Pattern Classification with Adaptive Distance Measures*

G. Scheler
Institut für Informatik
Technische Universität München
e-mail: scheler@informatik.tu-muenchen.de

February 15, 1994

Abstract

In this paper, we want to explore the notion of learning the classification of patterns from examples by synthesizing distance functions.

A working implementation of a distance classifier is presented. Its operation is illustrated with the problem of classification according to parity (highly non-linear) and a classification of feature vectors which involves dimension reduction (a linear problem). A solution to these problems is sought in two steps: (a) a parametrized distance function (called a 'distance function scheme') is chosen, (b) setting parameters to values according to the classification of training patterns results in a specific distance function. This induces a classification on all remaining patterns.

The general idea of this approach is to find restricted functional shapes in order to model certain cognitive functions of classification exactly, i.e. performing classifications that occur as well as excluding classifications that do not naturally occur and may even be experimentally proven to be excluded from learnability by a living organism.

There are also certain technical advantages in using restricted function shapes and simple learning rules, such as reducing learning time, generating training sets and individual patterns to set certain parameters, determining the learnability of a specific problem with a given function scheme or providing additions to functions for individual exceptions, while retaining the general shape for generalization.

*This paper is also available as Report FKI-188-94, Institut für Informatik, TU München

1 Introduction

The basic idea of a "distance classifier" is to use a variable distance measure and adapt it to the given problem by changing specific parameters in the distance measure and thus synthesize a distance function directly.

It seems clear that this is a general principle, comparable to learning hypersurfaces or learning prototypes, which may be used for many different pattern classification or pattern extrapolation problems. In particular, it might be used for supervised as well as for non-supervised training methods.

In this paper, however, we concentrate on a specific type of classification task, for which an implementation has been made. It can be described by the following points:

- The goal is the classification of **binary patterns**.
- Only **clear-cut classes** are concerned, i.e. there is no fuzzy class membership.
- At least one instance for each class is given (called **reference pattern**).
- A number of **training patterns** is available for the setting of parameters.
- Patterns can be labelled correctly by some independent method (i.e. **error** can be specified for each pattern).

Usually only a limited variability (adaptivity) of the distance measure is necessary. Assumptions are being made concerning the structure of the problem and these lead to a selection of a specific **distance function scheme**. A distance function scheme has a number of parameters, which are being set during learning. Function schemes can be distinguished by the kind of classifications that are expressible with them. A function scheme with limited expressive capability has the advantage of being learned fast and with a small training set. Generalization is determined by the function scheme and it is fixed when all the parameters are learned and thus a specific distance function has been synthesized.

An example is the abstraction of a phonological system from a set of phonetic features. There a certain kind of linear distance is sufficient, which performs feature extraction and requires only local, bitwise parameters. For classification according to parity, global parameters are used, which "count" the number of bits. Parameters in the function scheme correspond to features of the patterns compared.

For some of the distance measures used we have defined a simple single pass learning procedure, by which all classifications representable in a scheme can be learned. A variant of the linear perceptron is also presented, for which an iterative learning procedure, namely the delta-rule ([Pao89],p.119) is used. Finally we have defined a distance function scheme with universal expressive power for the two category case; i.e. a distance function scheme, by which all classifications of binary patterns in two classes can be expressed.

A classifier which is built by using distance function schemes together with the single pass learning rule has specific properties, which distinguishes it from other classifiers:

- Knowledge of the problem domain (i.e. characteristics of the goal function) is used to constrain the search space of possible classifications.

- The single pass learning rule finds a single classification function, if it fits to the training set. Generalization is not a separate issue, but constrained by function scheme and training set.

- If a classification function can not represent a training set, a failure can be generated and either exception handling or selection of a new distance function can be initiated.

- A set of training patterns sufficient for setting a certain number of parameters can be generated. These may be labeled by an independent method and given to the classifier for learning.

2 Examples for Distance Function Schemes

A distance function can be made *adaptive* and *trainable*, if we introduce certain parameters in the function definition, which can be set during a learning process and modify the influence of properties of the input patterns on the value of the distance function. Thus these parameters correspond to features of the input patterns.

We will call such a parametrized version of a distance function a **distance function scheme** to distinguish it from its instantiations (where all parameters are set to values), which are regular **distance functions** and have to fulfill all the properties of a metric:

- positive values only ($d(x, y) \geq 0$)

- reflexivity ($d(x, x) = 0$)

- symmetry ($d(x, y) = d(y, x)$)

- triangle inequality ($d(x, y) + d(y, z) \geq d(x, z)$)

A distance function computes a measure of similarity between patterns. Usually this will be real-valued. When we learn a distance function from examples, however, it depends on the available training data, which values for the distance function are possible.

Here, we have assumed that only the classificational data are given. I.e. it is sufficient to distinguish between two values for the distance function, one indicating that two patterns belong to the same class, and one indicating that they do not.

For the classification of binary patterns, the values of distance functions must be consistent with a clear-cut class distinction, i.e. they need to fulfill a stronger version of the metric property, the classification property.

$$\begin{array}{lll} \text{Given :} & & \text{Implies :} \\ d(x,y)=0 & d(y,z)=0 & d(x,z)=0 \\ d(x,y)=1 & d(y,z)=0 & d(x,z)=1 \\ d(x,y)=0 & d(y,z)=1 & d(x,z)=1 \\ d(x,y)=1 & d(y,z)=1 & d(x,z)\in\{0,1\} \end{array}$$

Two theoretically easily analysable, but still interesting and instructive distance function schemes are discussed in the next sections. The first one achieves a feature selection on a set of input patterns, the second one can be used to perform a classification according to parity, as well as to other classifications based on the number of bits set.

2.1 Feature Selection

Given two binary patterns a and b with:
$a := \langle a_1, \ldots, a_n\rangle$
$b := \langle b_1, \ldots, b_n\rangle$
$a_i, b_i \in \{0,1\},\ 1 \le i \le n.$
a distance function scheme d_λ can be defined as:

$$d_\lambda(a,b) = \sum_{i=1}^{n} \lambda_i * |a_i - b_i|$$

$\lambda_i \in R_0^+$

In order to prove that all instantiations of a distance function scheme are valid distance functions, we have to check the four conditions on a metric. This has been done in Appendix A.

The functioning of d_λ as a feature selection mechanism can be illustrated with an abstract example. Given are patterns of length 7 such as:

$A = \langle 0101011\rangle$

$B = \langle 0111010\rangle$

$C = \langle 1101011\rangle$

and a classification as $class(A) = class(B) \neq class(C)$ as a training set.

This means that the values for the distance functions are as follows:

$$d_\lambda(A, B) = 0$$

$$d_\lambda(A, C) = 1$$

$$d_\lambda(B, C) = 1$$

The last condition follows from the classification property, i.e. it is not used for learning.

These conditions are inserted into our definition of d_λ and we get the following equations:

$$\begin{array}{ll} d_\lambda(A, B) = 0 = \lambda_3 + \lambda_7 & \Rightarrow \lambda_3 = \lambda_7 = 0 \\ d_\lambda(A, C) = 1 = \lambda_1 & \Rightarrow \lambda_1 = 1 \\ (d_\lambda(B, C) = 1 = \lambda_1 + \lambda_3 + \lambda_7) & \end{array}$$

The result can be interpreted as follows: For this classification, a difference at positions 3 and 7 has no distinguishing function, while a difference at position 1 distinguishes two classes. These positions correspond to individual features of binary patterns. By applying d_λ, therefore, we can determine which features are necessary for classification and which features are irrelevant.

It is also intuitively clear that no setting of parameters can be found, if individual features are sometimes used as discriminants and sometimes not. E.g. a fourth pattern $D = \langle 0111011 \rangle$ cannot be classified in a separate class.

$$d_\lambda(A, D) = \lambda_3 = 0$$

$$d_\lambda(B, D) = \lambda_7 = 0$$

$$d_\lambda(C, D) = \lambda_1 + \lambda_3 = 1$$

In other words, generalization to pattern D is given by the form of the function scheme, which classifies according to distinguishing positions in input patterns.

This rather fixed generalization property can be an advantage or a disadvantage depending on the goals of the modeler. When a flexible classifier is needed, which can solve many problems, d_λ is not very useful. However, when a specific classification faculty is to be modelled, it is important to model the faculty exactly, i.e. both success in classification as well as failure to implement certain classifications (such as $class(A) = class(B) \neq class(D)$) must be accounted for.

If a faculty with the properties of a d_λ-classifier is the goal of modelling, there are also certain technical advantages in using a simple function shape instead of a general model.

Consider pattern $E = \langle 0001101 \rangle$. E differs from A in positions 2, 5 and 6 and from C in position 1. Thus E cannot belong to $class(C)$, but it may belong to $class(A)$ or not. However, when the distance value for A and E is 0, three parameters will be set to 0. This means that all patterns (i.e. 6) with $d_\lambda(A, X) = \sum_{i \in 2,5,6} \lambda_i$ will automatically be classified.

In a separate paper ([Sch94]) it has been shown that such a mechanism (where the class identity of different patterns is a powerful indicator for the class identity of other patterns with the same features) may be present in the classification of segmental phonetic descriptions to phonemes.

2.2 Classification according to Parity

Another distance function scheme, which uses global parameters, is called d_c:

$$d_c(a,b) = |\sum_{i=0}^{n} \gamma_i * \delta((\sum_{k=1}^{n} a_k) - i), 0) - \sum_{i=0}^{n} \gamma_i * \delta((\sum_{j=1}^{n} b_j) - i), 0)|$$

$\gamma_i \in R_0^+$

$$\delta(x,0) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The metric properties for d_c are shown in Appendix A. d_c allows to implement classification according to parity in a very simple way. There are $n+1$ different parameters γ_i, each indicating the number of 1-bits in a pattern. Obviously, a classification according to the parity of the number of 1-bits will be achieved, when all γ_i with odd i have the same value, and all γ_i with even i have the same value, which is different from the other. E.g. (for $n = 5$), $\gamma_0 = \gamma_2 = \gamma_4 = 0$ and $\gamma_1 = \gamma_3 = \gamma_5 = 1$.

d_c can be learned with the same learning procedure as d_λ. This shows that simplicity/complexity of learning a function (in the context of learning classifiers) is not the same distinction as that between linear/non-linear functions. Certain non-linear functions, e.g. parity, are much easier to learn than many linear problems, which require an iterative learning procedure, if it is known that e.g. global features (number of bits) are necessary for the classification.

Compare this to a general learning scheme such as back-propagation ([Whi93], [Hor93], [HSW89], [RHW86]).

Simulations show that with back-propagation classification according to parity can be learned, i.e. a set of training patterns can be stored, but there is no generalization when individual patterns are left out from the training set. E.g., for input patterns of length 5, even with 31 out of 32 patterns given as training set, the remaining test pattern is not classified correctly.

It seems that generalization is based on Hamming distance to a statistical average of the elements in a class. I.e. the similarity criterion, the metric of the input space is built-in, and can only statistically be changed.

Generalization is dependent on the function shape that is being learned, which in turn is dependent on the number of hidden units ([BH88]). Accordingly, there are phenomena of "overfitting", where too many hidden units lead to a false abstraction of the function shape. With "pruning" and "growing" techniques, which automatically tune the number of hidden units, this may be corrected, but not in the sense that a completely different generalization strategy is being employed. Furthermore these techniques are computationally costly, which make them unlikely candidates for basic models of biological computation. In a case where an analysis of the problem situation leads to an understanding of the type of function that is needed (such as feature selection–dimensionality reduction or classification taking the number of bits in a pattern into account), adaptation to a specific problem can be effected by a training set at less computational cost and with better, even with perfect generalization.

3 Representing a Set of Functions by Function Schemes

An important notion for function schemes is their **representational power**, i.e. the number of individual functions that can be represented by the parameter settings in the scheme.

Naturally, this provides also an upper bound on the number of functions that can be learned. When all theoretically possible parameter settings are effectively "reachable" with a learning algorithm, representable functions and learnable functions become the same thing. Questions of learnability are addressed in the next section.

In this section, we also define a distance function scheme which is equivalent in representational power to the linear perceptron. It becomes evident that we can easily transform classification functions in general into distance functions. Finally we define a function scheme with universal representational power for two-class categorizations of binary patterns with fixed length. This is achieved by using a function scheme for Boolean conjunctions of negated or nonnegated binary elements, and enhancing this with disjunctions to provide a distance function scheme for arbitrary classifications. It is easy to see that this is a universal function scheme, because each instantiation renders a propositional formula in disjunctive normal form (DNF) for the characteristic function of a single class.

3.1 Representational Power of d_λ

The representational power of a distance function scheme may depend on the valid values for its parameters. This is illustrated for d_λ here for $n = 2$. All possible classifications with d_λ (with nonnegative parameters) are shown in Table 1.

distance values	parameter setting	classification
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 01\rangle\}$ $C_2 = \{\langle 10\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 10\rangle\}$ $C_2 = \{\langle 01\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 01\rangle, \langle 10\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle\}$ $C_2 = \{\langle 01\rangle\}$ $C_3 = \{\langle 10\rangle\}$ $C_4 = \{\langle 11\rangle\}$

Table 1: Classification with d_λ with nonnegative parameters

When parameters are set to real values between 0 and 1, and a threshold determined, the classification property is not always guaranteed (cf. Table 2). Because $d_\lambda(\langle 00\rangle, \langle 11\rangle) = d_\lambda(\langle 01\rangle, \langle 10\rangle) = \lambda_1 + \lambda_2$ there is no classification that fulfills the third condition. Otherwise d_λ would be equivalent to d_l (see next section).

A different classifier results if we consider negative values and define an appropriate filter for the values as $d_\lambda^{neg} = sgn(sgn(d_\lambda) + 1)$ (cf. Table 3).

For arbitrary n, and nonnegative parameters the representational power of d_λ can be determined in the following way:

If p denotes the number of λ_i set to 0 in d_λ, p is the number of positions which have the same bit-value for all patterns within a class. This means that $\begin{pmatrix} p \\ n \end{pmatrix}$ different classifications with 2^{n-p} patterns of length n per class can be obtained with d_λ. In geometrical terms, patterns that all have the same value at a certain number of positions must be edges or planes of a hypercube. The representable classifications therefore form a subset of linearly separable patterns.

distance values	parameter setting	classification
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda_1 < \theta$ $\lambda_2 > \theta$	$C_1 = \{\langle 00\rangle, \langle 10\rangle\}$ $C_2 = \{\langle 01\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda_1 > \theta$ $\lambda_2 < \theta$	$C_1 = \{\langle 00\rangle, \langle 10\rangle\}$ $C_2 = \{\langle 01\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda_1, \lambda_2 < \theta$ $\lambda_1 + \lambda_2 > \theta$	*no classification*
$d_\lambda = \begin{cases} 0 \\ 1 \end{cases}$	$\lambda_1 + \lambda_2 < \theta$	$C_1 = \{\langle 00\rangle, \langle 01\rangle, \langle 10\rangle, \langle 11\rangle\}$

Table 2: Classification with d_λ with threshold values, $\theta \in R_0^+$

3.2 Linear Separability

A distance classifier with the representational power of the linear perceptron (and many equivalent formulations) is the **linear distance measure scheme**, d_l:

$$d_l(a,b) = 1 - sgn((\sum_{i=1}^{n} \alpha_i a_i) - \theta) \times ((\sum_{i=1}^{n} \alpha_i b_i) - \theta)$$

$$\alpha_i, \theta \in R$$

The number of classifications of binary patterns the classifier can represent is equal to the linear perceptron. This is shown here for the two-category case. (An extension to the multi-category case is always possible on the grounds of combining several two-category classifiers (cf. [DH73] p.132).)

Theorem 1 *d_l is sufficient for the classification of linearly separable patterns.*

Proof: Let a, b be two patterns.

Case 1: $class(a) \neq class(b)$, i.e. $d_l(a,b) = 1$. This means that

$$\sum_{i=1}^{n} \alpha_i a_i \leq \theta \; and \; \sum_{i=1}^{n} \alpha_i b_i > \theta \; or$$

distance values	parameter setting	classification
$d_\lambda = \begin{cases} 0 \\ -1 \end{cases}$	$\lambda = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 01\rangle\}$ $C_2 = \{\langle 10\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} -1 \\ -2 \end{cases}$	$\lambda = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 11\rangle\}$ $C_2 = \{\langle 10\rangle, \langle 01\rangle\}$
$d_\lambda = \begin{cases} 0 \\ -1 \end{cases}$	$\lambda = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 10\rangle\}$ $C_2 = \{\langle 01\rangle, \langle 11\rangle\}$
$d_\lambda = \begin{cases} 0 \\ -1/1 \end{cases}$	$\lambda = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 11\rangle\}$ $C_2 = \{\langle 10\rangle, \langle 01\rangle\}$
$d_\lambda = \begin{cases} 0 \\ 1/-1 \end{cases}$	$\lambda = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$	$C_1 = \{\langle 00\rangle, \langle 11\rangle\}$ $C_2 = \{\langle 10\rangle, \langle 01\rangle\}$

Table 3: Classification with d_λ with negative values

$$\sum_{i=1}^{n} \alpha_i a_i > \theta \ and \ \sum_{i=1}^{n} \alpha_i b_i \leq \theta$$

In vector notation:

$$\vec{\alpha}a + (-\theta) \leq 0 \ and \ \vec{\alpha}b + (-\theta) > 0. \ or$$

$$\vec{\alpha}a + (-\theta) > 0 \ and \ \vec{\alpha}b + (-\theta) \leq 0.$$

Case 2: $class(a) = class(b)$, i.e. $d_l(a,b) = 0$.

$$\sum_{i=1}^{n} \alpha_i a_i \leq \theta \ and \ \sum_{i=1}^{n} \alpha_i b_i \leq \theta \ or$$

$$\sum_{i=1}^{n} \alpha_i a_i > \theta \ and \ \sum_{i=1}^{n} \alpha_i b_i > \theta.$$

In vector notation:

$$\vec{\alpha}a + (-\theta) \leq 0 \ and \ \vec{\alpha}b + (-\theta) \leq 0. \ or$$

$$\vec{\alpha}a + (-\theta) > 0 \ and \ \vec{\alpha}b + (-\theta) > 0.$$

In both cases, $\vec{\alpha}$ is the discriminant of the linear perceptron.

We see that we have in effect a distance measure which is based on two identical classification functions for patterns a and b, and compares their functional values.

$$d(a,b) = 1 - sgn(f(a) \times f(b))$$

The same scheme can therefore be used for any other function that classifies patterns with negative and positive function values.

3.3 Conjunction of Elements and Disjunctive Normal Form

We have seen that learning a distance function can also be achieved by learning two classification functions directly. We can use this composition of a distance function to define a universal distance function scheme. Distance functions can be construed as Boolean functions related to characteristic functions of individual classes in a specific way.

$$\begin{array}{ll} d(a,b) = f^1(\langle ab \rangle) = 1 & \text{if} \quad f^2(a) \neq f^2(b) \\ d(a,b) = f^1(\langle ab \rangle) = 0 & \text{if} \quad f^2(a) = f^2(b) \end{array}$$

where $\langle ab \rangle$ is the concatenation of a and b, and f^1, f^2 the related Boolean functions. This corresponds to the following scheme:

$$d(a,b) = |f^2(a) - f^2(b)|$$

Accordingly, we may define a universal distance function scheme, i.e. a distance function scheme which can represent all possible (two-class) classifications of binary patterns of given length, as the disjunctive normal form of the characteristic function of either class. For patterns of length n, and $k \leq 2^{n-1}$, we define

$$d_{dnf}(x,y) = |(\sum_{j=1}^{k}(\prod_{i=1}^{n}(|\alpha_i^k - x_i|)) - \sum_{j=1}^{k}(\prod_{i=1}^{n}(|\alpha_i^k - y_i|)))|$$

k is a parameter for the number of disjunctions which are necessary to describe a pattern set fully. It has a theoretical limit in half of the total pattern set.

Retranslating arithmetic terms into Boolean terms, and disregarding the distance function scheme "environment" we get:

$$f(x) = \bigvee_{j=1}^{k} \bigwedge_{i=1}^{n} \alpha_i^k x_i$$

where α_i^k indicates if x_i is negated or not. This is equivalent with a formulation of the training set as disjunctive normal form of a formula enumerating the training set. It is obviously sufficient to describe an arbitrary set of patterns fully.

There is a simplification of the functional shape of d_{dnf} when all α_i^k with the same k-index have the same value:

$$d_{conj}(x, y) = |(\prod_{i=1}^{n}(|\alpha_i - x_i|) - \prod_{i=1}^{n}(|\alpha_i - y_i|)|$$

There are sets of patterns that can be described by setting certain parameters α_i to specific values (i.e. requiring the same bit value, 0 or 1, at a certain position), and setting others to 0, i.e. don't care positions. This means that a set of specific classification functions can be derived from this formulation.

In other words, functions that can be described by a Boolean formula of the type

$$x_i \wedge \neg x_j \wedge x_k \ldots$$

can be represented by d_{conj} alone. There are also various intermediate forms with a restricted number of k ($<< 2^{n-1}$) possible.

The restricted capacity of d_{conj} to represent classificational functions can be considered positively as imposing a certain generalization strategy on an incompletely specified classification function.

A learning rule to set parameters in d_{dnf} or d_{conj} has not been defined yet. Combining d_{dnf} with an iterative, approximative learning scheme (cf. Section 4) would render a universal function approximator.

However, it is important to note that d_{dnf} constitutes a distance function scheme with universal representational power by virtue of its translatability into a disjunctive normal form representation of a Boolean function.

4 Learning Procedures

When a function scheme has been defined, learning consists in setting parameters in the function scheme to values. We can distinguish between two different kinds of learning procedures:

- single-pass learning, where each parameter is set to a value only once
- iterative learning, where parameters are repeatedly set until some result (e.g., not exceeding a certain error level) has been achieved. Iterative learning may consist in a gradual approximation of the final value for a parameter, or in a random or controlled resetting of parameters to a few values, until a stable distribution of parameter values has been achieved.

Here we want to discuss the single pass learning rule which is sufficient for the setting of parameters in d_λ and d_c and briefly note an approximative, iterative learning procedure, the delta-rule, which is used for d_l. Finally we want to discuss an extension to learning which consists in the setting of parameters by the explicit storage of certain exceptions. Exceptions do not influence the functional shape, therefore generalization is based on the "regular" patterns only. This allows to implement the distinction between individual stored examples and patterns which induce a rule.

4.1 The Single-Pass Learning Rule

For the given distance function schemes d_λ and d_c, there exists an effective learning procedure.

A classification is expressible with a given distance function scheme, if there exists a setting of parameters that implements a distance function which achieves this classification. It is learnable when this setting of parameters can be reached by a learning procedure.

The parameters in d_λ and d_c can be set in a single step, when distances within classes have precedence over distances between classes.

(a) Let a, b be two patterns from the same class. $d(a, b)$ is a function, where certain parameters occur:

$$d(a, b) = F[\ldots \lambda_i \ldots] = 0$$

Then all occurring parameters λ_i can be set to 0.

(b) Let a, b be two patterns from different classes.

$$d(a, b) = F[\ldots \lambda_i \ldots] \geq 1$$

Then one of these parameters is set to 1.

Resetting of parameters is then unnecessary. If a parameter setting conflict occurs, a classification is not representable with the given function scheme.

Generalization has a chance element in a situation where not all parameters can be set with a given training set. An alternative formulation, setting all λ_i in (b) to 1, will lead - only in a situation with an underdetermined training set - to a different generalization strategy. As we have the possibility of generating training patterns to set individual parameters (cf. Section 4.2), the best practical variant consists in generating additional training patterns to allow a non-random setting of parameters. Note that this need not be the full training set that is required to set all parameters.

This is a learning rule with only linear complexity, i.e. for patterns of length n, there are n (or n+1) parameters to set, which is done in one step.

It is also a learning rule by which all representable classifications can be reached, i.e. the set of representable functions becomes the set of learnable functions.

4.2 Features of the single pass learning scheme

There are two useful features that this learning scheme offers in the context of a distance classifier:

1. During computation, the classifier will note if a classification is not expressible with a given distance measure.

   When a classification is not representable with a given distance measure, this means that no setting of parameters will achieve it. If all parameters are set during learning, a conflict will occur, i.e. an attempt to set a different value to a set parameter.

   Such a conflict can be easily monitored and reported.

2. The classifier can recognize an underdetermined training set and create a new pattern for training to complete the set.

   When the training set given is underdetermined, quite naturally, certain parameters will not be set by the learning mechanism. There may also exist an indeterminacy in setting a parameter which occurs only in combination with other equally indeterminate parameters.

   To solve these cases, it is not difficult to implement a mechanism of generating a training pattern which will set a given unknown parameter. For d_λ and d_c the following steps have to be performed:

d_λ
1. Assume λ_i is not set.
2. Take any classified pattern p
3. Find a pattern x, such that $$d_l(x,p) = \lambda_i$$ (Change the ith position of p in order to create x)

d_c
1. Assume γ_i is not set.
2. Take any classified pattern p, such that p has k 1-bits and γ_k is set
3. Find a pattern x, such that $$d_l(x,p) = \lvert\gamma_k - \gamma_i\rvert$$ (Any pattern with i 1-bits)

It is also possible to classify patterns with an incomplete number of known parameters: For a problem which contains only a subset of theoretically possible binary patterns, this reduces the complexity of learning.

4.3 Iterative Parameter Estimation

Iterative parameter setting is used to set the parameters in d_l. The delta-rule is used here:

$$\Delta\alpha^k = \rho(out^k - \alpha^{k-1}x^k)x^k)$$

(for details see [Pao89],p.119). This is also a learning rule that will always find the correct classification, if it is representable with d_l, i.e. if the patterns are linearly separable [MP69]. A failure of learning a training set can be indicated by an error measure, and it shows that a training set is not representable with the function scheme. In this case, there are the same options as with the single-pass learning procedure, namely selecting an alternative function scheme, or using exception handling.

4.4 Exception handling

By using a certain distance function scheme as the basis for classification, we impose a specific classification and generalization scheme. Some classifications are not learnable in the sense that certain training patterns are consistently misclassified. Instead of looking for a different function scheme, we may want to retain the generalization properties of a simple scheme, and add certain patterns as explicit exceptions to the classification function. In comparison to the option of choosing a distance function scheme with higher representational power (a more general function approximator) this has the advantage that generalization can be considered as rule abstraction which is restricted to "regular patterns" rather than including exceptions.

In a situation, where all exceptions can be listed as training examples, e.g. a situation, where exceptions will be statistically present in the learning set because of their extreme frequency, such as in natural languages, 100% correctness of the learned pattern classifier can be achieved. Thus exception handling offers an explanation to the analogical factor in language acquisition and language change, which has become known as "overgeneralization".

There are two cases of misclassifications of two patterns a and b without exception handling:

- $d_\lambda(a, b) = 1$ where it should be 0.

  The goal is to multiply the distance function by a term, which yields 0 when a and b are compared and 1 otherwise.

  The simplest expression for the multiplication term is the Boolean function $f_{\neg(ab)}$ which is

$$f_{\neg(ab)}(x) = \begin{cases} 0 & \text{if } x = \langle ab \rangle \\ 1 & \text{otherwise} \end{cases}$$

This function can be written as

$$f_{\neg(ab)}(x) = \neg(\bigwedge_i^n \chi_i \wedge \bigwedge_i^n \epsilon_i)$$

where

$\chi_i = x_i$	if $a_i = 1$	$\epsilon_i = y_i$	if $b_i = 1$
$\chi_i = \neg x_i$	if $a_i = 0$	$\epsilon_i = \neg y_i$	if $b_i = 0$

By substituting $\neg x$ by $(1-x)$ and $\wedge$ by $\times$ we can translate this function into an arithmetic expression. If n is the length of an input pattern, a Boolean function $f_{\neg(ab)}$ requires $2n$ variables.

- $d_\lambda(a,b) = 0$ where it should be 1.

  Here we want to add a term to the distance function which is 1 for a and b as input and 0 otherwise.

  Again the expression simplest to define is a function f_{ab}, defined like $f_{\neg(ab)}$ above.

There are several possibilities of simplifying an expression achieved in this way (multiplicational as well as additional terms) based on the observation that each Boolean function f may be represented by another function f^* as long as the following equations hold:

$f^*_{ab}(\langle xy\rangle) = 1 = f_{ab}(\langle xy\rangle)$	if $a = x$ and $b = y$
$f^*_{ab}(\langle xy\rangle) = 0 = f_{ab}(\langle xy\rangle)$	for x, y such that $class(x) = class(y)$
$f^*_{ab}(\langle xy\rangle) = \omega$	ω arbitrary, for all other patterns x, y with $class(x) \neq class(y)$ or x, y do not occur in problem space

From this it follows that simplifications have to be performed dependent on the actual classes that exist. A certain strategy for simplification has been developed in [Sch94], which allows the introduction of 'secondary rules', i.e. a generalization from the exceptions. This will not be pursued here.

5 Implementation of an Adaptive Distance Classifier

In Figure 1 a block diagram of the distance classifier is shown, as it is implemented using Mathematica [Wol90] as programming language.

In order to start a classification, we have to set up one example for each class, called the 'reference pattern' and define a set of training patterns as the set of reference patterns plus additional examples. Secondly, we have to

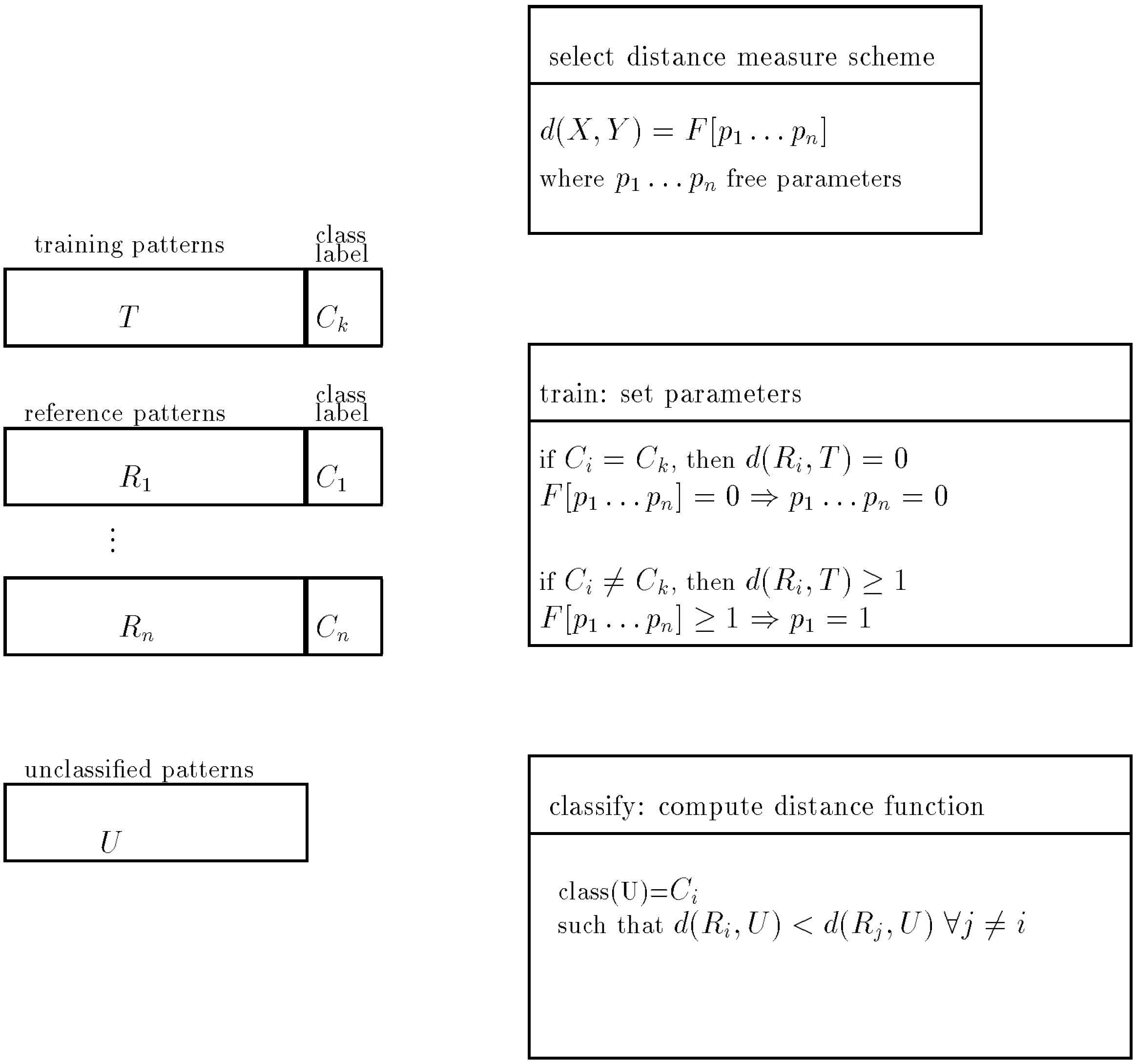

Figure 1: The distance classifier

select a distance function scheme as template for the distance function that is being applied.

In the current implementation, the distance function schemes d_λ (written as `dl`), d_c (`dc`), d_l (`dlin`), and d_{dnf} (`ddnf`) are provided:

```
dl[p1_,p2_]:= Sum[ lambda[i]*abs[p1[[i]],p2[[i]]],{i, patternlength} ]
dc[p1_,p2_]:= abs[ Sum[gamma[i]* gr[p1,i], {i,0,patternlength}],
        Sum[gamma[i]* gr[p2,i],{i,0,patternlength}] ]
        gr[p_,i_]:=If[qs[p]==i,1,0]
        qs[p_]:= Sum[p[[i]],{i,patternlength}]
dlin[a_,b_]:= 1-Sign[(Sum[alpha[[i]]*a[[i]],{i,n}]-theta)*
```

```
        (Sum[alpha[[i]]*b[[i]],{i,n}]-theta
ddnf[a_,b_]:= Abs[( Sum[ Product[f[gamma[i][k], a[[k]]],{k,n}] ,{i,p}]
        - [( Sum[ Product[f[gamma[i][k], b[[k]]],{k,n}] ,{i,p}]
```

The implementation of the learning rule has the following parts:

(a) Generate a full training set for a distance function scheme, i.e. a training set which can set all the parameters.

(b) For each pattern from the training set, determine its class. The distance value to every other member of this class has to be 0, i.e. every parameter in the sum (e.g. $\lambda_2 + \lambda_5 + \lambda_6$) is set to 0.

(c) Take two patterns from different classes. Set one of the parameters λ_i to 1. Here a conflict in parameter setting can occur, when the training set is not representable with the distance function scheme.

(d) For a full training set, all parameters will be set, and all further patterns will be classifiable.

Note that the generalization has a random element, if not the whole training set is used, which originates from the random setting of a parameter λ_i to 1. As a variant of this, consider the following:

(c') Take two patterns from different classes. If $d(a,b) = \lambda_i$, set λ_i to 1. If $d(a,b) = \lambda_i + \lambda_j + ..$, generate patterns to set $\lambda_i, ..$ individually. In this fashion a training set is generated which is sufficient to set a certain subgroup of parameters, yet may be smaller than the full training set (and leave certain parameters unset).

This approach is more useful, when a training set is given by the structure of a problem, and individual additional patterns may be classified correctly.

When all parameters in a distance function scheme have been set, a distance function which acts as a discriminant for the classification given by the examples has been found.

A distance function scheme will induce a classification from examples due to the 'bias' contained in its set of parameters. For instance, if a set of training patterns is given to d_λ and it is compatible with d_λ, the discriminant will always be linear, as described above. Giving the same set of patterns to another distance function scheme may result in a different, but equally possible discriminant. Thus we have the possibility of examining problems, coded as pattern sets, for the kinds of functions that can be used to represent them.

We may classify patterns with an incomplete set of parameters and let the classifier construct further training examples, when a more complete set of parameters is required. Generation of training examples however has not been implemented yet.

When the set of training patterns would require a single parameter to have different values, this shows that the classification is not representable with the distance function scheme. There are several possibilities to pursue in such a case, none of which has been implemented yet. One is an explicit storage of otherwise misclassified patterns (exceptions). Another consists in manually selecting a different distance function scheme. There might also exist the possibility of automatically enhancing or reducing the representational power of a function scheme, similar to "growing" and "pruning" of neural networks.

6 Conclusion

When a learning problem can be formalized as a set of patterns, which have to be assigned to different classes, the goal of learning consists in finding the classification function. A distance measure on the space of input patterns together with a nearest-neighbor classification function provides such a function.

A way of learning a function consists in providing an adaptive function, technically a function with additional parameters, and setting those parameters to values with the help of a set of classified patterns (supervised learning of adaptive functions).

The main interest of our work lies in finding minimal fits for problems, not in specifying a single function scheme that is capable of handling most problems.

We may assume that cognitive functions of humans are ontogenetically adaptive only to a small extent. Individual, ontogenetic learning must be seen in contrast with evolutionary learning. Possibly, in evolutionary learning function shapes are determined and thus the limits to learnability are set.

Our interest lies in determining the complexity of classification functions that are used in cognitive tasks such as phoneme formation, formation of natural cognitive classes, grammatical categorization etc., as well as the complexity of learning such functions.

For this goal we have investigated several function schemes, as well as two different learning procedures. Function schemes differ in their representational power, examples are d_λ, a feature selection mechanism which has sublinear power, and d_{dnf}, a universal function scheme. Learning procedures differ in their applicability to function schemes: single pass parameter setting is extremely fast but limited in applicability, in contrast to iterative approximation of parameters, which is a slower process, but capable of solving more complex learning problems.

When a problem is not representable with a function scheme, it would be convenient to have a possibility of automatically analysing the training set to find a function scheme with sufficient representational power, for which a learning rule can be defined. So far we have only provided exception handling as a means of automatically extending a function scheme. Two other approaches are an extension of representational power of a function scheme by adding certain parameters, or to provide a method of translating knowledge concerning the function shapes of the distance function directly into the formulation of the distance function scheme. As knowledge concerning invariance under transformation or similarity of patterns needs to be formalized in an appropriate way, this constitutes a whole new area of research.

A Metric Properties of Distance Measure Schemes

The general metric properties are:

positive values: $d(x,y) \geq 0$

reflexivity: $d(x,x) = 0$

symmetry: $d(x,y) = d(x,y)$

triangle inequality: $d(x,y) + d(y,z) \geq d(x,z)$

A.1 d_λ

$$d_\lambda(x,y) = \sum_{i=1}^{n} \lambda_i * |x_i - y_i|$$

positive values: $d(x,y) \geq 0$

$$d_\lambda(x,y) = \sum_{i \in N^n} \lambda_i |x_i - y_i| \geq 0, \;\; \lambda_i \in R_0^+ \;\forall i \in N$$

since $\lambda_i \geq 0$ and $|\ldots| \geq 0$.

reflexivity: $d(x,x) = 0$

$$d_\lambda(x,x) = \sum_{i=1}^{n} \lambda_i * |x_i - x_i| = \sum_{i=1}^{n} \lambda_i * 0 = 0$$

symmetry: $d(x,y) = d(x,y)$

$$d_\lambda(x,y) = \sum_{i=1}^{n} \lambda_i * |x_i - y_i| = \sum_{i=1}^{n} \lambda_i * |y_i - x_i| = d_\lambda(y,x).$$

triangle inequality: $d(x,y) + d(y,z) \geq d(x,z)$

$$\sum_{i=1}^{n} \lambda_i * |x_i - y_i| + \sum_{i=1}^{n} \lambda_i * |y_i - z_i| \geq \sum_{i=1}^{n} \lambda_i * |x_i - z_i|.$$

$$a_i = \lambda_i * |x_i - y_i| + \lambda_i * |y_i - z_i|, \quad b_i = \lambda_i * |x_i - z_i|$$

We have to show that

$$a_i \geq b_i \; for \; any \; i \in N.$$

b_i can only have the values λ_i or 0.

The following cases can occur:

(a) $|x_i - y_i| = 1$

$|y_i - z_i| = 1, a_i = 2\lambda_i$. OK.

$|y_i - z_i| = 0, a_i = \lambda_i$. OK.

(b) $|x_i - y_i| = 0$

$|y_i - z_i| = 1, a_i = \lambda_i$.OK.

$|y_i - z_i| = 0, a_i = 0, b_i = 0$.OK.

A.2 d_c

$$d_c(x, y) = |\sum_{i=0}^{n} \gamma_i * \delta((\sum_{k=1}^{n} x_k) - i), 0) - \sum_{i=0}^{n} \gamma_i * \delta((\sum_{j=1}^{n} y_j) - i), 0)| = |f(x) - f(y)|$$

with $x := \langle x_1, \ldots, x_n \rangle$, $y := \langle y_1, \ldots, y_n \rangle$, $\gamma_i \in R_0^+, 1 \leq i \leq n$, and

$$\delta(x, 0) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

positive values: $d(x, y) \geq 0$

$$d_c(x, y) = |f(x) - f(y)| \geq 0$$

reflexivity: $d(x, x) = 0$

$$d_c(x, x) = |f(x) - f(x)| = 0$$

symmetry: $d(x, y) = d(x, y)$

$$d_c(x, y) = |f(x) - f(y)| = |f(y) - f(x)| = d_c(y, x)$$

triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$

It holds that

$$|f(x) - f(y)| + |f(y) + f(z)| \geq ||f(x) - f(y)| + |f(y) - f(z)||$$

Considering the different cases for the sign of $f(x) - f(y)$ and $f(y) - f(z)$, it is easy to see that for all cases

$$||f(x) - f(y)| + |f(y) - f(z)|| \geq |f(x) - f(z)|$$

The individual cases

$f(x) \leq f(y) \leq f(z)$:

$$\begin{aligned} ||f(x) - f(y)| + |f(y) - f(z)|| &= |f(y) - f(x) + f(z) - f(y)| \\ &= |f(z) - f(x)| \\ &= |f(x) - f(z)| \end{aligned}$$

$f(z) \leq f(y) \leq f(x)$:

$$||f(x)-f(y)|+|f(y)-f(z)|| = |f(x)-f(y)+f(z)-f(x)| = |f(x)-f(z)|$$

$f(z) \leq f(x) \leq f(y)$:

$$\begin{aligned} ||f(x) - f(y)| + |f(y) - f(z)|| &= |f(y) - f(x) + f(y) - f(z)| \\ &= |2f(y) - f(x) - f(z)| \\ &\geq |2f(z) - f(x) - f(z)| \\ &= |f(z) - f(x)| = |f(x) - f(z)| \end{aligned}$$

$f(y) \leq f(z) \leq f(x)$:

$$\begin{aligned} ||f(x) - f(y)| + |f(y) - f(z)|| &= |f(x) - f(y) + f(z) - f(y)| \\ &= | - 2f(y) + f(x) + f(z)| \\ &\geq | - 2f(z) + f(x) + f(z)| \\ &= |f(x) - f(z)| \end{aligned}$$

$f(y) \leq f(x) \leq f(z)$:

$$\begin{aligned} ||f(x) - f(y)| + |f(y) - f(z)|| &= |f(x) - f(y) + f(z) - f(y)| \\ &= | - 2f(y) + f(x) + f(z)| \\ &\geq | - 2f(z) + f(x) + f(z)| \\ &= |f(x) - f(z)| \end{aligned}$$

$f(x) \leq f(z) \leq f(y)$:

$$\begin{aligned} ||f(x) - f(y)| + |f(y) - f(z)|| &= |f(y) - f(x) + f(y) - f(z)| \\ &= |2f(y) - f(x) - f(z)| \\ &\geq |2f(z) - f(x) - f(z)| \\ &= |f(z) - f(x)| = |f(x) - f(z)| \end{aligned}$$

A.3 d_l

$$d_l(a, b) = 1 - sgn((\sum_{i=1}^{n} \alpha_i a_i) - \theta) \times ((\sum_{i=1}^{n} \alpha_i b_i) - \theta)$$

with $x := \langle x_1, \ldots, x_n \rangle$, $y := \langle y_1, \ldots, y_n \rangle$, $\alpha_i \in R_0^+, 1 \leq i \leq n$, and $\theta \in R$
We abbreviate: $f(x) := (\sum_{i=1}^{n} \alpha_i x_i) - \theta)$

positive values: $d(x,y) \geq 0$ $d_l(x,y) \geq 0$ since $sgn(x) \in \{-1, 0, 1\}$.

reflexivity: $d(x,x) = 0$

$$d_l(x,x) = 1 - sgn(f(x)^2) = 0$$

except for $\sum_{i=1}^{n} \alpha_i x_i = \theta$. In this case, the pattern x is directly located on the discriminant.

symmetry: $d(x,y) = d(x,y)$

$$d_l(x,y) = 1 - sgn(f(x)f(y)) = 1 - sgn(f(y)f(x)) = d_l(y,x)$$

triangle inequality: $d(x,y) + d(y,z) \geq d(x,z)$

$$1 - sgn(f(x)f(y)) + 1 - sgn(f(y)f(z)) \stackrel{!}{\geq} 1 + sgn(f(x)f(z))$$

Since $sgn(x) \in \{-1, 0, 1\}$ a simple check of all occurring values shows this property.

$f(x)$	$f(y)$	$f(z)$	LHS	RHS
≥ 0	≥ 0	≥ 0	0	0
≥ 0	≥ 0	≤ 0	2	0
≥ 0	≤ 0	≥ 0	4	0
≥ 0	≤ 0	≤ 0	2	2
≤ 0	≥ 0	≥ 0	2	2
≤ 0	≥ 0	≤ 0	4	2
≤ 0	≤ 0	≥ 0	2	2
≤ 0	≤ 0	≤ 0	0	0
*	*	$= 0$	≥ 1	0
	$= 0$	*	≥ 2	≤ 2
$= 0$	*	*	≥ 1	1

References

[BH88] E.B. Baum and D. Haussler. What size net gives valid generalization? In *Advances in Neural Information Processing 1*, pages 81–90, 1988.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, 1973.

[Hor93] Kurt Hornik. Some new results on neural network approximation. *Neural Networks 6*, pages 1069–1072, 1993.

[HSW89] Hornik, Stinchcombe, and White. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, pages 359–366, 1989.

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons.* MIT Press, 1969.

[Pao89] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley, 1989.

[RHW86] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In J. McClelland and D. Rumelhart, editors, *Parallel Distributed Processing*, volume 1. MIT Press, 1986.

[Sch94] G. Scheler. Feature selection with exception handling— an example from phonology. In *Proceedings of EMCSR*, (to appear), 1994.

[Whi93] Halbert White. *Artificial Neural Networks: Approximation and Learning Theory.* Blackwell, 1993.

[Wol90] Stephen Wolfram. *Mathematica. User's Manual.* MIT Press, 1990.