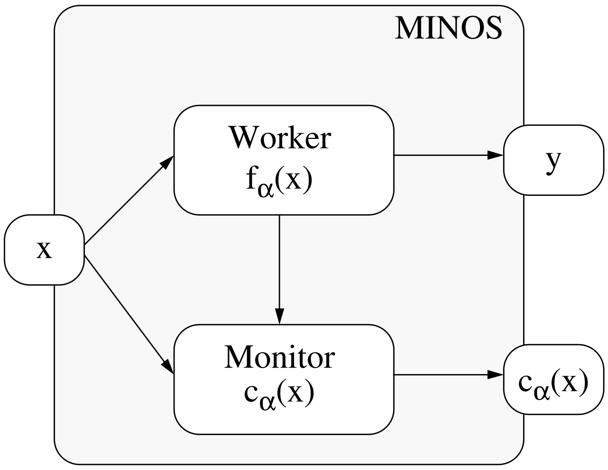

Figure 7: The MINOS reagent. The Worker neural network learns the function $f_\alpha(x)$ for those (x, y) tuples it has been allocated. The Monitor neural network learns to produce a 1 for those x learned by the Worker, and a 0 for those x not learned by the Worker.

4.4 The benefits of confidence

A good confidence function can be profitably employed for a pattern rejection technique. Rejection methods [24] are widely used in all serious pattern recognition applications, and it is implicitly assumed that the confidence–like principle on which the method must operate obeys to a greater or lesser extent the condition expressed below:

$$A_\alpha(\mathcal{M}') \gg A_\alpha(\mathcal{M}), \quad \text{where} \quad \mathcal{M}' = \{(x, y) \,|\, c_\alpha(x) > \kappa,\ (x, y) \in \mathcal{M}\}, \tag{13}$$

where $\kappa \in [0, 1]$ is an arbitrarily chosen threshold value. In other words, the accuracy based on the test set defined by those patterns not rejected should be significantly higher than that based on the test set including all patterns. (The confidence reliability used $\kappa = 0.5$).

Confined to the domain of the single reagent, confidence serves to increase the *reliability* of the approximator [8, 17]. Overall accuracy is, however, not increased in the reflection process. The question remaining is how to increase the overall accuracy of a model. As mentioned in section 3, we see two ways: either brute-force learning or a modular architecture. We describe the latter now within a team framework.

5 Teams and cooperation

An agent model $f(x|D, R, B, C, O)$ depends on many factors. It is not to be expected that for a particular problem, such as OCR, all can be chosen together such that f is optimal. One would expect different prescriptions for say R, B, and C to deal with slightly differing aspects of the process to be modeled. Additionally, the nature of the approximation iteration, O, will lead to characteristic forms of solution. (As an example, neural network

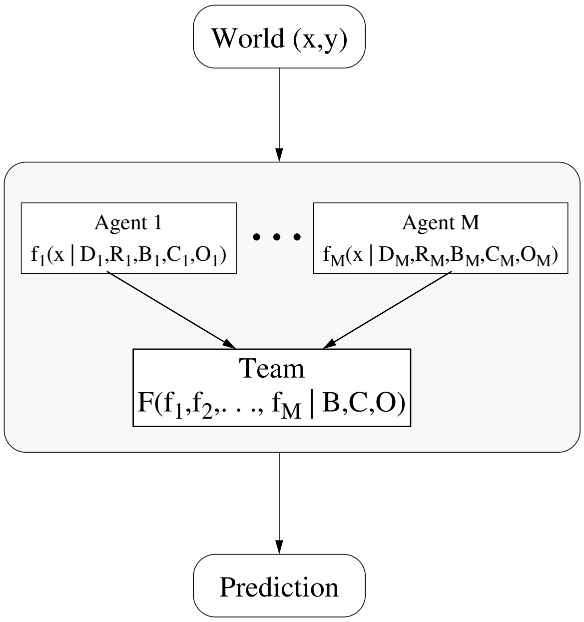

Figure 8: Non-reflective team.

construction algorithms might be expected to produce solutions with more "exceptions" than the standard back-propagation algorithm [23]).

For these reasons, one might envisage a combination of agents with different models to allow better modeling of the process than a single agent. In order to enable such a process, one must decide on the way in which the predictions from the various agents are to be combined. There are two broad classes of teams: non-reflective and reflective. Non-reflective teams do not use the reflective information from reagents, while reflective teams do.

5.1 Non-reflective teams

A non-reflective team is shown schematically in Figure 8. The approach can be viewed as constructing a team model F from a set of sub-models $\{f_\alpha\}$. The team agent is constructed in a similar way to the member agents, but does not consider the recoding (R) and distillation (D) stages of the modeling process. Thus it is sufficient to represent its model as $F(f_1, f_2, \ldots, f_M|B, C, O)$, where M is the number of agents in the team and B, C and O are the model parameters used in the construction of the team agent.

There exist in the literature various team architectures employing non-reflective agents [9, 18, 10, 28, 27]. Some are built during the construction of the individual agent models themselves, thus involving two levels of model (agent and team) being simultaneously optimized. Such a method avoids the generation of confidence functions from the agents by constructing the team on a priori knowledge. It is a possibility, but is likely to be more complicated than an independent agent model construction technique. Other non-reflective team models are

Class of team	Non-adaptive	Adaptive
Non-reflective	Majority vote	Simultaneous agent and team optimization
	Static modularization of input space	Stacked Generalization
Reflective	Winner–takes–all	Stacked Generalization

Table 2: Types of team architectures.

built using pre-determined specialization regions. They may be trained in an independent way, but no synergetic effect may be gained: the advantage is purely one of being able to find a team model through modularization of the input space quicker than a single agent model of the entire space. This may lead to quicker solutions for certain problems, but is not so promising for performance on the test set (what is to be done at the boundaries between agents?). Waibel [27] deals with this problem by cementing the cracks with a "connectionist glue", whereby the whole system is further adapted at the team level after the agents have been combined. This is similar to the general "stacked generalization" strategy [28] of second stage adaptation at the team level, except in the latter the constituting agents are not adapted further. The Waibel method is only then feasible when the constituting agents are all of the same type (e.g. neural networks). Table 2 summarizes the basic team strategies.

Stacked generalization [28] is in general the method of second phase learning. The agents used in the team are held fixed, and a further agent learns to associate their collected answers with the actual answer, using a further learn set $\mathcal{C}$, known as the calibration set.

Another model is the majority vote. This works only for classification problems, where the output y may be one of a finite set of classes $(l_1, l_2, \ldots, l_m)$. In this model the agents are trained independently beforehand, and the model accuracy, $A_\alpha(\mathcal{M})$ for each member of the team is noted. The model is simply:

$$F(f_1, f_2, \ldots, f_M) = \begin{cases} l_k & : \sum_\alpha \delta(f_\alpha, l_k) > \sum_\alpha \delta(f_\alpha, l_j), \; \forall\, j \neq k \\ f_\alpha & : A_\alpha(\mathcal{M}) > A_\beta(\mathcal{M}), \; \forall\, \beta \neq \alpha \text{ otherwise} \end{cases} \tag{14}$$

Thus that classification is accepted for which the majority of agents is of the same opinion. When there is no clear majority, the classification from the agent with the highest accuracy on the calibration set $\mathcal{M}$ is taken. This requires the employment of a calibration set for the accuracy calculation. One may forgo this by using a random mechanism to choose the second case in eqn. (14), but then team improvement is by no means guaranteed. A further disadvantage of the majority team architecture is that when there are a number of similar agents (whose predictions generally coincide) there is no way of allowing a more expert agent to veto their combined decision. We show in section 6 how the use of confidence allows this in a combined majority and winner–takes–all team.

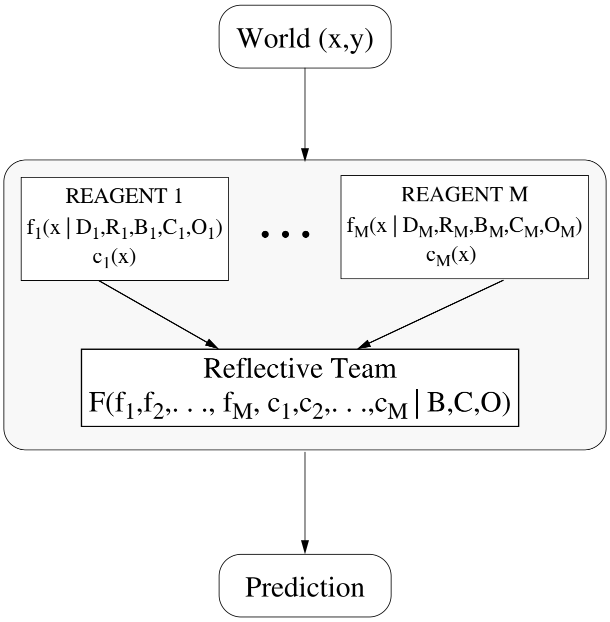

Figure 9: Reflective team.

5.2 Reflective teams

A reflective team is shown schematically in Figure 9. Here the team model F is constructed in a similar way to the non-reflective team, but receives as input the additional confidence information: $F(f_1, f_2, \ldots, f_M; c_1, c_2, \ldots, c_M|B, C, O)$, where M is the number of agents in the team.

Closest to our approach using confidence are the approaches based on probability and evidence, for classification problems. Here the agents produce real decision values of whether an input x represents a class k, which are transformed into a posteriori probabilities $p(k|x)$ that the class is k. Sets of classifiers can be combined using such probabilities [12, 5], but generally in a non-adaptive way. The parameters needed to be determined for the combination are fixed statistically or by using evidence [3] or belief [20].

In order that a team work, its constituting members should possess a minimum level of confidence reliability. Otherwise the areas where the answer from one agent should better be used than that of another will only rarely be found, and the desired synergy will not be attained. The objective of a single reagent outside of a team is to produce a highest possible model accuracy $A_\alpha(\mathcal{M})$. On the other hand, the objective for a agent acting as a team member is to produce an f_α and c_α such that the team model accuracy $A(\mathcal{M})$ is improved through its addition. This does not necessarily imply that $A_\alpha(\mathcal{M})$ must be maximized, but that the joint $(A_\alpha(\mathcal{M}), \rho_\alpha(\mathcal{M}))$ be optimal. The synergetic team effect can ensue from distinct specializations of the constituting reagents. This is why the confidence should be a function of x, and not a scalar.

The reflective team model constructed generally depends on the structure of the x–space covered by each agent. A simple model such as "winner–take–all" (WTA), or, in our case, "most confident takes all" will work well if the overlap of the agent sub-spaces is negligible. The team model is in this case called a **discrete modularization**, and is given by:

$$\begin{aligned} F(f_1, f_2, \ldots, f_M, c_1, c_2, \ldots, c_M) &= f_\alpha \quad \text{for tuple} \quad (x, y) \\ \text{where} \quad c_\alpha(x) &> c_\beta(x) \quad \forall \beta \neq \alpha \end{aligned} \tag{15}$$

If no knowledge is available regarding the statistical properties of the $f_\alpha(x)$, then there is also no theoretical clue as to how to combine them using F. In order to construct a team from a set of perhaps incompatible agents (i.e. they produce confidences that are not absolutely comparable), one can either estimate the calibration factor between each of the agents, using a calibration set and statistics (this is in end effect what the statistical and evidence methods do in order to generate comparable probabilities), or one can learn to reproduce the calibration set using an model F which is adaptable. This is then a stacked generalization method, but this time the answer from the agents is associated together with their confidence information to the actual answer in the calibration set.

6 Experimental results with teams

The various team architectures were compared using an OCR problem: recognizing handwritten upper case letters. The exemplars were obtained from the National Institute of Science and Technology (NIST) database [14]. The scanned images were transformed to 16x16 dimensional gray-scale x values with 26 dimensional N–class output coding for the associated y values. This determined the distillation D for all agents.

The learn set $\mathcal{L}$ consisted of 4616 tuples randomly chosen from the database. The test set $\mathcal{T}$ and calibration set $\mathcal{C}$ were similarly chosen. The test set had 4363 tuples and the calibration set 4379. All sets were disjunct from one another. All agents were trained with $\mathcal{L}$. All trainable teams were trained with $\mathcal{C}$. Accuracy of single agents and final teams was calculated using $\mathcal{T}$.

The agents could differ in the recoding R_α, which comprised grey values, black and white values, stroke information, contour information, features or simple counting of line and column values in the 16x16 image. Or they could differ in the adaptation method A_α. Possibilities included feed-forward neural network using back-propagation, MINOS/Pandemonium and kNN. The neural networks had 100 hidden nodes, and 3 MINOS agents were used in the Pandemonium system, with 30 hidden nodes for the Workers and 50 for the Monitors.

The team agent used for the adaptable teams was of the kNN type. Table 3 shows the individual accuracies for the agents using the test set $\mathcal{T}$.

In Figure 10 the progress of team accuracy $A(\mathcal{T})$ is shown as each new team member is included. The order in which team members were added was that of their decreasing individual accuracies (for the set $\mathcal{T}$). Since it is a (yet another) optimization problem to discover the best set of agents to use for any given team, the order we have chosen should

R_α	gray			b/w	line/column	stroke	feature	contour
α	Neural net	MINOS	kNN	kNN	kNN	kNN	kNN	kNN
$A_\alpha(\mathcal{T})$	0.884	0.851	0.851	0.844	0.683	0.336	0.859	0.870
Order in team	1	4	5	6	7	8	3	2
$\rho_\alpha(\mathcal{C})$	0.907	0.917	0.953	0.943	0.912	0.362	0.960	0.931

Table 3: Properties of the individual team members α and the order in which they were included in the team.

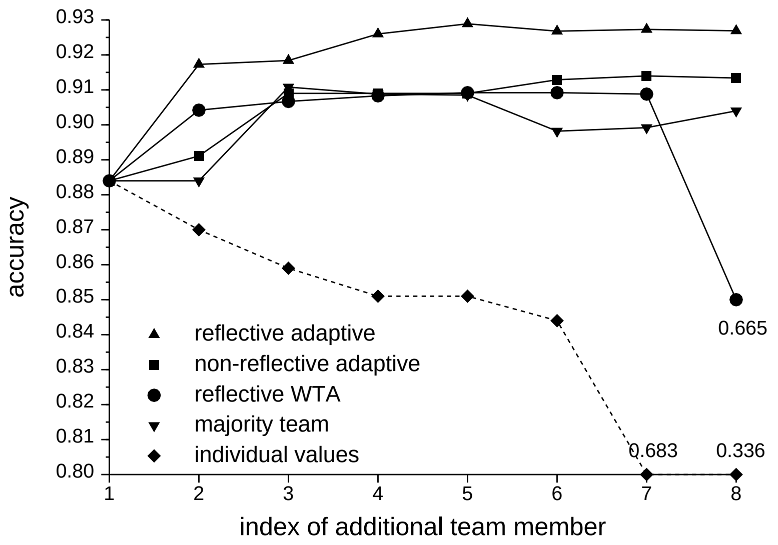

Figure 10: Progress of team accuracy as each new agent is included (solid lines), for reflective and non-reflective adaptive, and majority and WTA team methods. The dotted line shows the individual agent accuracies. The final two agents have much lower accuracies, the value of which is explicitly written.

not be taken to have any special significance. In this study we wish to compare performance among teams as a function of the number of agents used, and some kind of order must be chosen. The dotted curve shows the individual $A_\alpha(\mathcal{T})$ for each added agent, and the solid curves the team accuracy for the different strategies. The best individual agent accuracy was 0.884. We make the following observations:

- The reflective adaptive team was found to reach a peak accuracy of 0.929, and an accuracy of 0.927 when the team was complete.
- The non-reflective adaptive team reached a peak accuracy of 0.914, and a final accuracy of 0.913.
- The majority team reached a peak accuracy of 0.911 and a final value of 0.904.
- The reflective winner–take–all (WTA) team reached a peak accuracy of 0.909 but fell to 0.665 on addition of the final (unreliable) eighth agent. Use of a small calibration set would normally immediately filter such destructive elements out of a team using the confidence reliability as a measure. The final value of accuracy would then be 0.909.

One can express the results in terms of percentage increase for a full team relative to the best single agent accuracy: reflective adaptive 4.9%, non-reflective adaptive 3.3%, non-reflective majority 2.3%, reflective WTA 2.8%.

The improvements are all most noticeable in the first member addition. The first members are, however, the best ones, and provide the most varied information: neural network with pixel information and kNN with stroke information. Further team members are of diminishing variety of information. In the adaptive teams it can be clearly seen how the MINOS (agent 4) provides additional useful information although, as pointed out earlier, it cannot cope with its "don't know" areas by itself.

The last few (poorer) agents do not significantly disturb the reflective team behavior, whereas they are in general detrimental to the non-reflective methods. The additional influence of reflection information allows a statistically significant improvement for the OCR problem. The information has been used to dig more correct classifications out of the fuzzy regions between classes, than could otherwise be done with a posteriori error correcting (i.e. adaptive non-reflective team).

A major consideration with team methods is the calibration set $\mathcal{C}$. How large must the calibration set be? How long does the second phase of adaptation take? How would a single agent fare when shown the complete information apparently available to the experimenter, i.e. $\mathcal{L} \cup \mathcal{C}$ as learn material? In Figure 3a the single agent reached an accuracy of 0.905 using all the available information (the larger learn set was made up of $\mathcal{L} \cup \mathcal{C}$). The adaptive team methods managed to improve significantly on this (especially the reflective method), but the non-adaptive did not. However they did also not use a great deal of information from $\mathcal{C}$. We would like to show that a non-adaptive team better than a large single agent may be built using only $\mathcal{L}$ and a calibration set large enough to show that agent 8 is undesirable. The reflective WTA is the only method that uses such minimal information. The majority method requires a decision process to determine what is to be done when there is no clear

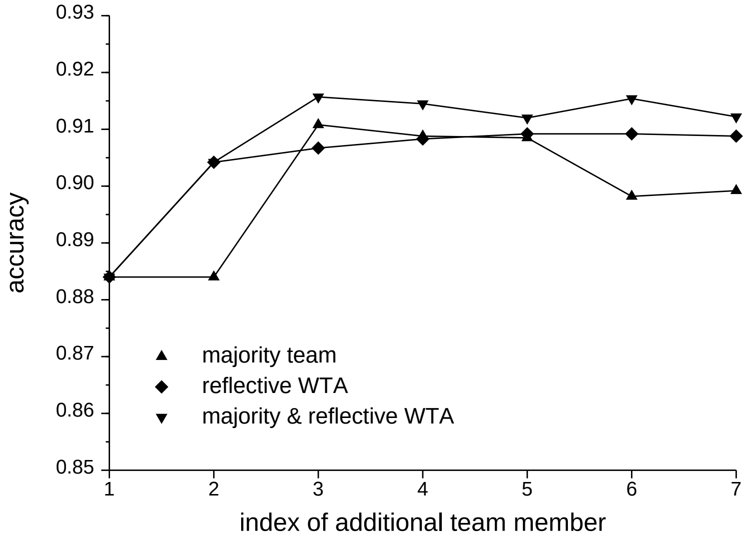

Figure 11: Comparison of three types of non-adaptive team.

majority (eqn. (14)), and it is important that it is the good agent whose answer is chosen. We therefore built a team based on majority and reflective WTA methods (it is valid, of course, just for classification problems). The method uses majority decision rule when the majority is absolute (more than 50% of the votes). When it is not absolute the WTA answer is used. The results of this method, which only requires the extra information that filters out the destructive agents from the team, are shown in Figure 11. The peak accuracy for the majority and WTA combined team was 0.916, with a final accuracy of 0.912. This compares favorably with the non-reflective adaptive team. The percentage increase relative to the best single agent is 3.2%. Thus a similar performance was reached although *only the learn set $\mathcal{L}$ was used.*

7 Conclusion

We have described the process of learning from examples in terms of real modeling considerations, and have broken the process down into five main stages: distillation, recoding, basis function choice, type of combination of the basis functions and parameter optimization. In general the freedom in choosing the type of model, for any specific process evidenced purely by observable data resulting from the process, must lead to the conclusion that finding the "perfect" agent is extremely unlikely. For this reason we suggested that team methods would tend to be more promising and perhaps cheaper in the long run. There are certain problems involved with team architectures, however, namely expensive double modeling stages and double-sized learn sets.

As a solution to this the concept of reflection was introduced in terms of a confidence function, whose task it is to generate an estimate of the probability of success of any particular prediction made by the associated agent. This information can then be used at the team level, either as a winner–take–all indicator (not requiring adaptation in a second modeling phase) or as extra information for an adaptive phase. Both methods were shown to result in significant improvements over single agent and non-reflective team methods. Team methods are in general most useful when a data-intensive approach is impractical (e.g. when the data is simply not there). Due to the cost of training many agents, one would generally express a preference for simple but numerous team members, rather than a few highly well-trained agents. The former approach we see as particularly relevant for the development of real-world (real-time) multiple agent systems.

Acknowledgements

The authors thank Heinz Mühlenbein, Jörg Kindermann, Gernot Richter, Dirk Schlierkamp-Voosen and Byoung-Tak Zhang for reading and correcting the draft, and for inspiring discussions. Thanks also go to Igor Cherednichenko for his contour filter used in the OCR experiment. This work is part of the GMD Adaptive Systems REFLEX project, supported by BMFT grant number 01IN111A/4.

References

[1] U. Beyer and F. J. Śmieja. Quantitative aspects of data-driven information processing. Technical Report 732, Gesellschaft für Mathematik und Datenverarbeitung, St Augustin, Germany, March 1993.

[2] Belur V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamos, 1986.

[3] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[4] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[5] J. Franke and E. Mandler. A comparison of two approaches for combining the votes of cooperating classifiers. In *Proceedings of the 11th IAPR*, pages 611–614, Den Haag, Netherlands, 1992.

[6] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[7] J. B. Hampshire and B. A. Pearlmutter. Equivalence proofs for multi-layer perceptron classifiers and the bayesian discriminant function. In Touretzky, Elman, Sejnowski, and Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, CA, 1990. Morgan Kaufmann.

[8] S. Hubrig-Schaumburg. Handwritten character recognition using a reflective modular neural network system. Master's thesis, Bonn University, Germany, 1992.

[9] R. A. Jacobs and M. I. Jordan. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991.

[10] K. Joe, Y. Mori, and S. Miyake. Construction of a large-scale neural network: simulation of handwritten Japanese character recognition on NCUBE. *Concurrency, Practice and Experience*, 2(2):79–107, June 1990.

[11] A. Linden and F. Weber. Implementing inner drive by competence reflection. In *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior, Hawaii*, 1992.

[12] E. Mandler and J. Schürmann. Combining the classification results of independent classifiers based on the Dempster–Schafer theory of evidence. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition and artificial intelligence*, pages 381–394. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1988.

[13] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: Continuous parameter optimization. *Evolutionary Computation*, 1, 1993.

[14] National Institute of Standards and Technology. OCR-system conference. Technical Report NISTIR 4912, NIST, Gaithersburg MD, August 1992.

[15] Stephen Omohundro. Geometric learning algorithms. Technical Report TR-89-041, Berkeley, 1989.

[16] N. Otsu. Nonlinear discriminant analysis as a natural extension of the linear case. *Behaviormetrika*, (2):24–59, 1975.

[17] T. L. D. Regulinski. On reliability of expert systems. *IEEE Transactions on Reliability*, 40(4):401, 1991. Editorial.

[18] D. L. Reilly, C. Scofield, C. Elbaum, and L. N. Cooper. Learning system archictectures composed of multiple learning modules. In *IEEE First International Conference on Neural Networks*, pages 495–503, 1987.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Nature*, 323(533), 1986.

[20] P. Smets, E. Mamdani, D. Dubois, and H. Prade, editors. *Non-standard Logics for Automated Reasoning*, chapter Belief Functions, pages 253–286. Academic Press, 1988.

[21] F. J. Śmieja. Multiple network systems (MINOS) modules: Task division and module discrimination. In *Proceedings of the 8th AISB conference on Artificial Intelligence, Leeds, 16–19 April, 1991*, 1991. Also available as GMD techical report 638.

[22] F. J. Śmieja. MINOS reflective agents and the Pandemonium system. in preparation, 1993.

[23] F. J. Śmieja. Neural network constructive algorithms: Trading generalization for learning efficiency? *Circuits, Systems and Signal Processing*, 12(2):331–374, 1993.

[24] F. J. Śmieja. Rejection of incorrect answers from a neural net classifier. In J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation (Proceedings of the IWANN 93)*. Springer Verlag, 1993. Lecture Notes in Computer Science 686.

[25] F. J. Śmieja and H. Mühlenbein. Reflective modular neural network systems. Technical Report 633, GMD, Sankt Augustin, Germany, February 1992.

[26] G. Wahba, C. Gu, Y. Wang, and R. Chappell. Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance. Technical Report 899, University of Wisconsin, January 1993.

[27] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1, 1989.

[28] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, LANL, Los Alamos, NM, 1990.

[29] Byoung-Tak Zhang and Gerd Veenker. Neural networks that teach themselves through genetic discovery of novel examples. In *International Joint Conference on Neural Networks (IJCNN-91), Vol 1*, pages 690–695. IEEE and INNS, 1991.