A Global Optimization Technique for Statistical Classifier Design*

David Miller, Ajit Rao, Kenneth Rose, and Allen Gersho
Center for Information Processing Research
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106
e-mail: gersho@ece.ucsb.edu
Tel: (805) 893-2037
Fax: (805) 893-3262

Abstract

A global optimization method is introduced for the design of statistical classifiers that minimize the rate of misclassification. We first derive the theoretical basis for the method, based on which we develop a novel design algorithm and demonstrate its effectiveness and superior performance in the design of practical classifiers for some of the most popular structures currently in use. The method, grounded in ideas from statistical physics and information theory, extends the deterministic annealing approach for optimization, both to incorporate structural constraints on data assignments to classes and to minimize the probability of error as the cost objective. During the design, data are assigned to classes *in probability*, so as to minimize the expected classification error given a specified level of randomness, as measured by Shannon's entropy. The constrained optimization is equivalent to a free energy minimization, motivating a deterministic annealing approach in which the entropy and expected misclassification cost are reduced with the temperature while enforcing the classifier's structure. In the limit, a hard classifier is obtained. This approach is applicable to a variety of classifier structures, including the widely used prototype-based, radial basis function, and multilayer perceptron classifiers. The method is compared with learning vector quantization, back propagation, several radial basis function design techniques, as well as with paradigms for more directly optimizing all these structures to minimize probability of error. The annealing method achieves significant performance gains over other design methods on a number of benchmark examples from the literature, while often retaining design complexity comparable to or only moderately greater than that of strict descent methods. Substantial gains, both inside and outside the training set, are achieved for complicated examples involving high-dimensional data and large class overlap.

*This work was supported in part by the National Science Foundation under grant no. NCR-9314335, the University of California MICRO program, Rockwell International Corporation, Hughes Aircraft Company, Echo Speech Corporation, Signal Technology Inc., Lockheed Missile and Space Company, and Qualcomm, Inc.

1 Introduction

The problem of designing a statistical classifier to minimize the probability of misclassification or a more general risk measure has been a topic of continuing interest since the 1950s. Much of the early, classical work focused on linear classifiers [40],[14],[46] and parametric classifiers, e.g. [9]. More recently, with the increase in power of serial and parallel computing resources, a number of more complex classifier structures have been proposed, along with associated learning algorithms to design them. The most prominent research has focused on several structures: decision trees [3], [32] and extensions thereof [5],[12]; nearest-prototype (NP) classifiers with the learning vector quantizer (LVQ) design [21]; radial basis function (RBF) classifiers [29]; and multilayer perceptrons (MLPs) [42]. Several review articles discuss the tradeoffs in performance, memory, implementation complexity, and design complexity for the various classification schemes as well as recent developments relating to these approaches [24],[16]. Much attention has focused on MLPs, primarily due to the increasing interest in neural network models and in their applicability for a variety of signal processing applications. MLPs and other neural network models have been investigated as alternatives to more traditional classifiers for engineering applications such as speech recognition [23], [13],[1], as well as in the contexts of statistical and scientific inquiry [35]. MLPs can form complex decision boundaries [25], with the associated classification rule efficiently implementable via parallel processing. While neural networks offer powerful structures for classification, their potential cannot be fully realized without effective learning procedures well-matched to the minimum classification-error objective.

1.1 Limitations of Conventional Methods

The standard back propagation learning algorithm for MLPs [42] uses as its design objective the minimization of the distance between the continuous network output and a target output associated with the discrete class label (which is binary for the two-class case). Essentially, this approach views the learning problem for classification as the design of a *regression* model to fit to the class targets. Recently several researchers have recognized that this objective is not equivalent to minimizing the probability of misclassification. Rather, back propagation learning for MLPs, as well as corresponding techniques for other classifiers, effectively train the networks to approximate the Bayes-optimal discriminant function or, equivalently, to estimate the a posteriori probabilities that data belong to a given class [34],[41],[52],[9]. While large networks can in principle provide a close fit to the Bayes discriminant function, in practice the network size must be constrained so as to avoid high complexity and the problem of overfitting the network to the (finite-length) training set. Thus, networks trained by back propagation or related learning algorithms might achieve substantially poorer classification performance than networks trained by alternative methods.

A number of researchers have proposed modified cost objectives and/or learning algorithms which better match the goal of minimizing classification error (or minimizing risk, if errors are not weighed equally) [8],[21], [13],[47],[19], [49],[31]. Many of these methods implement descent techniques such as gradient descent or related approaches, either of a sequential or batch nature, on a cost surface which is a "smoothed" approximation to the misclassification error cost surface. "Smoothing" the classification cost is necessary because it is a piecewise constant surface whose zero gradients are useless for a descent procedure [8],[19],[31]. This basic design approach has been described as discriminative learning [19]. Discriminative learning techniques have a potential advantage over regression-based approaches for classification (e.g., back propagation) in the following important sense: the regression approach weights all errors in the estimates of a posteriori probabilities *equally*, whereas implementation of the Bayes decision rule really only requires accurate estimation of the *largest* class a posteriori probabilities. Thus, in the regression approach parameters of the model are effectively required to learn "more" than what is necessary to achieve accurate classification, which may limit the performance achievable given a fixed model size. By contrast, discriminative learning essentially trains the model parameters solely to move the resulting classifier decision boundaries so as to directly reduce the error rate.

While this general paradigm optimizes MLPs and other classifiers to effectively minimize classification error, a serious concern is the potential to fall into poor local minimum traps, which often riddle the energy surface. Several researchers have noted problems of local optima in neural networks [47],[49],[53],[35], as well as related difficulties in optimizing the full complement of model parameters (see [48] for a study on RBFs). Moreover some studies have also reported on the complex nature of the energy surface, which can cause slow convergence of descent techniques [15]. Nevertheless, much of the work in the literature appears to ignore these problems, and there have been few practical approaches addressing them. Typically, efforts to avoid poor local optima of the cost have been limited to the practice of generating numerous solutions based on random initialization and then choosing the best result [35]. Since it is not well understood how the number and quality of local optima may depend on the data distributions, the network model, and its size, there are no guarantees that this "sampling" approach to classifier design will yield good solutions. Moreover, while stochastic optimization approaches such as simulated annealing (SA) [20] can be applied at least in principle, the computational complexity of such methods is often prohibitive.

1.2 Structurally-constrained Deterministic Annealing for Classifier Design

As an alternative to strict descent-based procedures, we propose a new deterministic learning algorithm for statistical classifier design with a demonstrated potential for avoiding local optima of the cost. Several deterministic, annealing-based techniques have been proposed for avoiding nonglobal optima in computer vision and image processing [54],[10],[2], in combinatorial optimization [51],[45], and elsewhere. Our approach is derived based on ideas from information theory and statistical physics, and builds on the probabilistic framework of the deterministic annealing (DA) approach to clustering and related problems [38][37][39]. In the DA approach for data clustering, the maximum entropy principle [17] was invoked to obtain the distributions at a given level of expected cost. In recent work [26],[27], we have extended this formulation to attack a larger class of optimization problems than was originally conceived. The new formulation emphasizes the quantity in statistical physics known as the Helmholtz free energy [4] as the effective cost function. This formulation explicitly characterizes the deterministic annealing process as a gradual reduction in both the entropy and cost of the system with decreasing "temperature", where the temperature parameter is a Lagrange multiplier controlling the system's cost and entropy. While equivalent to other formulations of deterministic annealing, the approach based on the Helmholtz free energy is found to be especially useful in extending DA-based optimization techniques to incorporate *structural constraints* on data assignments.

It is noted that prior work on deterministic annealing, e.g. [38],[54],[44],[10], [2],[51] addressed data association problems of a combinatorial nature such as data clustering, graph partitioning, and matching problems for which the cost can be expressed explicitly through binary (0-1) assignment variables. In these problems, the "data" of the problem is freely assigned to "classes" or "groups" via the binary assignment variables. The novel contribution of this work relative to prior work on deterministic annealing is its extension of the annealing framework to incorporate structural constraints on the assignments of data to classes. By *structurally-constrained* data assignments, it is meant that the data assignments are constrained to agree with a parameterized classification rule, such as that of a nearest prototype classifier, a decision tree, or a neural network model. Examples of structurally-constrained data association problems include fundamental statistical learning problems such as statistical classification, piecewise regression, and structured (e.g. tree-structured) vector quantization.

Appropriately, the extension of DA methods to incorporate structure is best seen within the learning context. We note that recent work has related techniques from statistical physics to the learning problem via a connection with likelihood estimation [36],[55],[28]. In [55] it was noted that the binary assignment variables in the combinatorial optimization problem can be related to the unknown or "hidden" data in methods for maximum likelihood estimation such as the Expectation/Maximization (EM) algorithm [7]. From this point of view, the cost associated with the combinatorial optimization problem is interpreted as the complete data likelihood of a corresponding estimation problem. Similar observations were made for problems specifically involving images, e.g. [56]. This connection between statistical physics and statistical estimation suggested that deterministic annealing could be applied to the learning problem when the learning objective is maximum likelihood estimation (MLE) [55], [28]. MLE techniques have long been applied to unsupervised learning [9], and supervised learning problems such as regression have also recently been posed as likelihood estimation [18]. However, while likelihood estimation is an important learning objective, maximizing the model's likelihood is not equivalent to optimizing parameters of a classifier so as to directly minimize probability of error if the goal is classification, nor to choosing model parameters to directly minimize mean-squared error if the goal is regression. These are often the desired supervised learning objectives. In the classification context, we have already pointed out the potential advantage of directly minimizing probability of error, which has been exploited by discriminative learning techniques. Previous physics-based approaches do not anneal over models to directly minimize desired supervised learning objectives such as probability of error, but as will be seen in the sequel, the formulation suggested here based on the Helmholtz free energy incorporates both the model's structure and its cost directly within the optimization framework, and anneals over the structure to directly minimize the supervised learning objective. In this work we develop a structurally-constrained extension of DA, specialized for the problem of designing statistical classifiers to minimize probability of error. In addition to its relation to prior work on deterministic annealing, we will also point out the connection between our method and discriminative learning techniques.

Whereas most design methods have been developed for specific classifier structures, e.g. [21],[29],[42] (an exception is the approach in [19]), the method we develop can be applied generally to optimize a variety of structures. In this work, we will develop algorithms and demonstrate results for three of the most widely used structures: nearest-prototype classifiers, radial basis function classifiers, and multilayer perceptrons. Our method will be demonstrated to provide substantial performance gains over conventional design techniques for all of these structures, while retaining design complexity in many cases comparable to the strict descent methods. Our approach often designs small networks to achieve training set performance that can only be obtained by a much larger network designed in a conventional way. The design of smaller networks may translate to more robust solutions and superior performance outside the training set, as we will note in particular for MLP and RBF networks. We thus provide a general approach for designing statistical classifiers based on training data which avoids many local minima that trap other known methods, and which achieves classifier designs superior to those obtained by other methods.

1.3 Outline of This Paper

In the next section, we first state the problem, introduce mathematical notation, and briefly review the commonly used NP, RBF, and MLP classifier structures. We then develop our probabilistic framework for optimization, which leads to a general classifier design method. The formulation is next specialized for the different structures, yielding separate DA learning algorithms for each structure. In section 3, we provide performance comparisons between classifiers designed by the DA approaches and by conventional techniques, drawing upon several sources of benchmark data from the literature. We then conclude with some discussion and suggestions for future work.

2 Classifier Design Formulation

2.1 Problem Statement

Let $\mathcal{T} = \{(\mathbf{x}, c)\}$ be a training set of N labelled vectors, where $\mathbf{x} \in \mathcal{R}^n$ is a feature vector and $c \in \mathcal{I}$ is its class label from an index set $\mathcal{I}$. A classifier is a mapping $C : \mathcal{R}^n \rightarrow \mathcal{I}$, which assigns a class label in $\mathcal{I}$ to each vector in $\mathcal{R}^n$. Typically, the classifier is represented by a set of model parameters $\Lambda = \{\Lambda_k\}$. The classifier specifies a partitioning of the feature space into regions $R_j \equiv \{\mathbf{x} \in \mathcal{R}^n : C(\mathbf{x}) = j\}$, where $\bigcup_j R_j \equiv \mathcal{R}^n$ and $\bigcap_j R_j \equiv \emptyset$. It also induces a corresponding partitioning of the training set into subsets $\mathcal{T}_j \equiv \{(\mathbf{x}, c) \in \mathcal{T} : C(\mathbf{x}) = j\}$. A training pair $(\mathbf{x}, c) \in \mathcal{T}$ is *misclassified* if $C(\mathbf{x}) \neq c$. The performance measure of primary interest is the empirical error fraction P_e of the classifier, i.e. the fraction of the training set (for generalization purposes, the fraction of the test set) which is misclassified:

$$P_e = \frac{1}{N} \sum_{(\mathbf{x},c) \in \mathcal{T}} \rho(c, C(\mathbf{x})) = \frac{1}{N} \sum_{j \in \mathcal{I}} \sum_{(\mathbf{x},c) \in \mathcal{T}_j} \rho(c, j), \qquad (1)$$

where $\rho(c,j) = 1$ if $c \neq j$ and 0 otherwise. Our goal is to optimize statistical classifiers for this performance measure. While the approach we will develop addresses a variety of structures, for concreteness we will focus on three of the most commonly used classifiers: the NP, RBF, and MLP classifiers. We now briefly review these network models and their associated classification rules.

Nearest-Prototype (NP) Classification

The NP classifier structure is shown in Figure 1. The classifier is specified by the parameter set $\Lambda = \{\mathbf{x}_{jk}\}$ where $\mathbf{x}_{jk} \in \mathcal{R}^n$ is the kth prototype associated with class $j \in \mathcal{I}$. The NP classifier maps a vector in $\mathcal{R}^n$ to the class associated with the nearest prototype, specifying a partition of $\mathcal{R}^n$ into the regions:

$$R_j \equiv \bigcup_k S_{jk} \quad \text{with} \quad S_{jk} \equiv \{\mathbf{x} \in \mathcal{R}^n : d(\mathbf{x}, \mathbf{x}_{jk}) \leq d(\mathbf{x}, \mathbf{x}_{lm}) \;\; \forall l, m\}, \tag{2}$$

i.e., each region R_j is the union of Voronoi cells S_{jk}. Here, $d(\cdot,\cdot)$ is the "distance measure" used for classification. For consistency with the neural network models to follow, which classify based on the *largest* output of the network ("winner takes all"), we note trivially that the classification rule can also be written as

$$R_j \equiv \bigcup_k S_{jk} \quad \text{with} \quad S_{jk} \equiv \{\mathbf{x} \in \mathcal{R}^n : F_{jk}(\mathbf{x}) \geq F_{lm}(\mathbf{x}) \;\; \forall l, m\}, \tag{3}$$

by choosing $F_{jk}(\mathbf{x}) \equiv -d(\mathbf{x}, \mathbf{x}_{jk})$.

One weakness of prototype-based classification is the need to specify the allocation of model parameters (the prototypes) to the respective classes. Typically this allocation is heuristic, leading to a suboptimal distribution of parameters, with some classes deficient and others maintaining a surplus of prototypes. The RBF and MLP models we next examine eliminate this problem, as all parameters (excepting weights to the output layer) contribute to each class output.

Radial Basis Function (RBF) Classification

The RBF classifier structure is shown in Figure 2. The classifier is specified by a set of Gaussian receptive field functions, $\{e^{-\frac{|\mathbf{x}-\mu_k|^2}{{\sigma_k}^2}}\}$, and by a set of scalar weights $\{\lambda_{kj}\}$ which connect each of the receptive fields to the class outputs of the network. Thus, $\Lambda = \{\{\mu_k\}, \{{\sigma_k}^2\}, \{\lambda_{kj}\}\}$. The parameter μ_k is the "center" vector for the receptive field and ${\sigma_k}^2$ is its "width". In the "normalized" representation for RBFs [29] which we will adopt here, the network output for each class is written in the form

$$F_j(\mathbf{x}) = \sum_k \lambda_{kj} \phi_k(\mathbf{x}), \tag{4}$$

where

$$\phi_k(\mathbf{x}) = \frac{e^{-\frac{|\mathbf{x}-\mu_k|^2}{{\sigma_k}^2}}}{\sum_l e^{-\frac{|\mathbf{x}-\mu_l|^2}{{\sigma_l}^2}}}. \tag{5}$$

Since $\phi_k(\cdot)$ can be viewed as a probability mass function, each network output is effectively an average of weights emanating from each of the receptive fields. The classifier maps the vector $\mathbf{x}$ to the class with the largest output:

$$R_j \equiv \{\mathbf{x} \in \mathcal{R}^n : F_j(\mathbf{x}) \geq F_k(\mathbf{x}) \ \ \forall k \in \mathcal{I}\}. \tag{6}$$

Multilayer Perceptron (MLP) Classification

The MLP classifier structure is shown in Figure 3. We restrict ourselves to the MLP structure with a binary output unit per class [1]. The classification rule for MLPs is the same as that for RBFs (6), but the output functions $\{F_j(\cdot)\}$ are parametrized differently.

The input $\mathbf{x}$ passes through K layers with M_k neurons in layer k. We define u_{kj} to be the output of hidden unit j in layer k, with the convention that layer 0 is the input layer $u_{0j} = x_j$, and layer K is the output layer $u_{Kj} = F_j(\mathbf{x})$. To avoid special notation for thresholds, we define the augmented output vector of layer k as $\tilde{\mathbf{u}}_k = [u_{k1} \ \ u_{k2} \ \ \ldots \ \ u_{kM_k} \ \ 1]^T$. This is a standard notation allowing us to replace thresholds by synaptic weights which multiply a fixed input value of unity. The weight matrix $\mathbf{W}_k$ connects the augmented outputs of layer $k-1$ and the neurons of layer k. The activation function of the kth layer is the vector valued function $\mathbf{f}_k : \mathcal{R}^{M_k} \rightarrow \mathcal{R}^{M_k+1}$defined as $\mathbf{f}_k(\mathbf{v}) = [f_k(v_1) \ \ f_k(v_2) \ \ \ldots \ \ f_k(v_{M_k}) \ \ 1]^T$ where $f_k(\cdot)$ is the scalar activation function used by all neurons in the kth layer. In our experiments, we used the logistic activation function ($f_k(v) = \frac{1}{1+e^{-v}}$) for the hidden layers $k = 1, \ldots, K-1$, and the linear activation function ($f_K(v) = v$) for the output layer. The *activity level* at the input of the kth layer is given by

$$\mathbf{v}_k = \mathbf{W}_k \tilde{\mathbf{u}}_{k-1}. \tag{7}$$

Thus, the network's operation can be described by the following recursion formula:

$$\tilde{\mathbf{u}}_k = \mathbf{f}_k(\mathbf{v}_k) = \mathbf{f}_k(\mathbf{W}_k \tilde{\mathbf{u}}_{k-1}) \qquad k = 1, 2, \ldots, K. \tag{8}$$

2.2 Approach to Classifier Design

Although we have described three distinct classifier structures, the operation of each is consistent with that of a general network model: given a feature $\mathbf{x}$ the network produces competing class outputs $\{F_j(\mathbf{x})\}$, and then classification decisions are made based on the largest, "winning" output. Note that this model is identical with the standard (or canonical) representation of classifiers via maximization over a set of *discriminant functions* [9], where the discriminant functions are given here by $F_j(\mathbf{x})$. Any (hard) classifier can be represented by this model, albeit possibly with complicated discriminant functions. We now use this convenient representation to develop a general optimization approach for statistical classifier design.

[1] Note that other architectures for classification are also possible, e.g. MLP or RBF architectures with a binary output code, wherein the number of output units is logarithmically related to the number of classes. For this classifier model, individual binary classification decisions are made for each output unit, and the resulting bits specify the class.

2.2.1 Maximum Entropy and Equivalent Principles

Our probabilistic design approach can be motivated by several different points of view, which are practically equivalent (since they lead to the same optimization method), but which are philosophically distinct. One perspective is that of the maximum entropy principle. According to [17], the least biased distribution is that which maximizes entropy subject to constraints which incorporate what is known about a system. Often, the constraints are expectation constraints, specifying quantities such as "energy", using the physical analogy. In the context of the optimization problem, constraints on entropy maximization can be used to introduce the expected cost. The maximum entropy principle was used to develop a DA method for clustering [37] and related optimization problems [38],[39]. However, this principle is not universally accepted [2]. An equivalent, alternative approach which may at least have more intuitive appeal for optimization is to find the distribution which minimizes the expected cost given a constrained level of randomness in the solution. A natural, information-theoretic measure of randomness or uncertainty is the Shannon entropy. Thus, we can state the problem of finding the *best* distribution in the sense of minimum expected cost given a constrained level of entropy. The equivalence between these two methods – i) maximizing entropy given a constraint on expected cost, and ii) minimizing expected cost given constrained entropy – is easily seen through the (unconstrained) Lagrangian cost objective which both approaches share. In the derivation that follows, the minimum expected cost formulation is chosen. Both methods can be connected with statistical physics by noting that the Lagrangian cost objective can be interpreted as the Helmholtz free energy [4] of a simulated system. ¿From this perspective, the expected cost is the "energy" of the system and the Lagrange multiplier is the "temperature". This connection with statistical physics will be made more concrete in the sequel.

2.2.2 Randomized Classifier Partition

Whereas statistical classifiers that minimize probability of error almost invariably implement a deterministic function, producing "hard" classification decisions[3], it may still be useful in the context of the classifier *design* to allow points to be assigned to classes *in probability.* As in the original DA approach for data clustering [37], we cast the optimization problem in such a probabilistic framework, considering a "random" classifier characterized by a probabilistic assignment of data to classes. Accordingly, we define the *probabilities of association* between a feature $\mathbf{x}$ and the class regions, *i.e.* $\{P[\mathbf{x} \in R_j]\}$. As our design method, which optimizes over these probabilities, must ultimately form a classifier that makes "hard" decisions based on a specified network model, the distributions must be chosen to be consistent with the decision rule of the model. In other words, we need to introduce randomness into the classifier's partition. Clearly, there are many ways one could define probability distributions which

[2] The maximum entropy principle was given an axiomatic basis in [43] and was shown to be a consequence of conditional probability theory in [50] and [6].

[3] We note that fuzzy or multi-valued decisions may sometimes be useful when there is uncertainty in the class labels or when the data is not well-represented by "crisp" categories or classes [22]. However, otherwise, classifier systems designed to minimize probability of error may introduce fuzziness at intermediate stages of the data processing, but ultimately make hard decisions.

are consistent with the hard partition at some limit. We use an information-theoretic approach. We measure the randomness or uncertainty by the *Shannon entropy*, and determine the distribution for a given level of entropy. At the limit of zero entropy we should recover a hard partition. For now, suppose that the values of the model parameters $\Lambda = \{\Lambda_k\}$ have been fixed. We can then write an objective function whose maximization determines the hard partition for a given Λ:

$$F_h = \frac{1}{N} \sum_{j \in \mathcal{I}} \sum_{(\mathbf{x},c) \in \mathcal{T}_j} F_j(\mathbf{x}). \tag{9}$$

Note that the winner-take-all rule (6) is optimal in the sense of F_h. Specifically, maximizing (9) over all possible partitions captures the decision rule of (6). The probabilistic generalization of (9) is

$$F = \frac{1}{N} \sum_{(\mathbf{x},c) \in \mathcal{T}} \sum_j P[\mathbf{x} \in R_j] F_j(\mathbf{x}), \tag{10}$$

where the partition is now represented by association probabilities, and the corresponding entropy is

$$H = -\frac{1}{N} \sum_{(\mathbf{x},c) \in \mathcal{T}} \sum_j P[\mathbf{x} \in R_j] \log P[\mathbf{x} \in R_j]. \tag{11}$$

It is emphasized that H measures the average level of uncertainty *in the classification decisions.* We determine our assignment distribution at a given level of randomness as the one which maximizes F while maintaining H at the prescribed level $\hat{H}$:

$$\max_{\{P[\mathbf{x} \in R_j]\}} F \quad \text{subject to} \quad H = \hat{H}. \tag{12}$$

The result is the *best* probabilistic partition, in the sense of the structural objective F, *at the specified level of randomness.* For $\hat{H} = 0$ we get back the hard partition, maximizing (9) and thus satisfying the winner-take-all classifier structure. At any $\hat{H}$, the solution of (12) is the Gibbs distribution

$$P[\mathbf{x} \in R_j] = \frac{e^{\gamma F_j(\mathbf{x})}}{\sum_k e^{\gamma F_k(\mathbf{x})}} \equiv P_{j|x}(\Lambda), \tag{13}$$

where γ is the Lagrange multiplier controlling the level of entropy. For $\gamma \to 0$, the associations become increasingly uniform, while for $\gamma \to \infty$, they revert to hard classifications, equivalent to application of the rule in (6). Thus, (13) is a probabilistic generalization of the winner-take-all classifier which satisfies its structural constraint, specified by (6), for the choice $\gamma \to \infty$. Note that the probabilities depend on Λ through the network outputs. Here we have emphasized this dependence through our choice of concise notation. Equation (13) applies as is for MLPs and RBFs. However, for prototype-based classification we randomize over the sub-regions S_{jk}, leading to

$$P[\mathbf{x} \in S_{jk}] \equiv P_{jk|x}(\Lambda) = \frac{e^{\gamma F_{jk}(\mathbf{x})}}{\sum_l \sum_m e^{\gamma F_{lm}(\mathbf{x})}}, \tag{14}$$

and since $R_j = \bigcup_k S_{jk}$, we have

$$P_{j|x}(\Lambda) = \sum_k P_{jk|x}(\Lambda). \tag{15}$$

This distribution reduces to the decision rule of (3) for $\gamma \to \infty$. In the next section, we will describe an optimization technique which gradually introduces the objective of maximizing F, ultimately yielding a hard classifier which maximizes F_h, and thus achieves the winner-take-all structure.

2.2.3 Design by Deterministic Annealing

Until now we formulated a controlled way of introducing randomness into the classifier's partition while enforcing its structural constraint. However, the derivation assumed that the model parameters were given, and thus produced only the *form* of the distribution $P_{j|x}(\Lambda)$, without actually prescribing how to choose the values of its parameter set. Moreover the derivation did not consider the ultimate goal of minimizing the probability of error. Here we remedy both shortcomings.

The method we suggest gradually enforces formation of a hard classifier minimizing the probability of error. We start with a highly random classifier and a high expected misclassification cost. We then gradually reduce both the randomness and the cost in a deterministic learning process which enforces formation of a hard classifier with the requisite structure. While the motivation behind this basic approach remains as yet unclear, in the sequel we will relate this process to annealing methods from statistical physics and show the method to be useful for avoiding poor local optima of the cost. As before, we need to introduce randomness into the partition while enforcing the classifier's structure, only now we are also interested in minimizing the expected misclassification cost. While satisfying these multiple objectives may appear to be a formidable task, the problem is greatly simplified by restricting the choice of random classifiers to the set of distributions $\{P_{j|x}(\Lambda)\}$ as given in (13) or (15) – these random classifiers naturally enforce the structural constraint of (3) or (6) through γ, as we explained earlier. Thus, from the parametrized set $\{P_{j|x}(\Lambda)\}$, we seek that distribution which minimizes the average misclassification cost while constraining the entropy:

$$\min_{\gamma,\Lambda} < P_e > \equiv \min_{\gamma,\Lambda} \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} P_{j|x}(\Lambda)\delta(c,j), \tag{16}$$

subject to

$$H = \hat{H}.$$

The distribution is chosen by optimization over its parameter set. The solution yields the best random classifier in the sense of minimum $< P_e >$ for given $\hat{H}$. At the limit of zero entropy, we should get the best *hard* classifier in the sense of P_e with the desired structure, specified by (3) or (6).

The constrained minimization (16) is equivalent to the unconstrained minimization of the Lagrangian:

$$\min_{\Lambda,\gamma} L \equiv \min_{\Lambda,\gamma} \beta < P_e > -H, \tag{17}$$

where β is a Lagrange multiplier controlling the tradeoff between H and $< P_e >$. For $\beta = 0$, the sole objective is entropy maximization, which is achieved by the uniform distribution. This solution, which is the global minimum for L at $\beta = 0$, can be obtained by choosing $\gamma = 0$. At the other extreme, for

$\beta \to \infty$, the sole objective is to minimize $< P_e >$, and is achieved by choosing a non-random (hard) classifier (hence minimizing P_e). The hard solution satisfies the classification rule ((3) or (6)) and is obtained for $\gamma \to \infty$.

Motivation for minimizing the Lagrangian can be obtained from a physical perspective by noting that L is the Helmholtz free energy of a simulated system, with $< P_e >$ the "energy" of the system, H its entropy, and $\frac{1}{\beta}$ the "temperature". Thus, from this physical view we can suggest a *deterministic annealing* (DA) process which involves minimizing L starting at the global minimum for $\beta = 0$ (high temperature) and tracking the solution while increasing β towards infinity (zero temperature). In this way, we obtain a sequence of solutions of decreasing entropy and expected misclassification cost. Each such solution is the best random classifier in the sense of $< P_e >$ for a given level of randomness. The annealing process is useful for avoiding local optima of the cost $< P_e >$, and minimizes $< P_e >$ directly at low temperature. The approach we suggest is an extension of previous DA methods which considers the probability of error as the cost objective. However, the approach that we suggest also extends DA-based methods in a more fundamental way as well, as it allows annealing to occur *in the structural objective*, as well as in the system's expected cost and entropy. To clarify this statement, we observe that while the annealing process suggested here ostensibly involves the quantities H and $< P_e >$, the restriction to $\{P_{j|x}(\Lambda)\}$ from (13) ensures that the process also enforces the structural constraint on the classifier in a controlled way, through the objective F. *Note in particular that γ has not lost its interpretation as a Lagrange multiplier determining F.* Thus, $\gamma = 0$ means that F is unconstrained – we are free to choose the uniform distribution. Similarly, sending $\gamma \to \infty$ requires maximizing F (see (12) and (13)) – hence the hard solution, which maximizes F_h and satisfies (6). Since γ is chosen to minimize L, this parameter effectively determines the level of F – *the level of structural constraint – consistent with* H and $< P_e >$ for a given β. As β is increased, the entropy constraint is relaxed, allowing greater satisfaction of *both* the minimum $< P_e >$ and maximum F objectives. At the limit of $\beta \to \infty$, γ is also driven to infinity, so that a hard classifier results, maximizing F_h and hence achieving the winner-take-all structure, as well as directly minimizing the probability of error objective. Thus, annealing in β gradually enforces both the structural objective F (via γ) and the minimum $< P_e >$ objective [4].

The physical interpretation of the Lagrangian makes the connection with stochastic annealing methods clear: rather than generating a stochastic solution process, which spends a long time at each temperature in order to reach thermal equilibrium, the DA approach performs a direct, deterministic optimization of the quantity which is stochastically optimized by SA at equilibrium – the free energy[5]. In addition to the connection with SA, our formulation also clearly identifies the relationship between the DA and discriminative learning (DL) methods. Note that the DA method performs a randomization at a given temperature which achieves cost smoothing in much the same way as discriminative

[4] Note, too, that while our method varies β and optimizes γ, an alternative annealing approach which will not be discussed here could possibly involve varying γ, i.e. introducing an increasing constraint on the classifier's structure, and optimizing a parameter that determines the tradeoff between $< P_e >$ and H. The feasibility of such an approach will not be investigated in this paper.

[5] However, whereas SA has been shown to converge in distribution to a uniform distribution over the set of globally optimal solutions, no similar proof exists for DA.

learning techniques. Thus, DL can be interpreted from the perspective of our DA framework as essentially minimizing the smoothed cost $< P_e >$ directly over the classifier parameters, without controlling the entropy of the data assignments along the way. DL must choose the initial classifier parameters and the initial amount of cost smoothness. These choices may have significant bearing on the achieved solution quality. By contrast, the DA approach controls the level of cost smoothness – now characterized by the entropy of the random classifier – as well as F and $< P_e >$. These quantities are gradually varied through incremental temperature reduction, starting from a high entropy state, with the ultimate objective a global minimum energy configuration (error count) and a desired classifier structure at low temperature. As we will experimentally verify through simulations, our annealing approach outperforms design based on directly minimizing $< P_e >$.

2.3 Necessary Optimality Conditions

Minimization of the Lagrangian at a given β can be realized by gradient descent or any other local function minimization technique. Here, we derive necessary optimality conditions, presenting them in a convenient form to aid interpretation. For concreteness, in this section we will assume a model consistent with the RBF and MLP classifiers. We first rewrite the Lagrangian of (17) explicitly as:

$$L = \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} P_{j|x}(\Lambda)\{\beta\delta(c,j) + \log P_{j|x}(\Lambda)\} \tag{18}$$

$$\equiv \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \left\{ \sum_{j} P_{j|x}(\Lambda) L_{xj} \right\} \equiv \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} L_x. \tag{19}$$

The braces in (18) identify L_{xj}, the contribution to the cost when the feature $\mathbf{x}$ is assigned to class j, and the braces in (19) identify $L_x \equiv < L_{xj} >_j$, the average contribution for this feature[6]. Noting that the contributions to the derivative of L are additive over the data and applying the chain rule, we may write:

$$\frac{\partial L_x}{\partial \Lambda_k} = \sum_j \frac{\partial L_x}{\partial P_{j|x}(\Lambda)} \frac{\partial P_{j|x}(\Lambda)}{\partial \Lambda_k} \tag{20}$$

and

$$\frac{\partial L}{\partial \Lambda_k} = \frac{1}{N} \sum_x \frac{\partial L_x}{\partial \Lambda_k}. \tag{21}$$

Setting $\frac{\partial L}{\partial \Lambda_k} = 0$ and simplifying, we obtain the necessary optimality condition:

$$\sum_x \sum_j L_{xj} P_{j|x} \left(\frac{\partial F_j(\mathbf{x})}{\partial \Lambda_k} - < \frac{\partial F_l(\mathbf{x})}{\partial \Lambda_k} >_l \right) = 0, \qquad \forall k. \tag{22}$$

(Note that in (22) we have dropped explicit representation of the distribution's dependence on Λ for convenience.) In a similar fashion, the condition $\frac{\partial L}{\partial \gamma} = 0$ yields

$$\sum_x \sum_j L_{xj} P_{j|x} \left(F_j(\mathbf{x}) - < F_l(\mathbf{x}) >_l \right) = 0. \tag{23}$$

[6] Here we have used $< A_{xj} >_j$ to denote the average of a random quantity A_{xj} over the distribution $\{P_{j|x}\}$, *i.e.* $< A_{xj} >_j = \sum_j P_{j|x} A_{xj}$.

We have obtained the mathematical conditions for optimality. In what follows, we give some intuitive interpretation. In particular, conditions (22) and (23) can be viewed, appropriately, within the context of supervised learning. Toward this end, note that L can be interpreted as a generalized "risk" function, with $L_{xj}P_{j|x}$ the contribution associated with classifying $\mathbf{x}$ to class j. ¿From this perspective, equations (22) and (23) specify the optimal parameters as those which provide the best risk tradeoff over the entire training set. The influence which an individual classification decision has on the learning is proportional to its associated risk $L_{xj}P_{j|x}$. Moreover, for a given parameter this influence also depends on the "tuning" of the parameter for the class output – i.e., the degree to which the parameter can affect the class output (and hence the ownership of the point). In (22) this "tuning" is measured by the sensitivity $\frac{\partial F_j(\mathbf{x})}{\partial \Lambda_k}$, relative to its average value over all classes. In (23), since γ linearly weights all class outputs, this sensitivity is simply the class output $F_j(\mathbf{x})$, relative to the average over all classes. Risk minimization normally involves making "hard" decisions. However, the entropy constraint, *which is explicit in the "risk" through the terms log* $P_{j|x}$, guarantees that $P_{j|x}$ remains "soft" at intermediate β so as to achieve a specified level of H while minimizing $< P_e >$.

We can easily relate these interpretations of our design procedure to a method which performs direct descent on $< P_e >$. Note that the necessary conditions for minimizing $< P_e >$ have the same form as (22) and (23) if we replace L_{xj} by $\delta(c,j)$. In this case, since there is no entropy term, minimization of the cost should be achieved by making the probabilities "hard". By contrast, in our approach a "hard" solution is only obtained for $\beta \to \infty$, *i.e.*, when the Lagrangian is equivalent to $< P_e >$. At this limit, $\gamma \to \infty$, $\{P_{j|x} \to \{0,1\}\}$, $< P_e > \to P_e$, $F \to F_h$, and for the "winning" output class j we have $L_{xj} = L_x$, $F_j(\mathbf{x}) = < F_l(\mathbf{x}) >_l$, and $\frac{\partial F_j(\mathbf{x})}{\partial \Lambda_k} = < \frac{\partial F_l(\mathbf{x})}{\partial \Lambda_k} >_l$. Thus, it is readily seen that for $\beta \to \infty$, the hard cost P_e is minimized and the necessary optimality conditions are satisfied by sending $\gamma \to \infty$, so that the annealing method converges at the limit, maximizing F_h and achieving the structure (6).

2.4 Algorithm Description

Our annealing-based optimization method is a continuation method which tracks the parameters $\{\Lambda, \gamma\}$ that minimize the Lagrangian L for a sequence of increasing values of β (decreasing temperatures). The method starts at high temperature ($\beta = 0$), for which the Lagrangian cost is convex. Thus, the global minimum is found at high temperature, independent of initialization. The optimal parameters $\{\Lambda^*, \gamma^*\}$ found at a given β are then used as initialization for the optimization at the next (larger) β. At each β, L can be minimized by a gradient descent technique, or any other function minimization technique. The partial derivatives in (22) and (23) can be used as the basis for a gradient descent procedure. The conditions (22) and (23) also specify a convergence condition for the optimization at given β. Alternatively, the convergence condition at given β may be based on diminishing values of the drop in L, i.e. the optimization stops at β if $\frac{\Delta L}{L} < \epsilon$, for specified ϵ. The annealing method terminates for a value of β at which the entropy H has dropped below a specified threshold level δ. At this point, the parameters Λ are used to specify a hard classification rule, as described in section 2.1. Thus, γ is not used by the final (hard) classifier.

2.5 Specialization to Different Structures

In this section, we specialize our general approach for three major classifier structures, deriving necessary optimality conditions for each structure. This will provide the basis for the corresponding algorithms used in our simulations.

2.5.1 Optimality Conditions for Nearest-Prototype Classifiers

The probabilistic associations have the form

$$P_{j|x} = \frac{\sum_k e^{-\gamma d(\mathbf{x},\mathbf{x}_{jk})}}{\sum_l \sum_m e^{-\gamma d(\mathbf{x},\mathbf{x}_{lm})}}, \tag{24}$$

which reduce to the rule of (2) for $\gamma \to \infty$. Necessary conditions for minimizing L at any β are derived in a straightforward fashion, and are found to be:

$$\frac{\partial L}{\partial \mathbf{x}_{jk}} = 0 \Rightarrow \sum_{(\mathbf{x},c)\in\mathcal{T}} (L_{xj} - L_x) P_{jk|x} \frac{\partial d(\mathbf{x},\mathbf{x}_{jk})}{\partial \mathbf{x}_{jk}} = 0, \quad \forall j,k \tag{25}$$

and

$$\frac{\partial L}{\partial \gamma} = 0 \Rightarrow \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_j L_{xj} P_{j|x} (d_{xj} - d_x) = 0. \tag{26}$$

Here d_{xj} is the average distance from $\mathbf{x}$ to a prototype from class j, i.e. $d_{xj} = \, < d(\mathbf{x},\mathbf{x}_{jk}) >_k$, and d_x is the average distance to any prototype, i.e. $d_x = \, < d(\mathbf{x},\mathbf{x}_{jk}) >_{jk}$. The condition for a prototype vector suggests moving it away from (towards) feature vectors that it "owns" probabilistically through $P_{jk|x}$ and for which L_{xj} is greater than (less than) the average, L_x. The condition for γ suggests finding the value which achieves the best tradeoff between the (potentially) conflicting goals for individual data, seeking to "harden" or "soften" ownership based on the associated risk.

2.5.2 Optimality Conditions for Radial Basis Functions

The probabilistic associations are those based on (13), using (4). Since the network outputs are linear in the weights $\{\lambda_{kj}\}$, γ is implicitly defined through the weight magnitudes. Prior to writing down necessary optimality conditions, it is helpful to first define several quantities: the difference $\Delta_{kj}(\mathbf{x}) = \lambda_{kj} - F_j(\mathbf{x})$, and this difference averaged over all classes, *i.e.* $< \Delta_{kj}(\mathbf{x}) >_j \equiv \Delta_k(\mathbf{x})$. Then, as a function of these quantities, the necessary optimality conditions are found to be:

$$\frac{\partial L}{\partial \mu_k} = 0 \Rightarrow \sum_x \frac{\partial \phi_k(\mathbf{x})}{\partial \mu_k} \left(\sum_j P_{j|x} L_{xj} (\Delta_{kj}(\mathbf{x}) - \Delta_k(\mathbf{x})) \right) = 0, \qquad \forall k \tag{27}$$

$$\frac{\partial L}{\partial {\sigma_k}^2} = 0 \Rightarrow \sum_x \frac{\partial \phi_k(\mathbf{x})}{\partial {\sigma_k}^2} \left(\sum_j P_{j|x} L_{xj} (\Delta_{kj}(\mathbf{x}) - \Delta_k(\mathbf{x})) \right) = 0, \qquad \forall k \tag{28}$$

$$\frac{\partial L}{\partial \lambda_{kj}} = 0 \Rightarrow \sum_x \phi_k(\mathbf{x}) P_{j|x}(L_{xj} - \quad L_x) = 0 \qquad \forall k, j. \tag{29}$$

As before, these conditions can be seen to specify a type of supervised learning rule which matches intuition. In (27) and (28) the influence an individual classification decision has on the learning is a function of the associated risk $L_{xj}P_{j|x}$, as well as the sensitivity of the class output to the given parameter. In this case, the sensitivity of class output $F_j(\mathbf{x})$ to changes in a parameter from receptive field k is $(\Delta_{kj}(\mathbf{x}) - \Delta_k(\mathbf{x}))\frac{\partial \phi_k(\mathbf{x})}{\partial \Lambda_k}$, where Λ_k is either the receptive field center or its width. Equation (29) is also directly interpretable – effectively, the weight from receptive field k to class j is increased (decreased) if L_{xj} is smaller (larger) than the average L_x.

2.5.3 Optimality Conditions for Multilayer Perceptrons

Using the probabilistic associations of (13) and the MLP recursion of (8), the optimality condition of (22) specializes to

$$\sum_x \sum_j P_{j|x} L_{xj} \tilde{\mathbf{u}}_{k-1} \left(\frac{\partial F_j(\mathbf{x})}{\partial \tilde{\mathbf{u}}_k} - < \frac{\partial F_l(\mathbf{x})}{\partial \tilde{\mathbf{u}}_k} >_l \right)^T \mathbf{f}_k{}'(\mathbf{v}_k) = 0 \qquad 0 \le k \le K, \tag{30}$$

where, $\mathbf{f}_k{}'(\mathbf{v})$, the derivative of layer k's outputs with respect to its activation level is a diagonal matrix with diagonal elements $\frac{\partial f_l(v_i)}{\partial v_i}$. The optimality condition for γ remains as in (23). The only elements of (30) that are not readily available are the terms $\frac{\partial F_j(\mathbf{x})}{\partial \tilde{\mathbf{u}}_k}$ which describe the network output sensitivity to hidden layer outputs. But these can be conveniently computed using a backward recursion similar in spirit to standard back-propagation:

$$\frac{\partial F_j(\mathbf{x})}{\partial \tilde{\mathbf{u}}_{k-1}} = \mathbf{f}_k'(\mathbf{v}_k)\mathbf{W}_k \frac{\partial F_j(\mathbf{x})}{\partial \tilde{\mathbf{u}}_k} \qquad 1 \le k < K \quad \forall j, \tag{31}$$

with the initialization,

$$\frac{\partial F_j(\mathbf{x})}{\partial \tilde{\mathbf{u}}_K} = \mathbf{e}_j \qquad \forall j \tag{32}$$

where $\mathbf{e}_j$ is the jth natural basis vector of $\mathcal{R}^{M_K+1}$ (the unit vector having value one at the jth component).

3 Experimental Comparisons

In this section, we demonstrate the performance of our design approach, both on synthetic examples we have generated and on benchmark data obtained from the literature. Since our main focus is on comparing design methods, rather than classifier structures, we have performed separate experiments for each of the classifier structures. In each case, two types of comparisons may be indicated. The first relates to the performance achieved inside the training set, which demonstrates the ability of our method to optimize its design criterion, and which also illustrates the problems of local optima for conventional design techniques. The second comparison involves the test set performance, which gives an indication of how well the methods generalize to new data. In addition to discussing the solution quality of the various designs, we will also consider other issues such as the design complexity. For all the DA algorithms that follow, steepest descent was used to minimize L at a sequence of exponentially increasing β, given by $\beta(n+1) = \alpha\beta(n)$, for α between 1.05 and 1.1. We have found that, especially for RBF and MLP design, much of the optimization occurs at or near a critical temperature in the solution process. Beyond this critical temperature, the annealing process can often be "quenched" to zero temperature by sending $\gamma \to \infty$ without incurring significant performance loss, and with substantial reduction in design complexity. Quenching the process often makes the design complexity of our method comparable to (and in some cases smaller than) that of descent-based methods such as back propagation or gradient descent on $< P_e >$. In Figure 4, we illustrate the basic annealing process, showing that with increasing inverse temperature β, H and $< P_e >$ are reduced, while F increases, eventually reaching the maximum value of F_h. The structure in this case was an RBF classifier, trained on simulated data.

3.1 Prototype Examples

We have performed experimental comparisons of our nearest-prototype method with the learning vector quantizer (LVQ) [21]. As an example, consider the two-class data of Figure 5. Each class consists of a Gaussian mixture with three components. We designed prototype-based classifiers with three prototypes per class, using both the LVQ and DA optimization methods. LVQ solutions were generated based on the public domain LVQ-pak software, version 2.1, running both an optimized LVQ (OLVQ) learning phase with 500,000 iterations, as well as a fine-tuning phase using LVQ1, with the learning parameter α set to 0.03. Ten LVQ solutions were generated based on different initializations produced by the LVQ-pak's *eveninit* routine. In all cases the method was unable to discriminate the class 0 "minority" component in the upper right of Figure 5(a) (which retains only 5 % of the training set mass). Apparently, the initialization did not select a prototype from the class 0 minority component, and LVQ is unable to move class 0 prototypes through the "wall" of class 1 data which separates them from this component. The best LVQ solution, which is shown in Figure 5(a), achieved $P_e = 7.7\%$. Increasing the number of prototypes, we found that LVQ was only able to discriminate the minority component when 21 prototypes per class were introduced, and in this case the method achieved $P_e = 3.4\%$. While the extremity of this suboptimality suggests that the LVQ-pak initialization could be improved[7] this example does demonstrate LVQ's susceptibility to finding poor solutions. In fact, we also performed gradient descent on $< P_e >$ and found that poor solutions were obtained in this case as well – excepting omniscient initialization in the vicinity of the optimal solution, the best performance obtained for six prototypes was $P_e = 7.0\%$. It thus appears that strict descent methods will fail on this example unless given an excellent initialization. By contrast, the DA method using only five prototypes achieved the solution shown in Figure 5(b), with $P_e = 2.7\%$. Note that the DA method is independent of the initialization, placing all prototypes together at the global data centroid (marked by X) at $\beta = 0$ so

[7] For example, if an initialization of prototypes based on Isodata clustering followed by allocation of prototypes to the majority class of the cluster were used, much fewer than 42 prototypes (but greater than six) would suffice to find good solutions.

as to maximize entropy[8]. Then, as β is increased, the prototypes separate, reducing the entropy as well as $< P_e >$. This example demonstrates the ability of the method to avoid local optima, since the DA optimization does succeed in moving a class 0 prototype from X directly *through* the class 1 data "wall" to correctly classify the minority class 0 component and achieve what appears to be the optimal piece-wise linear result for the given number of prototypes. (Here, two of the class 0 prototypes are non-distinct, so the solution effectively uses five prototypes.) The main point of this two-dimensional example is to visually illustrate the problem of local minima and the potential which DA has for avoiding them. For experiments involving high dimensional data sets (to be described shortly), visual illustration is not possible, but our results will likewise demonstrate that local minimum problems do exist and that the DA method can be applied to provide significant performance improvement.

We also tested our approach on the "synthetic" example from [35], as well as on some other complicated synthetically generated mixture examples. On the example from [35], our approach achieved $P_e = 8.9\%$ on the test set using eight prototypes and $P_e = 8.6\%$ using twelve prototypes, in comparison to LVQ's $P_e = 9.5\%$ based on twelve prototypes. For general reference, an MLP with six hidden units achieved $P_e = 9.4\%$. For complicated mixture examples, with possibly twenty or more overlapping mixture components and multiple classes, we have found our method to consistently achieve substantial performance gains over LVQ. As an example, consider the training data for a four-class problem involving twenty-four overlapping, non-isotropic mixture components in two dimensions, shown in Figures 6 and 7. We designed NP-classifiers with 16 prototypes (four per class) using both LVQ and DA. Figures 6a and 7a display the data and partitions formed by the two solutions. Figures 6b and 7b display the prototype locations along with the partitions. The best LVQ solution based on ten random initializations, shown in Figure 6, achieved $P_e = 31\%$. Note that the method has failed to distinguish a component of class 0 in the upper left of Figure 6a, as well as a component of class 1 near the lower right of the figure. By contrast the DA solution shown in Figure 7 succeeds in discriminating these components, and achieves $P_e = 23\%$.

In addition to the above experiments, we tested our design approach on the Finnish phoneme data set that accompanies the standard LVQ package. The training set consists of 1962 vectors of 20 dimensions each. Each vector represents speech attributes extracted from a short segment of continuous Finnish speech. These vectors are labeled according to the phoneme uttered by the speaker during the corresponding segment. There are 20 classes of phonemes in the training set. In both LVQ and DA approaches, we set the number of prototypes associated with a particular class to be proportional to the relative population of that class in the training set. This is referred to as the *propinit* initialization in the standard LVQ package. We compared the performance of the DA and LVQ approaches over a range of values for the total number of prototypes. It was observed that the DA method consistently outperformed LVQ over the entire range. The experimental results are shown in Table 1. This comparison is typical of what we have seen through extensive experimentation.

[8] Such an initialization is in fact "fatal" for a strict descent-based approach, as the associated learning rule will not permit a class 0 prototype to pass through the "wall" of class 1 data.

3.2 RBF Examples

We have compared our RBF design approach with the method of Moody and Darken [29] (MD-RBF), with a method described by Tarassenko and Roberts [48] (TR-RBF), and with steepest descent on $< P_e >$ (G-RBF). MD-RBF combines unsupervised learning of receptive field parameters with supervised learning of $\{\lambda_{kj}\}$ to minimize the squared distance to target class outputs. The primary advantage of this approach is its modest design complexity. However, the receptive fields are not optimized in a supervised fashion, which can cause performance degradation. TR-RBF, one of the methods described in [48], optimizes all of the RBF parameters to approximate target class outputs in a squared error sense. This design is more complex than MD-RBF and achieves better performance for a given model size (the number of receptive fields the classifier uses). However, as discussed previously, the TR-RBF design objective is not equivalent to minimizing P_e, but as in the case of back propagation, effectively aims to approximate the Bayes-optimal discriminant. While direct descent on $< P_e >$ may minimize the "right" objective, problems of local optima may be quite severe. In fact, we have found that the performance of all of these methods can be quite poor without a judicious initialization. For all of these methods, we have employed the unsupervised learning phase described in [29] (based on Isodata clustering and variance estimation) as model initialization. Then, steepest descent was performed on the respective cost surface. We have found that the complexity of our design is typically 1-5 times that of TR-RBF or G-RBF (though occasionally our design is actually faster than G-RBF). Accordingly, we have chosen the best results based on five random initializations for these techniques, and compared with the single DA design run.

As a first example, we used the same training data of Figure 5 (previously used to test the NP structure) to test RBF performance. In Figure 8, Xs are used to denote RBF centers. The best solution with $M = 3$ receptive fields achieved by TR-RBF is shown in Figure 8a, with $P_e = 7\%$. The DA solution for $M = 3$ is shown in Figure 8b, with $P_e = 2.7\%$. G-RBF with $M = 6$ obtained $P_e = 7.2\%$. While TR-RBF and G-RBF achieved good solutions by moderately increasing M (to $M = 4$ and 8 units, respectively), MD-RBF only obtained $P_e = 2.9\%$ by increasing M to 30.

To illustrate that increasing M may not help to improve performance on the *test* set, we compared our design with the results reported in [30], for two and eight dimensional mixture examples. For the 2D example, our method achieved $P_{e_{\text{train}}} = 6.0\%$ for a 400 point training set and $P_{e_{\text{test}}} = 6.1\%$ on a 20,000 point test set, using $M = 3$ units (These results are near-optimal, based on the Bayes rate.). By contrast, the method in [30] used 86 receptive fields and achieved $P_{e_{\text{test}}} = 9.26\%$. For the 8D example and $M = 5$, our method achieved $P_{e_{\text{train}}} = 8\%$ and $P_{e_{\text{test}}} = 9.4\%$ (again near-optimal), while the method in [30] achieved $P_{e_{\text{test}}} = 12.0\%$ using $M = 128$.

More comprehensive tests on higher-dimensional data have also been performed. Two examples reported here are the 21D waveform data and the 40D "noisy" waveform data used in [3] (obtained from the UC-Irvine machine learning database repository.) Rather than duplicate the experiments conducted in [3], we split the 5000 vectors into equal size training and test sets. Our results in Table 2 and Table 3 demonstrate quite substantial performance gains over all the other methods, and performance quite close to the estimated Bayes rate of 14% [3]. Note in particular that the other methods perform quite poorly for small M, and need to increase M to achieve training set performance comparable to our approach. However, performance on the test set does not necessarily improve, and may degrade for increasing M.

3.3 MLP Examples

We have compared the performance of our DA approach for designing MLPs with two other approaches – the standard back propagation (BP) algorithm of [42] and gradient descent on the $< P_e >$ cost surface (G-MLP). In our implementation of BP, we initialized the weights to uniform random numbers between ± 0.01. Next, we used 50,000 epochs of a batch gradient descent algorithm to minimize the mean-squared error between the desired and actual outputs of the MLP. As discussed previously, BP descends on a cost surface mismatched to the minimum P_e objective. Further, its performance is heavily influenced by the choice of initial weights. In G-MLP, we attempted to improve the performance of BP, by initializing the weights with the BP solution and then descending on $< P_e >$. However, in practice, we have found that the gains achieved by G-MLP over BP are only marginal, as the optimization performance sensitively depends on the choice of initialization.

We have experimented on several two dimensional examples and some examples with features of larger dimension. In all cases, the DA design approach produced significant performance improvements. First, we revisit the two-dimensional example used to test the other structures. We designed a sequence of neural networks for this example, each with a single layer of hidden neurons. Figures 9(a) and 9(b) show partitionings of the input space generated by BP and DA respectively. Figure 9(a) is the BP solution using four hidden units, which failed to separate the small cluster of 0s in the top-right corner, achieving $P_e = 7.4\%$. Figure 9(b) shows the partition generated by a DA solution with three hidden units, giving $P_e = 2.4\%$. Using the G-MLP design approach slightly improved on the performance of BP, reducing P_e to 7.2%. Although it is conceivable that BP or G-MLP would obtain the optimal solution with a fortuitous initialization, in our experiments they required 10 hidden units to approach the performance of DA. Table 4 shows the performance of the three methods for a variety of network sizes.

Another example we chose was the 19-dimensional, 7-class image segmentation data from the UC-Irvine machine learning database. The training set contains 210 vectors and the test set contains 2100 vectors, each 19-dimensional. The features represent various attributes of a 3x3 block of pixels. The classes correspond to textures (brickface, sky, foliage, cement, window, path, grass). We designed a sequence of single hidden layer neural networks for this data set. Table 5 summarizes the results we obtained for various hidden layer sizes (M). Networks designed by DA significantly outperformed the other approaches over the entire range of network sizes.

An important concern is the issue of design complexity. In our experiments we have found the DA learning complexity to be roughly 4-8 times higher than that of back propagation and roughly the same as that of G-MLP. This suggests that the potential for performance improvement would, in typical applications, greatly outweigh the somewhat higher design complexity of the DA approach.

4 Conclusions

In this paper, we introduced a new design method for statistical classifiers aimed at minimizing the cost of misclassification. The method is based on ideas from statistical physics and information theory, and extends the deterministic annealing method, both to incorporate structural constraints as well as to minimize the probability of error as the cost. The design methodology is general and applicable to a variety of classifier structures. We have specialized the general approach to obtain specific design algorithms for three distinct, commonly used classifier structures – NP, MLP, and RBF classifiers. For each structure our design approach yielded better classifiers than those obtained by other known methods. For some problems in pattern classification, where the NP, MLP, and RBF classifiers are used, the design methods presented here are immediately applicable. For other application areas, in particular, speech recognition and character recognition, more powerful classifier structures such as hidden Markov models are typically needed to exploit the dependence (either in time or space) between class labels of "neighboring" features. More work is needed to apply the design philosophy presented here to such structures. Optimization of these structures may be pursued in future work. Another potential area of investigation is the extension of the ideas we have developed here to address related optimization problems in source coding and statistics which involve "embedded" classifiers. These include the design of structured vector quantizers and generalized vector quantizers [11], as well as the problem of piecewise regression. Some promising work has already been done in the source coding context [33].

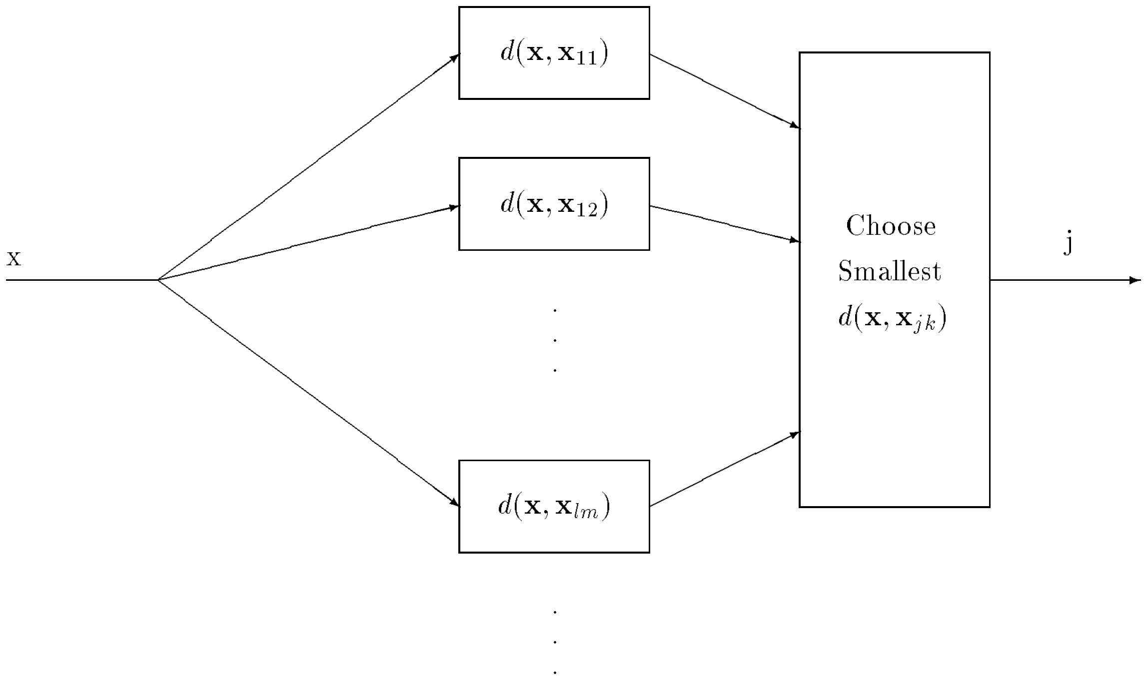


Figure 1: The prototype-based architecture for classification

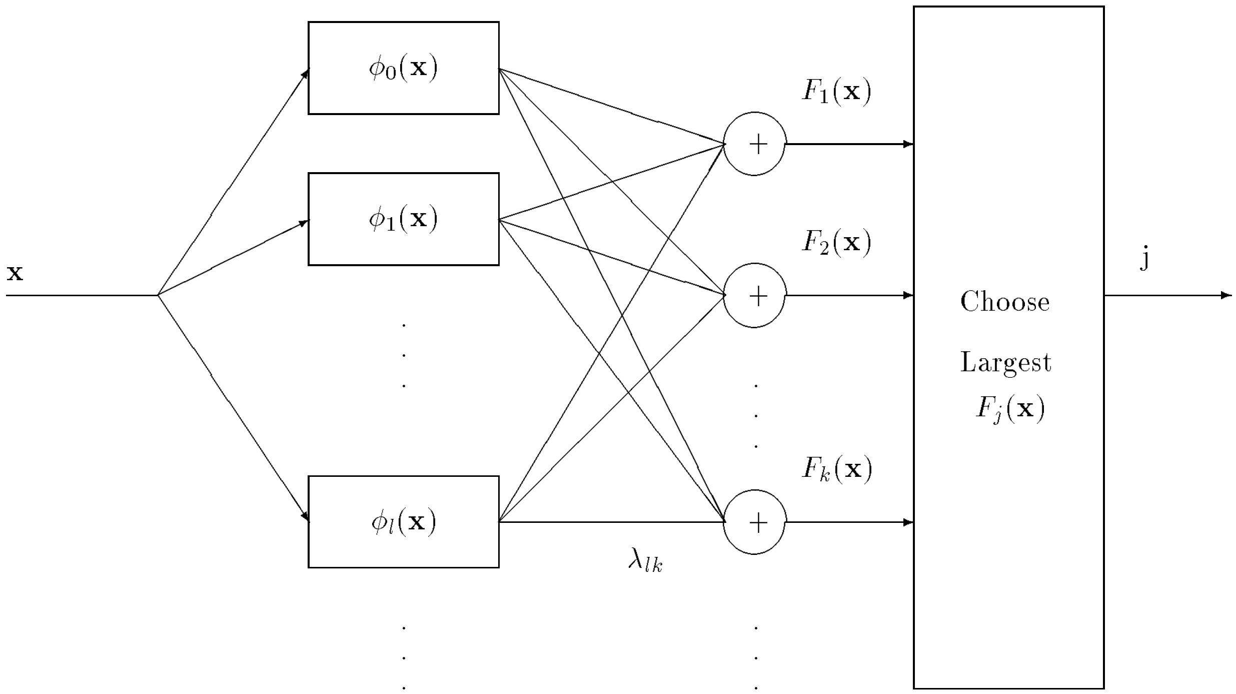


Figure 2: The RBF architecture for classification

References

[1] L. Atlas, R. Cole, Y. Muthusumy, A. Lippmann, J. Connor, D. Park, M. El-Sharkawi, and R. J. Marks II. A performance comparison of trained multi-layer perceptrons and trained classification trees. *Proc. of the IEEE*, 78:1614–1619, 1990.

[2] G. L. Bilbro, W. E. Snyder, and R. C. Mann. Mean-field approximation minimizes relative entropy. *Journal of the Opt. Soc. of Amer.*, 8:290–294, 1991.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series, Belmont,CA., 1980.

[4] D. Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.

[5] P. A. Chou. Optimal partitioning for classification and regression trees. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 13:340–354, 1991.

[6] I. Csiszar. Sanov property, generalized I-projection, and a conditional limit theorem. *Annals of Prob.*, 12:768–793, 1984.

[7] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Roy. Stat. Soc., Ser. B*, 39:1–38, 1977.

[8] H. Do-Tu and M. Installe. Learning algorithms for nonparametric solution to the minimum probability of error classification problem. *IEEE Trans. on Comp.*, C-27:648–659, 1978.

[9] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, NY, 1974.

[10] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRFs: Surface reconstruction. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 13:401–412, 1991.

[11] A. Gersho. Optimal vector quantized nonlinear estimation. In *IEEE Intl. Symp. on Inform. Theory*, San Antonio, TX, 1993.

[12] H. Guo and S. B. Gelfand. Classification trees with neural network feature extraction. *IEEE Trans. on Neural Net.*, 3:923–933, 1992.

[13] J. B. Hampshire and A. H. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Trans. on Neural Net.*, 1:216–228, 1990.

[14] W.H. Highleyman. Linear decision functions, with application to pattern recognition. *Proc. of the IRE*, 50:1501–1514, 1962.

[15] D. Hush, J. M. Salas, and B. Horne. Error surfaces for multi-layer perceptrons. *IEEE Trans. on Sys., Man, and Cyb.*, 22, 1992.

[16] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE Sig. Proc. Mag.*, pages 8–39, January 1993.

[17] E. T. Jaynes. Information theory and statistical mechanics. In R. D. Rosenkrantz, editor, *Papers on probability, statistics and statistical physics*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989. (Reprint of the original 1957 papers in *Physical Review*).

[18] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Comp.*, 6:181–214, 1994.

[19] B.H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. on Sig. Proc.*, 40:3043–3054, 1992.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[21] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE Proc. ICNN*, volume 1, 1988.

[22] B. Kosko. *Neural networks and fuzzy systems*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[23] R. Lippmann. Review of neural networks for speech recognition. *Neural Comp.*, 1:1–39, 1989.

[24] R. P. Lippmann. Pattern classification using neural networks. *IEEE Comm. Mag.*, pages 47–64, 1989.

[25] J. Makhoul, A. El-Jaroudi, and R. Schwartz. Formation of disconnected decision regions with a single hidden layer. In *Intl. Joint Conf. on Neural Networks*, volume 1, pages 455–460, Washington, DC, 1989.

[26] D. Miller. *An Information-theoretic Framework for Optimization with Applications in Source Coding and Pattern Recognition*. PhD thesis, University of California, Santa Barbara, 1995.

[27] D. Miller, A. Rao, K. Rose, and A. Gersho. An information-theoretic framework for optimization with application to supervised learning. In *Abstracts of the International Symposium on Information Theory*, 1995.

[28] D. Miller, K. Rose, and P. A. Chou. Deterministic annealing for trellis quantizer and HMM design using Baum-Welch re-estimation. In *IEEE Intl. Conf. on Acoust., Speech, and Sig. Proc.*, Adelaide, Australia, 1994.

[29] J. Moody and C. J. Darken. Fast learning in locally-tuned processing units. *Neural Comp.*, 1:281–294, 1989.

[30] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels. On the training of radial basis function classifiers. *Neural Net.*, 5:595–604, 1992.

[31] V. Nedeljkovic. A novel multilayer neural networks training algorithm that minimizes the probability of classification error. *IEEE Trans. on Neural Net.*, 4:650–659, 1993.

[32] J. R. Quinlan. Simplifying decision trees. *Intl. Journal Man-Machine Studies*, 27:221–234, 1987.

[33] A. Rao, D. Miller, K. Rose, and A. Gersho. Generalized vector quantization. (To be submitted for publication.), 1995.

[34] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Comp.*, 3:461–483, 1991.

[35] B. D. Ripley. Neural networks and related methods for classification. *Journal of the Royal Stat. Soc., Ser. B*, 56:409–456, 1994.

[36] K. Rose. *Deterministic Annealing, Clustering, and Optimization.* PhD thesis, California Institute of Technology, 1991.

[37] K. Rose, E. Gurewitz, and G. C. Fox. Statistical mechanics and phase transitions in clustering. *Phys. Rev. Lett.*, 65:945–948, 1990.

[38] K. Rose, E. Gurewitz, and G. C. Fox. Vector quantization by deterministic annealing. *IEEE Trans. on Inform. Theory*, 38:1249–1258, 1992.

[39] K. Rose, E. Gurewitz, and G. C. Fox. Constrained clustering as an optimization method. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 15:785–794, 1993.

[40] F. Rosenblatt. The perceptron – a perceiving and recognizing automaton. Technical report, Cornell, 1957.

[41] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Trans. on Neural Net.*, 1:296–298, 1990.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing.* MIT Press, Cambridge, MA, 1986.

[43] J. E. Shore and R. W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Trans. on Inform. Theory*, IT-26:26–37, 1980.

[44] P. D. Simic. Statistical mechanics as the underlying theory of elastic and neural optimization. *Network*, 1:89–103, 1990.

[45] P. D. Simic. Constrained nets for graph matching and other quadratic assignment problems. *Neural Comp.*, 3:268–281, 1991.

[46] F.W. Smith. Pattern classifier design by linear programming. *IEEE Trans. on Comp.*, C-17:367–372, 1968.

[47] E. D. Sontag and H. J. Sussman. Back propagation separates where perceptrons do. *Neural Net.*, 4:243–249, 1991.

[48] L. Tarassenko and S. Roberts. Supervised and unsupervised learning in radial basis function classifiers. *IEE Proc.-Vis. Image Sig. Proc.*, 141:210–216, 1994.

[49] B. A. Telfer and H. H. Szu. Energy functions for minimizing misclassification error with minimum complexity networks. *Neural Net.*, 7:809–817, 1994.

[50] J. M. Van Campenhout and T. M. Cover. Maximum entropy and conditional probability. *IEEE Trans. on Inform. Theory*, IT-27:483–489, 1981.

[51] D. Van den Bout and T. K. Miller. Graph partitioning using annealed neural networks. *IEEE Trans. on Neural Net.*, pages 192–203, 1990.

[52] E. A. Wan. Neural network classification: a Bayesian interpretation. *IEEE Trans. on Neural Net.*, 4:303–305, 1990.

[53] L. F. A. Wessels and E. Barnard. Avoiding false local minima by proper initialization of connections. *IEEE Trans. on Neural Net.*, 3:899–905, 1992.

[54] A. L. Yuille. Generalized deformable models, statistical physics, and matching problems. *Neural Comp.*, 2:1–24, 1990.

[55] A. L. Yuille, P. Stolorz, and J. Utans. Statistical physics, mixtures of distributions, and the EM algorithm. *Neural Comp.*, 6:334–340, 1994.

[56] J. Zhang. The mean field theory in EM procedures for Markov random fields. *IEEE Trans. on Sig. Proc.*, 40:2570–2583, 1992.

M (# of units)	20	30	40	50	80	100	200
P_e (LVQ)	13.25	12.44	11.47	10.96	10.09	8.17	6.78
P_e (DA)	11.67	9.99	8.36	5.55	4.83	4.23	3.26

Table 1: A comparison of the DA and LVQ methods for designing prototype-based classifiers on the 20-dimensional, 20 class Finnish phoneme data set that accompanies the standard LVQ package. M represents the total number of prototypes.

Method	DA	TR-RBF				MD-RBF		G-RBF		
M (# of units)	3	3	5	15	25	10	30	5	10	15
P_e (training set)	14.0	38.0	15.0	14.0	10.0	25.0	18.0	48.0	21.0	14.0
P_e (test set)	16.0	38.0	22.0	18.0	17.0	26.0	19.0	47.0	19.0	16.0

Table 2: A comparison of DA with known design techniques for RBF classification, on the 21-dimensional waveform data from [3]. M is the number of receptive fields. We compare our DA approach with the method described in [38] (TR-RBF), the method in [22] (MD-RBF), and with gradient descent on $< P_e >$ (G-RBF). The test set performances have maximum 95 % confidence interval of half-length 2%.

Method	DA		TR-RBF				MD-RBF		G-RBF
M (# of units)	4	30	4	10	30	50	10	50	10
P_e (training set)	11.0	2.8	33.0	16.2	14.5	12.9	30.0	19.0	18.0
P_e (test set)	13.0	16.7	35.0	16.5	16.8	17.9	37.0	18.0	20.0

Table 3: A comparison of DA with known design techniques for RBF classification on the 40-dimensional noisy waveform data from [3]. The test set performances have a maximum 95 % confidence interval of half-length about 3.0 %.

	DA		BP				G-MLP			
M	3	4	3	4	8	10	3	4	8	10
P_e (training set)	2.4	2.4	7.8	7.4	7.4	3.0	7.4	7.4	7.4	3.0
P_e (test set)	3.8	2.4	7.4	7.2	6.8	4.2	7.2	7.0	7.2	4.4

Table 4: Misclassification performance P_e for the DA, BP, and G-MLP design approaches on the two-dimensional mixture example. The performance is shown for a varying hidden layer size (M). The test set performances have a maximum 95 % confidence interval of half-length about 0.3%.

	DA			BP				G-MLP			
M	4	6	8	4	6	8	10	4	6	8	10
P_e (training set)	19.1	8.1	6.2	45.7	31.9	20.0	6.1	45.7	31.4	19.6	6.0
P_e (test set)	20.1	11.2	10.5	48.1	31.7	25.3	13.3	47.2	34.4	23.2	13.0

Table 5: Misclassification performance P_e for the DA, BP, and G-MLP design approaches on the 19-dimensional 7-class segmentation data example. The test set performances have a maximum 95 % confidence interval of half-length 2.1%.

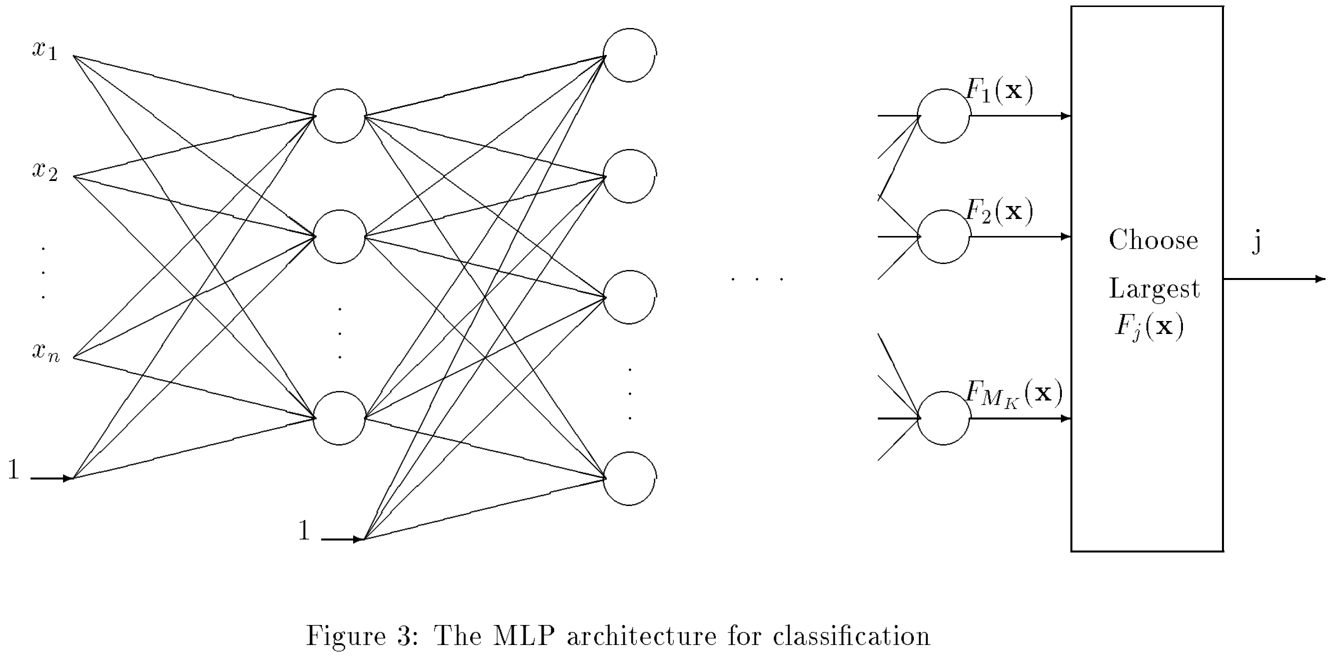


Figure 3: The MLP architecture for classification

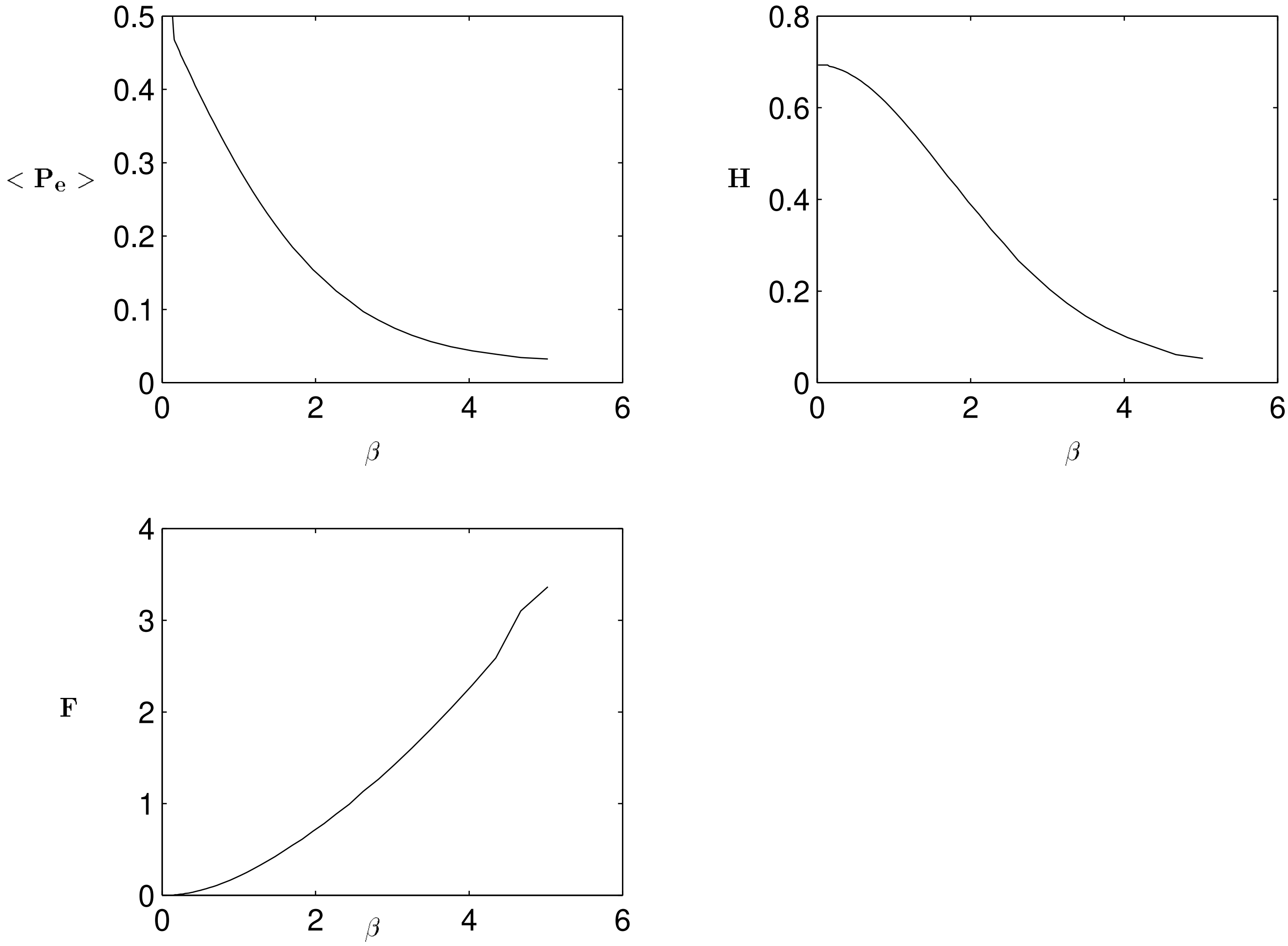


Figure 4: Variation of entropy, H, probability of error, $< P_e >$, and the Winner-Take-All objective, F with β during the training of an RBF classifier.

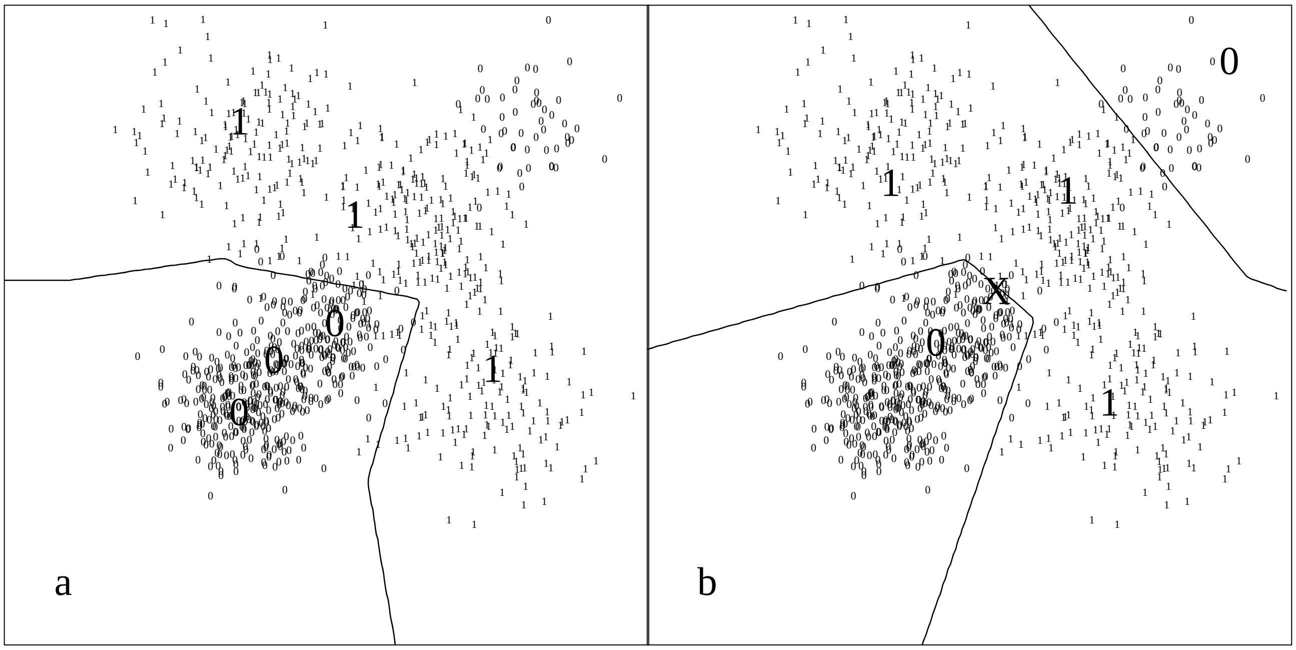
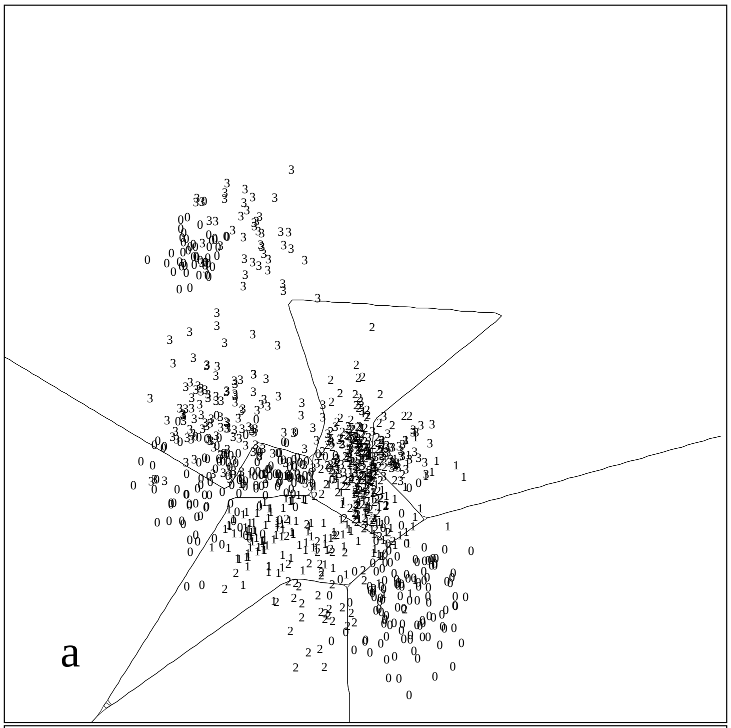


Figure 5: A two-class example for prototype-based classifier design. Each class is a Gaussian mixture consisting of three components: a) The best LVQ solution, using six prototypes, with $P_e = 7.7\%$. b) The DA solution, using five prototypes, with $P_e = 2.7\%$. Note that since the solution at $\beta = 0$ placed all prototypes at the global centroid (X), the DA optimization has allowed a prototype for class 0 to "pass through a wall" of class 1 data in order to correctly classify the minority "0" mixture component.



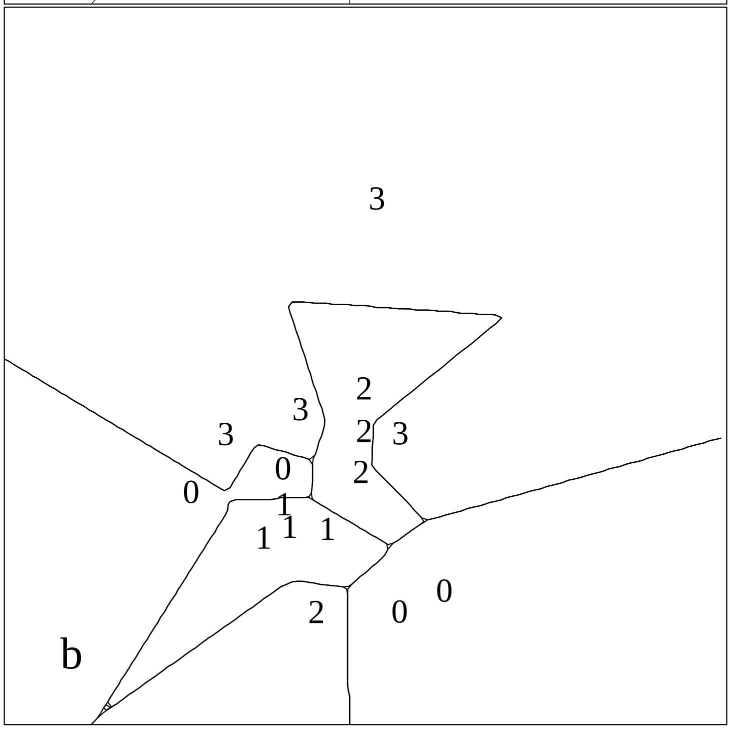


Figure 6: a) A four-class Gaussian mixture training set for a prototype-based classifier design and the partition produced by LVQ. b) The LVQ solution, with the sixteen class prototype. Locations of prototypes shown. The error rate is $P_e = 31\%$.

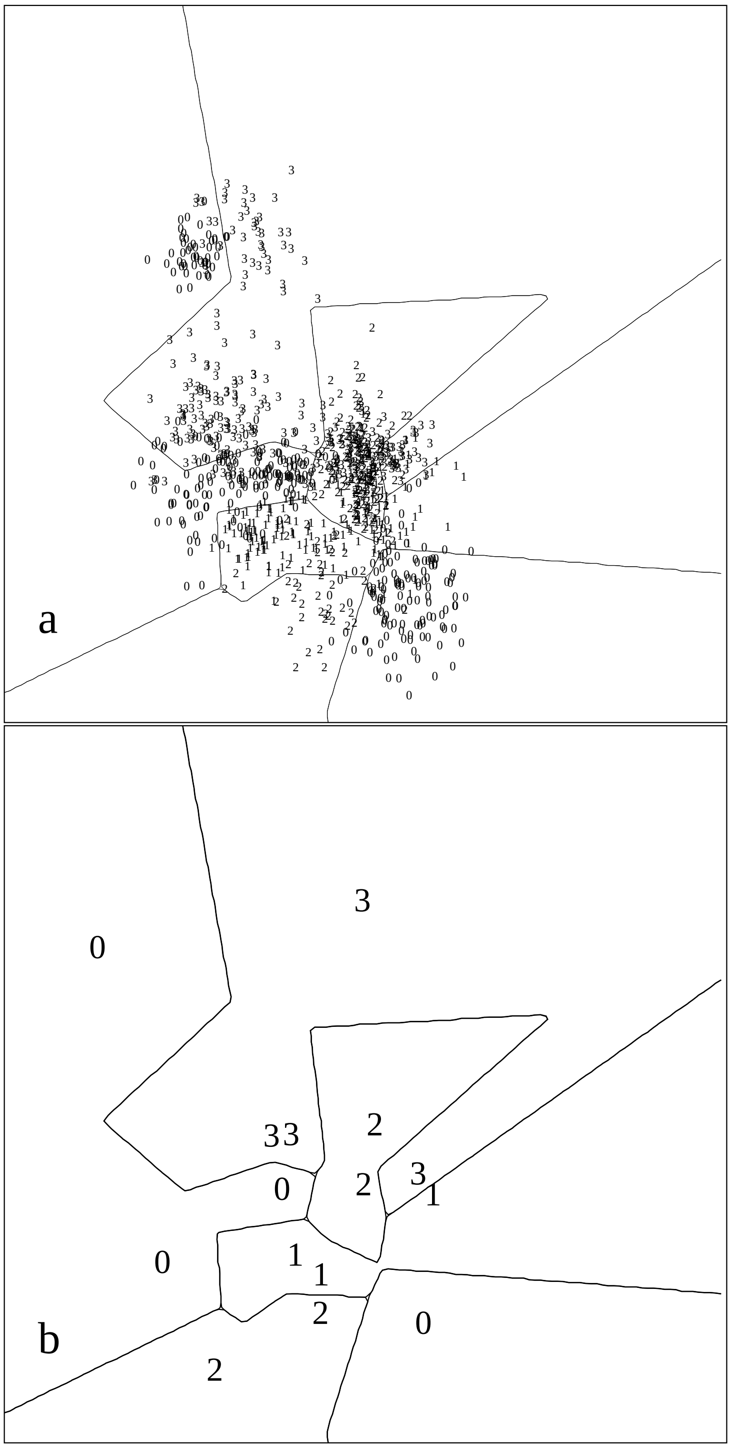


Figure 7: a) The four-class Gaussian mixture training set for a prototype-based classifier design and the partition produced by DA. b) The DA partition with the sixteen class prototypes shown. The error rate is $P_e = 23\%$.

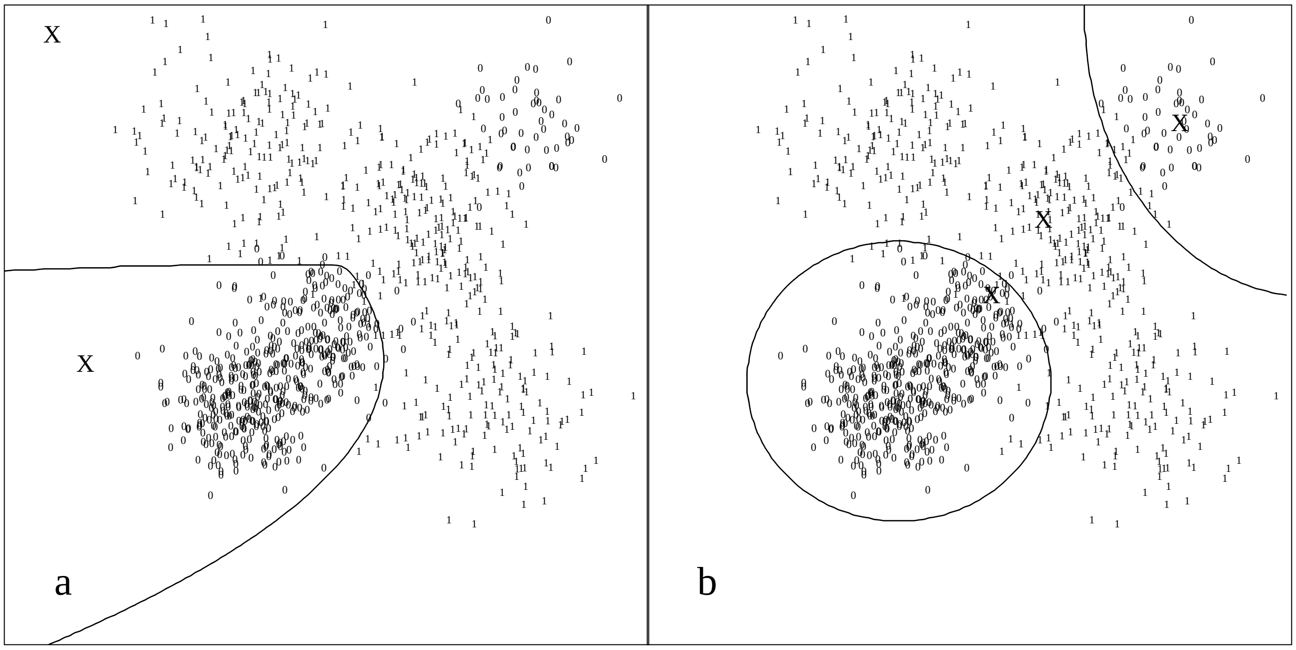


Figure 8: a) An RBF classifier solution with three receptive fields, produced by the method from [38]. The error rate is $P_e = 7.0\%$. b) The DA solution with three receptive fields, which yields $P_e = 2.7\%$. In each case, the receptive field "centers" are indicated with an X. Note that in (a) one of the "centers" lies outside the boxed area.

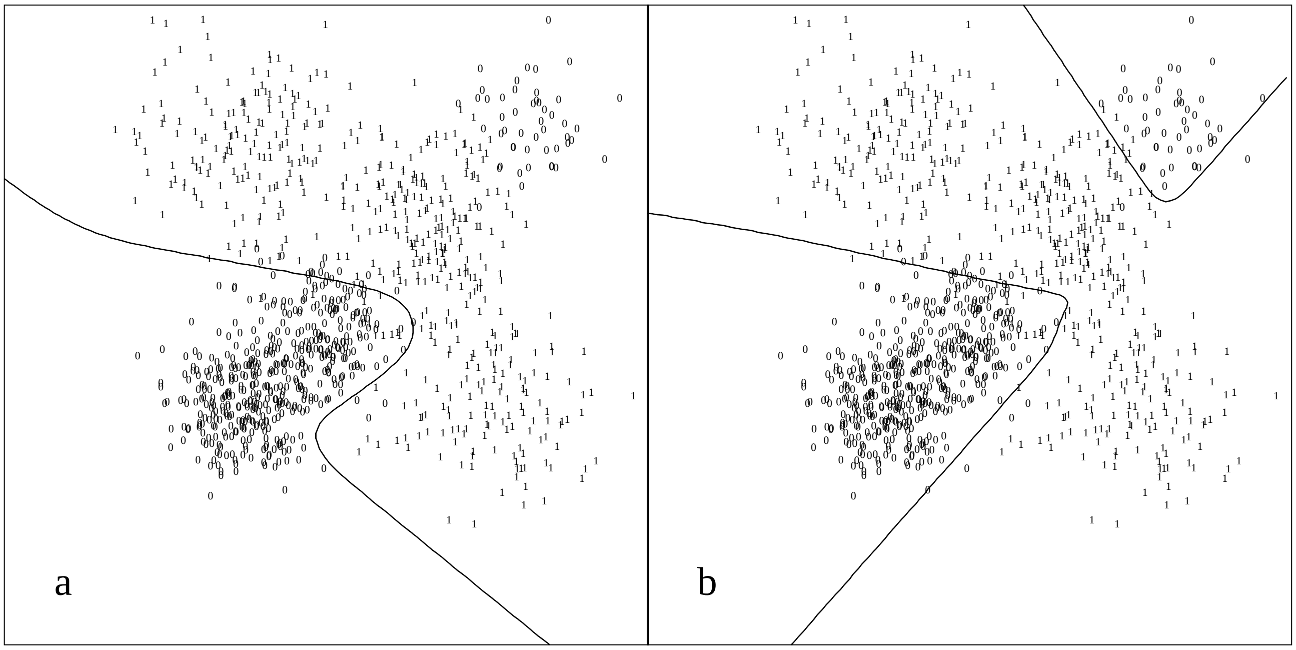


Figure 9: a) The MLP solution with four hidden units found by back propagation, with $P_e = 7.4\%$. b) The DA solution with three hidden units and $P_e = 2.4\%$.