THE UNIVERSITY OF CALGARY

PERFORMANCE OF SOME OLD AND NEW ADAPTIVE DECISION STRATEGIES IN PRACTICAL MACHINES

by

IAN H. WITTEN

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS,

STATISTICS & COMPUTING SCIENCE

CALGARY, ALBERTA

AUGUST, 1970

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Performance of some old and new adaptive decision strategies in practical machines" submitted by Ian H. Witten in partial fulfillment of the requirements for the degree of Master of Science.

.....

Supervisor, D.R.Hill
Mathematics, Statistics,
& Computing Science.

.....

W.C.Chan
Electrical Engineering.

.....

C.M.Fenyvesi
Mathematics, Statistics,
& Computing Science.

John Slater
.....

J.Slater
Mathematics, Statistics,
& Computing Science.

25th August 1970
.....
(date)

ABSTRACT

This thesis is concerned with the problem of decision-making in the context of automatic pattern recognition. Almost all decision strategies which have been employed in practical machines are variants of two simple schemes: the perceptron decision strategy and the maximum likelihood decision strategy. The main part of this thesis is devoted to a critical study of these strategies.

Although the perceptron decision strategy behaves extremely well under favourable conditions, it tends to be misled by patterns which are unavoidably misclassified when noise is present. This phenomenon is thoroughly investigated (Chapter 3), and a variant of the strategy, which has a better chance of performing well in noisy conditions, is defined. This variant proves itself in a limited series of experiments (Chapter 7).

The maximum likelihood decision strategy is optimal (in a precise sense) under all conditions, but unfortunately its implementation is impractical unless restrictive assumptions are made (Chapter 4). Implications of these assumptions are considered in some detail (Chapter 6). Adaptation of the maximum likelihood strategy can only be achieved by estimation of probabilities, and some standard procedures for this are discussed, together with problems arising from the existence of storage limitations (Chapter 5).

Part of the research reported here was concerned with the problem of finding a basis for theoretical investigation of the two

decision strategies mentioned above with a view to combining their virtues. This is considered in Chapter 8.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, David Hill, not only for his ideas and opinions which prompted the research reported here, but also for a splendid introduction to the fascinating domain of Artificial Intelligence.

I am also indebted to Mrs C.M.Fenyvesi for her help in proofreading this thesis.

Financial support during the preparation of this thesis was provided by the Commonwealth Scholarship and Fellowship Plan.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION, MOTIVATION, AND GOALS

This thesis is concerned with the problem of decision-making in the context of automatic pattern recognition. It is generally agreed that pattern recognition is a fundamental, some say the central, problem of artificial intelligence:

> The problem of sorting events and situations into useful categories arises in so many ways that it is tempting to regard it as the central problem of artificial intelligence. (Minsky, 1958.)

It is evident that all problems can be re-formulated in terms of sets, and that in this way, problems which are at first sight entirely unconnected with pattern recognition can be reduced to what is superficially a problem of classification into sets. However, it is a fact that very many problems which occur naturally in fields like artificial intelligence and process control are usefully treated as pattern recognition problems.

The bewildering variety of sense-data available to a machine equipped with receptors or devices which make it sensitive to its environment highlights the need for pattern recognition. In the context of problem solving, a resourceful machine must classify problem situations into categories associated with the domains of effectiveness of the machine's different methods, in order not to try all possibilities. This process is particularly transparent in the "Logic Theory Machine" of Newell *et al.* (1957).

In order to indicate further the usefulness of a pattern recognition approach to problems which, at first sight, seem to be unconnected with pattern recognition, I shall briefly consider Samuel's studies in automatic checker playing (1959, 1967). The essence of Samuel's strategy is to play by looking ahead a few moves and evaluating the resulting board positions, much as a human player might do. The best move is chosen by "minimaxing" the move chains considered, using the static board evaluation to determine the "goodness" of each move at each stage. It is the evaluation procedure which can be viewed as a pattern recognition process; although it is not strictly a classification procedure, Samuel both used techniques and encountered problems which occur in conventional pattern classification work. To evaluate the board positions, he originally used a linear polynomial whose terms represented features or attributes of the board position, with coefficients which indicated the importance or weight of that feature. Various methods were used for selecting features out of a large man-generated list, and for adapting the weights according to the machine's experience. One of the major snags encountered was the limitation inherent in the use of a linear scoring polynomial--this is a fundamental difficulty in pattern recognition work--and a number of different proposals were considered for overcoming this.

Pattern recognition is concerned with the reduction or *structuring* of a complex environment into a relevant and manipulable form in order to facilitate goal achievement. Conventional computers are not able to organize or classify information in any very subtle or generally applicable way--they perform only highly specialized

operations on carefully prepared inputs. Only through classification into categories can we hope to introduce "general" or "informal" problem-solving methods. Pattern recognition involves both pattern classification and pattern discovery, although the latter has been rather neglected in the literature. STeLLA (see Chapter 8; Andreae, 1964, 1969; and Gaines & Andreae, 1966) is a general purpose learning machine which discovers patterns in a manner highly relevant to goal achievement. Despite their obvious differences, pattern classification and pattern discovery require the use of similar techniques; unfortunately machines which are intended to discover patterns are more liable to instability and oscillation because of the difficulty of reinforcing decisions.

The conventional formulation of the pattern classification problem is this: sets of data are supplied to the classifier, either as raw sense-data or in some preprocessed form. These sets of data are divided by some consensus of opinion (on the basis of usefulness of treating them as the same) into pattern classes, and the pattern classes are also supplied to the machine. After this training period the machine is given data sets, possibly ones which it has not "seen" before, and is required to classify these correctly with respect to the consensus of opinion. Interesting problems occur when the rules governing the correct classifications are not explicitly known (hence the similarity between pattern classification and pattern discovery). The purpose of the training period is to give the recognizer a chance to form its own rules, or to utilize pre-programmed rules to maximum effect. It is for this reason that such decision strategies are called *adaptive*. This will be taken up again later. The pattern

classification problem has three natural subproblems: data input, description (structuring of the input data), and decision.

The input phase was largely circumvented in the early pattern recognizers by hand digitization of data. These recognizers were usually concerned with optical patterns which were projected on to a "retina" consisting of a rectangular matrix of squares. A square containing any part of the pattern was considered to be filled. This process was not designed to eliminate all stray noise; in fact these machines sometimes have a preliminary clean-up stage to take care of noise. Selfridge & Neisser's "Pandemonium" (1963) exhibits this. Nevertheless, it seems fairly safe to say that noise will appear to a much greater extent in real situations. A large number of modern pattern classification machines take their input directly from the real world via microphones, optical scanners, and the like. This involves technical problems in the area of man-machine communication (see for example McCarthy *et al.*, 1968).

The description or feature extraction phase has the task of extracting some of the relevant and meaningful features from the vast mass of input data. In many cases it must first be determined whether or not any meaningful data is actually present, and if so, where (in time, space, or both) the pattern to be recognized is situated. Recognition of connected speech or handwriting exemplifies difficulties which can occur in deciding where a pattern starts or ends. The conventional approach to this is tentatively to segment and/or normalize the input (Gold, 1959), although other approaches exist (Hill, 1969). Feature extraction can take place according to pre-programmed features (Selfridge, 1958), or according to machine-

generated masks (Uhr & Vossler, 1963). Early character recognition machines usually computed individual features independently from small sub-sections of the retina. While this method of feature extraction is quite adequate for many purposes, it does reflect the use of a number of computations performed independently of each other (see Selfridge, 1958, for a discussion of this). This is not the most efficient--though it may be the quickest--way to use computations, and it seems that it is not adequate for general pattern recognition. Minsky & Papert (1969) devote almost a complete book to a consideration of limitations of machines which use features obtained in this way--this is discussed briefly at the end of this chapter. Several ways of getting round this difficulty exist. One could use primitive features for recognition of elementary parts out of which patterns may be built (small sections of lines, vertices, etc.) and knit these together using list processing techniques (this is discussed by Minsky, 1961; Guzman, 1968, provides a practical example of such a machine). A rather similar way of overcoming the limitations arising from the use of elementary features is to compound these features with each other before (or in conjunction with) the decision phase. An example is given by Hill (1969). Alternatively, multilayer decision networks can be used to give complex decisions by "cascading" simple ones. Unfortunately these are beyond the scope of this thesis: the subject is touched on briefly in Chapter 9.

The input to the decision phase consists of a vector of measurements, which may or may not be binary, representing the features present in the input pattern. This vector is called the *query vector* or *feature vector* throughout this thesis. During the training period,

the correct classification of the pattern currently being considered must be provided to the decision phase for reinforcement purposes. The output of the decision phase is a number representing the pattern class thought to contain the input pattern. The pattern classes may include a *reject class* ("information not sufficient for a firm decision") and/or a *noise only class* ("no pattern present"). It is important to realize that two identical query vectors may arise from two patterns in different classes, owing to either noise perturbation or insufficient resolution at the feature extraction or input stages. Hence the "correct" (according to the consensus of opinion) class for any given pattern is not necessarily a function merely of the query vector obtained from that pattern.

This thesis is concerned with *adaptive* decision strategies. These are used in cases where

1) the rules governing the classification are not known: adaptive techniques can be used to automate data acquisition,

2) the situation may be changing slowly so that the information is context sensitive,

3) a versatile machine which can be used for different situations is required.

These cases often overlap in practice. Speech recognition is a good example of a situation where the rules for classification are not known well enough to embody them in an automatic recognition system. Slowly changing situations are illustrated by the problem of recognizing hand-sent Morse code. When a long message sent by a single operator is analysed, it frequently turns out that some dots are longer than some dashes, and so an efficient recognizer must use

some sort of contextual information (Selfridge & Neisser, 1963). The search for a versatile machine is prompted by the fact that it may be cheaper to produce a general purpose pattern recognition machine and train it for the task in hand than to build special purpose machines.

I would like to stress that I am not suggesting that adaptation is a panacea. Many pattern recognition problems can be solved efficiently by techniques which are not adaptive. However, in cases where classification is required but the data on which classifications are to be based are not sufficiently structured, adaptation is certainly a very good tonic until something else turns up (*sic*).

The feature extraction phase is the major point of attention in pattern recognition today--and quite rightly so. Although automatic recognition of hand-printed characters has been investigated since around 1958, even now there appears to be no consensus of opinion on what features are among the best. Automatic speech recognition is a field where feature extraction is of paramount importance because of the vast mass of data available in speech waveforms.

It is true that if ideal features can be found, the decision stage is trivial. Features could consist of a single number representing the class to which the pattern belongs, or, less trivially, a bit-pattern which requires only exact matching with some stored patterns to ascertain the class. One of the more successful speech recognizers currently operating, that of Bobrow and his colleagues (1968, 1969), uses a particularly simple decision scheme--a *voting* scheme.

> It has been experimentally determined that the voting scheme works as well or better than a number of other measures that make use of the same information. (Bobrow & Klatt, 1968.)

Unfortunately little mention is made of which other decision schemes were tried, or of how performances compared. The fact that such a simple scheme proved so effective indicates the suitability of the features used. Bobrow's system was, however, designed for use with a single speaker only. If one wishes to recognize many speakers with large vocabularies, then it seems likely that a more powerful decision strategy will be required.

Almost all decision strategies which have been employed in practical machines ("machines" is used in this thesis in an all-embracing sense which covers programmed computers) are variants of two simple schemes: the perceptron decision strategy and the maximum likelihood decision strategy. While I originally intended to use these as a jumping-off point for my research, preliminary reading revealed that nowhere in the literature is there a critical review in depth of the performances of these basic adaptive decision strategies, and worse still, it is extremely rare to find any treatment of the two together. This appears to stem from the fact that each strategy has its proponents who are unwilling to acknowledge the merits of other strategies. In addition to this, difficulties which occur in connection with the decision strategies are often ignored altogether, or brushed aside with an airy remark (the independence assumption is a good example of this; see Chapter 6).

One of the main difficulties which crops up while investigating adaptive decision strategies is the necessary compromise between the optimal and the practical. A decision strategy which is

optimal (in a precise sense) for any situation in fact exists; this is the maximum likelihood decision strategy. However, implementation of this strategy in a practical machine requires, for any real situation, restrictive assumptions to be made, and these usually impair the optimality of the decision strategy. Conventional mathematics does not seem to be well-tailored to situations in which practical implementation is an important consideration, and on the other hand, experimentation is a time-consuming process which cannot hope to be exhaustive. It is this, I feel, that accounts for the lack of satisfactory treatments of adaptive decision strategies in the literature.

The foregoing remarks are a necessary prerequisite for a sympathetic understanding of the goals of the research reported here. My major objective is to develop a (necessarily) heuristic insight into the characteristics of the perceptron and maximum likelihood decision strategies, always bearing in mind that these decision strategies are intended for use in practical machines. A secondary objective is to provide a basis for theoretical investigation of these strategies with a view to combining their virtues.

As mentioned above, experimental investigation, while worthwhile and necessary in the context of feature extraction for particular problems, is not well suited for probing into the general characteristics of decision strategies. Consequently the small amount of experimental work I have done (reported in Chapter 7) is intended only to illustrate some of the points made in the course of theoretical investigation. Mathematics is my major tool, but I try to remain conscious of the danger of neglecting practical considerations;

the result is that mathematics is used to probe special cases in order to provide a basis for heuristic generalization. Here I run the risk of being accused of "sloppiness"; I hope that the following pages provide evidence to the contrary. I also hope that non-mathematicians are able to understand the arguments presented in this thesis, for it is intended to help designers of practical pattern recognition machines to select a suitable decision process for their particular machine. To this end, proofs and occasionally precise statements of theorems are relegated to Appendix A, unless this would seriously disrupt the flow of thought through the main arguments. No deep mathematical results are used, and all the mathematics here is of an *ad hoc* character.

Chapter 2 is an introduction to the subject of adaptive decision processes, on a more technical level than the present chapter. In Chapters 3 and 4, the two basic decision strategies are considered in turn. Further topics relevant to maximum likelihood decisions are dealt with in the next two chapters. These contain rather more mathematics than I would like, but I feel that the mathematical arguments are the very essence of these chapters, and so they are left in the main text. Chapter 7 describes some experiments which were undertaken in order to illustrate points made earlier. This completes the investigation of the perceptron and maximum likelihood decision strategies. My secondary objective, that of providing a basis for theoretical investigation of the afore-mentioned strategies with a view to combining their virtues, is considered in Chapter 8. The last chapter is devoted to some concluding remarks. Appendix A, as mentioned above, contains proofs of some theorems quoted in the text,

and Appendix B is a glossary of special symbols, terms, and abbreviations used.

The literature on artificial intelligence is somewhat scattered among various journals and conference proceedings. A fairly comprehensive bibliography of papers appears at the end of this thesis; all the papers therein are referenced at relevant points in the text. There are few good books on pattern recognition, but I will indicate those which I have found useful. For background reading in artificial intelligence, both Feigenbaum & Feldman (1963) and Uhr (1966) provide a good, if fairly dated, introduction to the field. The latter is very much neurophysiologically oriented. I should say here that my research is directed away from the fields of psychology and neurophysiology--my aim, in J.H.Andreae's words, is "to build a useful machine"--but some acquaintance with these subjects is useful because, after all, the brain is by far the most versatile and competent pattern recognizer in existence, and studies of the brain may provide hints relevant to automatic pattern recognition. Further background material in neurophysiology and associated topics can be found in McCulloch (1965). Relevant philosophical problems are discussed by Koestler (1964) and Craik (1952).

So much for general material. Rosenblatt (1962) considers perceptron-like machines in some detail. Sebestyen (1962) and Nilsson (1965) both treat much the same topics as I do, but not, I feel, in a manner so relevant to practical machines. Nagy (1967) gives a comprehensive "state of the art" report on automatic pattern recognition, while a collection of papers, many of them concerned with practical work, is to be found in Tou & Wilcox (1964).

Last year one of the foremost workers in the field of artificial intelligence, Marvin Minsky, published a book in collaboration with Seymour Papert (Minsky & Papert, 1969) which has considerable relevance to this thesis.

Minsky is primarily concerned with inherent limitations of linear decision strategies (see Chapter 2) if features are computed independently from small subsections of the retina. While his discussion of this topic is not directly connected with my work, since I make no assumptions about what kinds of features are used, it should be realized that experimental evidence indicates that both the frog (Lettvin *et al.*, 1959) and the cat (Hubel & Wiesel, 1962) employ "features", in some sense, which are computed independently from small subsections of the retina. These elementary features may be combined into more complex ones in a hierarchical manner. Indeed, although introspective psychology is out of fashion these days, it does seem that examining a figure for a topological property like connectedness (Minsky discusses this property extensively) involves an essentially serial[1] operation, and Minsky's suggestion that inability to recognize such serial properties is a serious limitation of perceptron-like machines is perhaps a little trite. On the other hand, Minsky's main concern is to debunk the idea, rather prevalent a few years ago among some of the perceptron's proponents, that the perceptron is a universally applicable learning machine and a panacea for all the problems of artificial intelligence, and he certainly accomplishes this goal.

---

[1] In the sense that independent computations do not provide adequate features.

Minsky devotes a few pages to a discussion of the perceptron and maximum likelihood decision strategies, but admits that he can offer no general theory of learning (learning and adaptation are used synonomously in this thesis). He is interested in genuine foolproof mathematics, rather than in the heuristic sort of mathematics used here, and I attribute his lack of a theory of learning to the difficulties mentioned earlier, that conventional mathematics is applicable to ideal rather than real situations.

Chapter 2

PATTERN RECOGNITION AND LINEAR DECISION SCHEMES

An important point which should be kept in mind when considering adaptive decision strategies is that the feature extraction process is inextricably bound up with the decision phase. J.H.Andreae suggests that one should imagine a continuum embracing the feature extraction and decision procedures, the distinction between the two stages being made on a rate of adaptation/time scale basis. He considers any process in a pattern classifier which adapts slowly (relative to the rate of adaptation of the other processes) to be part of the feature extraction phase. Thus the decision phase has the task of attempting short term optimization, while long term optimisation is achieved by adapting the features. It is, however, convenient to separate the two phases because feature extraction is very dependent on the particular kind of problem being considered (and also on the kind of implementation intended, whether by electronic computer or special-purpose hardware), while the decision phase is dependent on the characteristics of the features used rather than on the kind of recognition problem being considered.

It is worthwhile looking at some examples of general characteristics of features. One such example, mentioned in the last chapter, concerns cases where the features are "ideal" and only exact matching with stored templates, one for each pattern class, is required. Cases where this works well are rarely encountered in practice. Features may be non-redundant, in which case if they

exhibit perturbations due to noise no decision strategy can be expected to perform well in an absolute sense, or they may be highly redundant, and if so, it is reasonable to expect a decision strategy to perform well in noisy conditions. If the features are statistically independent with respect to the pattern classes then one can use a decision strategy which is both optimal and implementable in practical machines (see Chapters 4 and 6), while if dependencies exist but their form is known, special *ad hoc* techniques can be used.

In referring to the interface between the feature extraction and decision phases, I often use the term *environment*. This denotes the set of possible query vectors, with their frequencies and classes. Two extreme types of environment may be distinguished, although in practical situations a combination of both invariably occurs:

1) environments where the pattern classes are distinct, so that the query vectors arising from patterns in each class occupy separated portions of hyperspace (*feature space* or *query space*);

2) environments where each pattern class has a single basic query vector but random noise perturbations exist, such that the noise acts on each feature independently. Here the pattern classes may be thought of as unimodal distributions of query vectors in hyperspace, probably overlapping to a considerable extent.

The first kind of environment corresponds to situations in which the perceptron strategy works well, while the second kind corresponds to situations where the maximum likelihood strategy works well.

Environments are usually considered in this thesis to comprise a core of noiseless query vectors, each with its associated frequency of occurrence, with noise superimposed on these. Naturally the

noiseless pattern classes should be non-overlapping if the pattern recognizer is to perform at all well. As far as the decision phase is concerned, the pattern recognition process can be thought of as selecting points at random from the noiseless pattern classes in feature space (inter-dependencies in the presentation sequence are not envisaged), choosing each point with its associated frequency. Features are computed merely by measuring the feature space co-ordinates of the selected point. The features are then corrupted by noise in the required probabilistic manner, and the corrupted query vector is presented to the decision phase. This way of looking at environments is found to be helpful, especially for perceptron-like decisions.

The notion of *convergence* of an adaptive decision machine is used frequently in this thesis. In noiseless conditions, we say that an adaptive decision machine has converged if it correctly discriminates between the pattern classes. After convergence, the performance of a decision machine may improve or deteriorate if the training period is continued. Fortunately this is not true of perceptron-like machines, and the term "convergence" is used mainly in connection with these. Perceptrons do not adapt themselves after convergence has been reached, and so their *convergence time* (mean number of patterns presented before convergence) is a good indication of the required length of training period. Further ramifications of the idea of convergence are introduced in the following when necessary.

Determining if a machine has converged is not an easy task in real situations. This is basically because the set of training patterns is invariably a rather small subset of the total number of

patterns which the machine is expected to classify (this is discussed extensively by Nagy, 1967). One must resort to statistical sampling to determine if convergence has been reached, but because of the very real danger that the training set is not sufficiently representative this is not usually reliable. Since the only indication of the length of training period required for certain types of decision machine is convergence time, it is difficult to determine when to stop training.

It has been suggested that "tracking" or "selective bootstrapping" techniques be employed in order to combat possible inadequacy of the training set (Nagy, 1967). A selective bootstrapping system employs a teacher to supervise the learning process. When, in the judgement of the teacher, the performance over long chains of patterns is acceptable, the machine is left to reinforce its own decisions (see for example Widrow & Smith, 1963). Instability rears its ugly head here; for most pattern classes are defined by convention alone (the consensus of opinion of Chapter 1) and unless the machine has been taught the bare bones, at the very least, of the convention it may, when left to itself, begin to reinforce incorrect decisions and in doing so destroy its whole pre-taught body of knowledge. A few noisy query vectors could start such a disastrous landslide. I have not investigated such selective bootstrapping systems.

We now introduce some notation and look at a general formulation of the decision problem. Query vectors are denoted throughout this thesis by the symbol Φ. The i'th component, ϕ_i, represents the extent to which the i'th feature is present. The range of ϕ_i may be continuous or discrete, binary or many-valued; unless otherwise specified. Pattern classes are denoted by $F^{(j)}$, and the

output of the decision phase is a number representing the index of the pattern class. (Ranges of indices are given in this thesis only when absolutely necessary to avoid confusion--the ranges of summations etc. are generally obvious. This simplifies writing and reading considerably.) Although pains were taken to point out in the last chapter that two identical query vectors may arise from two patterns in different classes, the notation

$$\Phi \varepsilon F^{(i)}$$

is used for

"The pattern class currently giving rise to Φ belongs to the class $F^{(i)}$",

since this should cause no confusion.

Classification is effected by *discriminatory functions* $f^{(i)}$, one associated with each pattern class, such that (ideally)

$$f^{(i)}(\Phi) > f^{(j)}(\Phi) \quad \text{for all } j \neq i \quad \text{if and only if} \quad \Phi \varepsilon F^{(i)}.$$

Note that only non-randomized decision strategies are included in this formulation. These correspond to "pure" rather than "mixed" strategies in game theory. Chow (1957) showed that optimum strategies in pattern classification are pure, and mixed strategies are not considered here. The surfaces in feature space given by

$$f^{(i)}(\Phi) = f^{(j)}(\Phi) \qquad (i \neq j)$$

are called *discriminatory surfaces*. It is assumed for convenience that points lying on discriminatory surfaces are classified according to some convention.

The decision phase of a pattern classifier is just the implementation of the discriminatory functions. An adaptive decision strategy must provide a mechanism for learning the correct

discriminatory functions. This is practically impossible unless a special form is assumed for the discriminatory functions, and adaptive decision techniques are usually discussed only for linear discriminatory functions. (Some strategies considered in this thesis are not in fact linear decisions, but similarly restrictive assumptions are always made about the form of the discriminatory functions.) Because of the importance of linear decisions, both historically and practically, the remainder of this chapter is devoted to their consideration.

Let us suppose that the query vectors are n-dimensional, and we are interested in dividing up feature space using linear discriminatory functions:

$$f^{(j)}(\Phi) = w_1^{(j)}.\phi_1 + w_2^{(j)}.\phi_2 + \ldots + w_n^{(j)}.\phi_n + w_{n+1}^{(j)} .$$

The coefficients w are generally called *weights*, and these are the only elements of the decision phase which are subject to adaptation. These weights may be positive or negative, and it is assumed in this thesis that they may take any values. In order to simplify the notation, we define the *augmented query vector* Φ', dependent on Φ and of dimension n+1, whose components are

$$\phi'_i = \phi_i \quad \text{for} \quad 1 \leq i \leq n;$$

$$\phi'_{n+1} = 1.$$

Let the *weight vector for the i'th class* be defined as

$$W^{(j)} = (w_1^{(j)} , w_2^{(j)} , \ldots , w_{n+1}^{(j)}).$$

Then the discriminatory functions are

$$f^{(j)}(\Phi) = W^{(j)}.\Phi',$$

and the query vector Φ is assigned to the pattern class $F^{(j)}$ for which

$$W^{(j)}.\Phi'$$

is largest. The introduction of the augmented query vector enables us to write a linear form in Φ as a homogeneous linear form in Φ'.

This method of linear classification is equivalent to separating each pair of pattern classes with a hyperplane in feature space. Thus if there are m pattern classes, m(m-1)/2 hyperplanes exist, defined by

$$W^{(i)}.\Phi' = W^{(j)}.\Phi' \quad (i \neq j).$$

The pattern classes are said to be *linearly separable* if there exist weight vectors satisfying the following:

$$W^{(i)}.\Phi' > W^{(j)}.\Phi' \text{ for all } j \neq i \text{ if and only if } \Phi \varepsilon F^{(i)}.$$

A necessary and sufficient condition for sets of points in hyperspace to be linearly separable is that the intersections of the convex hulls of the sets, taken in pairs, are empty (Papert, 1960). The convex hull of a set can be visualized by throwing a cloth round the set and drawing it tight. If the set is finite, its convex hull is a convex polyhedron with points of the set as vertices.

Other methods of using hyperplanes to separate sets exist, as was pointed out by Griffin *et al.* (1963). For m classes, if p is an integer such that

$$2^p \geq m > 2^{p-1},$$

it is in principle possible to use just p hyperplanes to separate the classes, provided that the regions in which the various classes are concentrated are well spaced out in hyperspace. Alternatively one could attempt to use a hyperplane to separate one class from all the other classes taken together. The method described above is at least as powerful as the latter method, and it seems highly plausible that it is more powerful than the first alternative. To my knowledge

neither of these alternative methods has been used in practical machines.

Minsky & Papert (1969) generalized the above formulation by allowing situations where each pattern class has many weight vectors, the vector $W^{(i)}$ being associated with the class $F^{(j(i))}$. j here is a mapping,

$$j: \{1,2,\ldots,m'\} \rightarrow \{1,2,\ldots,m\} \quad (m' \geq m)$$

which is *onto*, that is to say,

for all k $(1 \leq k \leq m)$ there exists k' with $j(k') = k$.

This permits coverage of cases where each F-class is localized into many relatively isolated regions by allowing a weight vector for each cluster. This is called a *piecewise linear* decision scheme, and is a simple extension of my formulation which is not explicitly catered for here but to which all results and techniques given here are applicable.

One of the weight vectors is redundant. If we define

$V^{(j)} = W^{(j)} - W^{(i)}$ for all j and some particular i,

and assign Φ to the class $F^{(j)}$ for which $V^{(j)}.\Phi'$ is largest, we evidently obtain the same classification as before, with $V^{(i)}=0$. This is particularly useful in the rather special case where there are only two pattern classes, F^+ and F^-. This formulation is used frequently in the following chapters with the single weight vector being denoted by W instead of V. If the two pattern classes are linearly separable, then there exists a vector W* with

$W^*.\Phi' > 0$ for all $\Phi \varepsilon F^+$;

$W^*.\Phi' < 0$ for all $\Phi \varepsilon F^-$.

This *discriminating weight vector* is always denoted by W* in the two class case.

It will frequently be necessary to assume that there exists $\delta>0$ with

$$W^*.\Phi' > \delta \quad \text{for all } \Phi \varepsilon F^+;$$

$$W^*.\Phi' < -\delta \quad \text{for all } \Phi \varepsilon F^-.$$

This does not follow from the assumption that F^+ and F^- are linearly separable, as can be seen by considering a one-dimensional feature space with the pattern classes

$$F^+ = \{ (1) , (2^{-1}) , (2^{-2}) , (2^{-3}) , \ldots \},$$

$$F^- = \{ (-1) , (-2^{-1}) , (-2^{-2}) , (-2^{-3}) , \ldots \}.$$

If the pattern classes are finite, however, or if the components of the query vectors are discrete (as will always be the case if a digital computer is used for implementation), then linear separability of F^+ and F^- does imply the existence of a $\delta>0$ satisfying the above. It will be assumed for convenience in this thesis that linear separability does indeed imply the existence of such a δ whenever necessary.

Several common decision methods can be realized using linear discriminatory functions. One of these is the *minimum-distance* decision strategy, where a point $P^{(i)}$ is chosen for each pattern class $F^{(i)}$, and classification is effected by choosing i such that

$$|P^{(i)} - \Phi|^2 < |P^{(j)} - \Phi|^2 \quad \text{for all } j \neq i.$$

This is equivalent to choosing i such that

$$P^{(i)}.\Phi - |P^{(i)}|^2/2 > P^{(j)}.\Phi - |P^{(j)}|^2/2 \quad \text{for all } j \neq i,$$

which is clearly a linear decision with weights

$$W^{(i)} = (P_1^{(i)} , P_2^{(i)} , \ldots , P_n^{(i)} , -|P^{(i)}|^2/2).$$

Some other decision methods which are in fact linear are discussed in Chapter 4.

The major objection to linear decision strategies is that some environments are not linearly separable. This again is dependent on the feature extraction phase, but some facts about linear separability are worth noting. It is manifest that the more features there are, the more likely it is that the patterns are linearly separable. For adding a feature cannot destroy the property of linear separability, but it may separate linearly an environment which was not originally linearly separable.

When given features which are integral but not binary, it is common to encode them into binary notation before the decision phase, and use the new binary features in the decision mechanism. This makes some decisions easier to implement. It is easy to see that a positional binary encoding actually enhances the possibility of linear separability. Let Φ be the old feature vector, and

$$\Phi^* = (\xi_{11} , \xi_{12} , \ldots , \xi_{1k} , \xi_{21} , \ldots , \xi_{2k} , \ldots , \xi_{nk})$$

be the new binary feature vector, where

$$\xi_{i1}\xi_{i2} \cdots \xi_{ik}$$

is the k-digit positional binary encoding of ϕ_i.

Then if $f^{(j)}(\Phi)$ is linear in ϕ_i,

$$f^{(j)}(\Phi) = w_1^{(j)}.\phi_1 + w_2^{(j)}.\phi_2 + \ldots + w_n^{(j)}.\phi_n + w_{n+1}^{(j)} ;$$

$f^{(j)}(\Phi^*)$ is clearly linear in ξ_{im}:

$$f^{(j)}(\Phi^*) = w_1^{(j)}.2^{k-1}.\xi_{11} + w_1^{(j)}.2^{k-2}.\xi_{12} + \ldots + w_1^{(j)}.\xi_{1k} + w_2^{(j)}.2^{k-1}.\xi_{21} + \ldots + w_n^{(j)}.\xi_{nk} + w_{n+1}^{(j)} .$$

However, there are many functions linear in ξ_{im} which are not linear in ϕ_i. For example, if

$f^+(\Phi^*) = \xi_{11}$;

$f^-(\Phi^*) = \xi_{22} + 0.1$;

then the discrimination is as shown in Figure 2.1, and is certainly

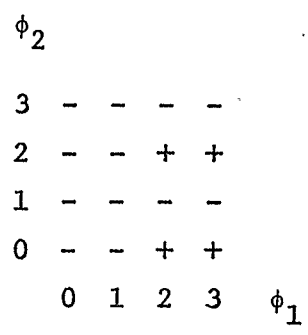

Figure 2.1

A classification which is linear if the features are binary-encoded.

not linear if the non-binary features are used.

One great advantage of linear decisions is that compound features can be added to make a non-linear decision linear. If one wishes to implement an adaptive decision then one must in practice assume some form for the discriminatory functions, and having done this, the decision may be implemented in a linear manner. For example, suppose Φ has two components, and the following forms are assumed for the discriminatory functions:

$$f^+(\Phi) = a_1(\phi_1)^2 + a_2(\phi_2)^2 + a_3\phi_1 + a_4\phi_2 + a_5;$$

$$f^-(\Phi) = b_1(\phi_1)^2 + b_2\phi_1\phi_2 + b_3;$$

where the a's and b's are constants. Then if we define a new query vector which can be computed from the old:

$$\Phi^* = \Phi^*(\Phi) = ((\phi_1)^2 , \phi_1\phi_2 , (\phi_2)^2 , \phi_1 , \phi_2),$$

the discriminatory functions are linear in Φ^*. Given any query vector

Φ, we need only compute $\Phi^*(\Phi)$ and use this as a new query vector, discarding the old one.

In conclusion, let us summarize the properties of linear decisions which make them worth considering.

1) They may be implemented easily, either by special-purpose hardware or by digital computer (see for example Highleyman, 1961 and 1962).

2) New or compound features can be added to make a non-linear decision linear. Thus if the form of the optimum decision is known, the decision can be implemented and adapted using linear techniques.

3) Piecewise linear decisions can be used to give a more general classification with the same basic mechanism.

4) Many common types of decision are in fact linear (for example, the minimum-distance strategy--further examples are provided in Chapter 4).

Although the properties of linear decisions have been discussed, no mention has been made of how to achieve adaptation to best effect. One of the most common adaptation methods is discussed in the next chapter. Other ways of implementing linear decisions can be found in Chapter 4, where the decisions, although non-linear in general, become linear in an important special case.

Before closing this chapter, some further notations should be introduced. It is frequently necessary in this thesis to use the Boolean value of an expression. This is done by enclosing it in the *corners* $\ulcorner$ and $\urcorner$. Thus

$$y = \ulcorner x=5 \urcorner$$

has the value

1 if $x=5$,

0 otherwise.

Another term which is often used is the *size* or *length* of vectors. This refers to the modulus of the vector, as in vector algebra. The symbol Φ^* is sometimes used to denote a special query vector. This has no connection with the notation W^* for a correctly discriminating weight vector.

Chapter 3

THE PERCEPTRON DECISION STRATEGY

Perceptrons have excited considerable attention since they were first introduced in 1957 by Rosenblatt (1957, 1962). Over the years large numbers of mutations and variations have appeared, and consequently the term "perceptron" has acquired a number of different and ill-defined connotations. My interest lies solely in the power of the simple adaptation, learning, or reinforcement procedures which are employed by perceptrons, to learn to classify abstract vectors. I do not require randomly connected association networks (Papert, 1960), nor that features be calculated in an essentially parallel manner from local or conjunctively local points of a retina on which patterns are projected (Minsky & Papert, 1969). My terminology and formulation of the learning procedure are based on Chapter 11 of Minsky & Papert (1969).

It is assumed in this section that query vectors contain a completely redundant component, obviating the necessity for the distinction between Φ and Φ'. The weight vectors $W^{(j)}$ are initially chosen at random. If a query vector $\Phi \varepsilon F^{(i)}$ with

$$W^{(i)}.\Phi \leq W^{(j)}.\Phi \quad \text{for some } j,$$

is encountered, $W^{(i)}$ is replaced by $W^{(i)}+\hat{\Phi}$, and $W^{(j)}$ by $W^{(j)}-\hat{\Phi}$. ($\hat{\Phi}$ denotes the unit vector in the direction of Φ, i.e. $\Phi/|\Phi|$.)

It is worth noting that the weight vectors are changed only if the perceptron would have classified the query vector wrongly. In general this process is sensitive to the outer boundaries of the

F-classes and relatively insensitive to the points inside.

For the two class case we need only a single weight vector W, and the learning procedure can be represented thus:

START: Choose any value for W;

TEST: Choose $\Phi \varepsilon\ F^{+} \cup F^{-}$;

If $\Phi \varepsilon F^{+}$ then if $W.\Phi > 0$ then go to TEST,

else go to ADD;

If $\Phi \varepsilon F^{-}$ then if $W.\Phi < 0$ then go to TEST,

else go to SUBTRACT;

ADD: Replace W by $W+\hat{\Phi}$;

Go to TEST;

SUBTRACT: Replace W by $W-\hat{\Phi}$;

Go to TEST.

Writing

$$F' = \{\Phi \mid \Phi \varepsilon F^{+}\} \cup \{-\Phi \mid \Phi \varepsilon F^{-}\},$$

this is equivalent to the following:

START: Choose any value for W; . . . (A)

TEST: Choose any $\Phi \varepsilon F'$;

If $W.\Phi \leq 0$ then replace W by $W+\hat{\Phi}$;

Go to TEST.

One of the reasons for the interest which has been shown in the perceptron strategy is that there exists a theorem, the oft-quoted Perceptron Convergence Theorem, which states that the learning scheme must lead to a weight vector which discriminates correctly between the pattern classes if one exists, that is, if the environment is linearly separable. The proof of the theorem involves no assumptions about the order in which the query vectors are presented, the finiteness of the

set F', or the dimensionality of the feature space. An elegant proof is given by Minsky & Papert (1969); this is based on a proof by Papert (1960). The former show how the theorem can be generalized without difficulty to the case where discrimination between n (n>2) pattern classes is required. They also point out that the perceptron convergence theorem is merely another way of looking at the results obtained on relaxation methods for linear inequalities (see for example Agmon, 1954).

The Perceptron Convergence Theorem. Let F' be a set of vectors such that there exists W* and $\delta>0$ with $\hat{W}^*.\hat{\Phi}>\delta$ for all $\Phi\varepsilon F'$. Then the program (A) above will alter W only a finite number of times, provided the weight vector W is initialized to have unit modulus.

Assuming that the program is presented a sequence of query vectors in which each $\Phi\varepsilon F'$ is repeated sufficiently often, it follows that a weight vector W for which

$W.\Phi>0$ for all $\Phi\varepsilon F'$

will eventually be found. With such a solution vector, the discrimination problem is solved for the two class case, since

$\Phi\varepsilon F^+$ implies $W.\Phi>0$;

$\Phi\varepsilon F^-$ implies $W.(-\Phi)>0$ implies $W.\Phi<0$.

The theorem also applies when a misclassified vector is added to the weight vector without being normalized first, i.e. when the program (A) is modified by replacing $\hat{\Phi}$ (in the second line) by Φ, provided that the query vectors Φ are bounded in length. This is always the case in real situations, and since normalization is a time-consuming operation on conventional computers, this variant is used henceforth.

It is important to realize that the theorem guarantees learning in a stronger sense than merely cycling through, or randomly trying, the states of a discrete machine until an acceptable state is found. The learning of a perceptron is self-directing, and can genuinely be described as goal-seeking; albeit with a goal which is perhaps rather trivial. There is an interesting parallel here with the evolutionary process:

> Some biologists have argued that the process of random mutation and natural selection is insufficient to account for evolutionary changes as they have occurred, and that some other guiding principle must play a part. Whether or not this is so will not be argued here but, whatever the mechanism, natural evolution is a slow and wasteful process. (Andrew, 1963.)

Random selection or mutation of states by a naive perceptron is also a slow and wasteful process, and the training procedure is designed to inject a sense of direction into the perceptron's wanderings.

It is natural to ask what happens to the perceptron learning scheme if the environment is not linearly separable. It has been noticed that the weight vector eventually oscillates in this case; the apparent frustration providing a valuable clue as to when to stop training (Efron, 1963). Minsky & Papert (1969) formalized this in their "Perceptron Cycling Theorem", showing that the weight vector remains bounded in length, and thus, if the set F' of query vectors is finite, the system eventually oscillates.

As far as I know, there has been no indication in the literature of the performance of the perceptron in situations with a controlled amount of noise. It is clear that if the size of the weight vector is roughly the same as that of the query vectors, and a query vector which has been corrupted by noise is presented and

identified wrongly by the perceptron, then the weight vector will be changed significantly. This means that a weight vector which correctly discriminates the noiseless pattern classes could be changed radically by just one noisy query vector, and if this happens a correctly discriminating weight vector will have to be relearnt. In a computer simulation the perceptron reached a discriminating state on the 91'st iteration, and for over half of the next 900 moves it was in an incorrectly discriminating state, due to the effects of noise which struck on the average only one out of every 20 query vectors. In this run there was no restriction on the length of the weight vector: it increased from 7 on the 91'st iteration to 12 on the 1000'th. The query vectors had an average length of 2.

If the weight vector is significantly longer than the query vectors, the perceptron appears to be much more stable during the learning period than it is when a small weight vector is used. The difference between the partially learned discriminations before and after modification of the weight vector by an incorrectly classified query vector is usually very great if the weight vector is small, and the perceptron gives the appearance of oscillating wildly. With a large weight vector, on the other hand, the transitions of the learning process take place much more smoothly.

There is reason to suspect that a larger weight vector will be less vulnerable to noise perturbations. For, suppose W is a correctly discriminating weight vector for the two class case,

$W.\Phi>\delta$ for all $\Phi\varepsilon F^{+}$,

$W.\Phi<-\delta$ for all $\Phi\varepsilon F^{-}$; for some $\delta>0$.

Suppose all query vectors have length at most α. Now if instead of W

we use the weight vector $W'=n.W$ $(n>0)$, we can permit up to $n\delta/\alpha^2$ wrongly categorized (due to noise) query vectors to modify W' without changing the discrimination effected by the perceptron. For, let $m < n\delta/\alpha^2$, and suppose $\Phi_1', \Phi_2', \ldots, \Phi_m'$ are the corrupted query vectors. Then the new weight vector W'' is given by

$$W'' = W' \pm \Phi_1' \pm \Phi_2' \pm \ldots \pm \Phi_m',$$

where the alternative signs depend on the class to which Φ_i' was assigned. Now

$$\begin{aligned} W''.\Phi &= nW.\Phi \pm \Phi_1'.\Phi \pm \ldots \pm \Phi_m'.\Phi \\ &> n\delta - m\alpha^2 \\ &> n\delta - (n\delta/\alpha^2).\alpha^2 = 0, \end{aligned}$$

so $W''.\Phi > 0$, for all $\Phi \epsilon F^+$;

and similarly,

$W''.\Phi < 0$ for all $\Phi \epsilon F^-$.

With a sufficiently long and unlucky sequence of noisy query vectors, the discrimination for the noiseless vectors will eventually become incorrect no matter how large the weight vector is. One hopes that the learning scheme will put the perceptron back on the right track with less loss, in terms of incorrectly classified noiseless points, than would have been occasioned had the weight vector been small.

The snag is that a longer training sequence is needed if the weight vector is to be large. The perceptron convergence theorem provides us with a theoretical upper bound to the number of mistakes made during the training period, and an extension to this theorem, given in Appendix A (Theorem 1), shows that this upper bound increases linearly with the size of the weight vector. This result holds if the weight vector is set initially to a fixed size and allowed to vary

during learning, as it does in the perceptron outlined above. Non-trivial lower bounds for the number of mistakes made are difficult to find, since the weight vector is initialized arbitrarily and this arbitrary vector may itself discriminate correctly between the pattern classes.

If the size of the weight vector is constrained only initially, there is a chance that it will decrease significantly during learning, thus destroying all point in having a size restriction. It is intuitively clear that this can happen in sufficiently unlucky circumstances; but for disbelievers the following result is proved in Appendix A (Theorem 3): If the weight vector is set initially to length λ and allowed to vary according to the usual perceptron adaptation rules, then there is a non-trivial environment for which the final weight vector is small in length, provided an unlucky choice is made for the initial weight vector. It can of course happen that the length of the weight vector increases during learning.

Because of this variability in the length of the final weight vector, it was thought best to consider an adaptation rule which renormalized the weight vector to length λ each time it was changed. Unfortunately, preliminary investigation revealed that the perceptron convergence theorem does not hold in this case. For a simple counter-example to the theorem, suppose $\lambda>1$ and consider $F'=\{(1,0)\}$, a subset of R^2 (the space of pairs of real numbers) with only one element. $W^*=(1,0)$ is a unit vector for which

$$\Phi \varepsilon F' \text{ implies } W^*.\Phi > 1/2, \quad (\text{with } \delta = 1/2);$$

but if the initial choice of W is $W_0=(-\lambda,0)$, then

$$W_1 = \lambda(-\lambda+1,0)/[(-\lambda+1)^2 + 0^2]^{1/2} = (-\lambda,0) = W_0.$$

Hence the weight vector remains unchanged no matter how many times the query vector (1,0) is misclassified. Less trivial counterexamples can easily be found; and if one is prepared to impose some kind of order on the sequence in which query vectors are presented it is possible to find counterexamples which involve a pattern class F of considerable complexity. However, the convergence theorem in its strict form does not depend on the order in which the query vectors are presented, and the above example shows that it does not hold for the adaptation rule described here.

An obvious solution to our dilemma is to renormalize the weight vector after convergence has been reached. The snag here is the difficulty, discussed in the last chapter, of determining when convergence has been reached. Some feedback about how adaptation is progressing is often required to decide when to terminate the training period, and this may be found helpful in determining when to renormalize. Indeed, if the machine is trained for a certain period of time and then left to fend for itself, rather than being more or less continuously monitored and partially trained all its life, then it seems reasonable to renormalize the weight vector to the desired length on termination of training. This may be considered undesirable if the machine is to continue running unmonitored, though, since the last thing one wants before leaving it to itself is a radical change in the perceptron's internal structure. Note that since the training patterns will generally produce noisy query vectors, the weight vector should be large during training, and if necessary it could be re-normalized during this period at the trainer's discretion, provided he bears in mind that this could considerably retard or even halt the

learning process. I am of the opinion that this apparent requirement for interfering with the internal structure of the perceptron severely weakens any claim it may have to being a "self-organizing" machine.

A modification of the perceptron learning procedure, used by Griffin *et al.* (1963), ensures that the weight vector becomes large but does not appear to suffer from the disadvantages of the methods discussed above. It consists of seeking a *corridor* of specified width separating the pattern classes, rather than merely a line, and this is effected by using the rules

If $\Phi \varepsilon F^{+}$ and $W.\Phi \leq d$ then replace W by $W+\Phi$;

If $\Phi \varepsilon F^{-}$ and $W.\Phi \geq -d$ then replace W by $W-\Phi$;

for some constant $d>0$. These rules are rather similar to the use of a hysteresis corridor to help a machine to "make up its mind" when digitizing a continuous input signal (Hill & Wacker, 1969). Griffin reports "a simple but significant improvement" in his character recognizer if the usual perceptron adaptation strategy is replaced by these adaptation rules, and I shall call this scheme the *threshold perceptron* strategy. The constant d is referred to as the *threshold*. The rules can easily be generalized to more than two pattern classes. A reject class can be used when classifying unknown vectors:

Reject Φ if $-\theta d < W.\Phi < \theta d$,

where θ is a positive constant, usually less than 1.

It is easy to see that the threshold perceptron strategy has the effect of forcing the size of the weight vector up as d increases. For, suppose ε is a number such that there does not exist a unit vector X with

$X.\Phi > \varepsilon$ for all $\Phi \varepsilon F'$,

where

$F' = \{\Phi | \Phi \varepsilon F^{+}\} \cup \{-\Phi | \Phi \varepsilon F^{-}\}$ as before.

(Such ε's exist: $\varepsilon=\alpha$ will do the trick.) Then if W is a solution vector for the threshold perceptron,

$W.\Phi > d$ for all $\Phi \varepsilon F'$.

Hence

$\hat{W}.\Phi > d/|W|$ for all $\Phi \varepsilon F'$.

Now $(d/|W|) \geq \varepsilon$ implies $\hat{W}.\Phi > \varepsilon$ for all $\Phi \varepsilon F'$,

which contradicts the assumptions;

so $(d/|W|) < \varepsilon$.

Hence

$|W| > d/\varepsilon$.

So given $\lambda > 0$, we can find d such that if the threshold perceptron converges, its final weight vector has length greater than λ (e.g. take $d=\lambda\alpha$). The convergence theorem for the threshold perceptron is given in Appendix A (Theorem 2); the upper bound on the number of times W is changed increases linearly with d.

The threshold perceptron has additional resources to fight noise, apart from its guarantee of a large weight vector. Suppose a solution weight vector W has been found, but owing to noisy conditions this has been perturbed to W' (by misclassified noisy query vectors). Then the machine is able to "realize" that its weight vector has been perturbed before it begins to misclassify noise-free vectors, and it begins to correct itself before making mistakes. For, suppose W' gives a discrimination which is dangerously close to misclassifying some noise-free vectors, i.e.

$W'.\Phi < d$ for some $\Phi \varepsilon F'$.

Then if one of the vectors in danger is encountered soon enough, the perceptron adjusts itself to alleviate the danger before a noise-free vector is misclassified, whereas an ordinary perceptron only adjusts itself after a vector has been misclassified.

The threshold perceptron still, however, persists in the futile attempt to correct itself for noisy vectors, as does the ordinary perceptron. The above argument only applies to situations where the noise level is low, so that one can think of the query vectors as comprising a large core of noise-free vectors with some stray noisy ones. It is expected that the behaviour of the threshold perceptron will deteriorate rapidly as the noise level increases.

Having decided that the threshold perceptron has a better chance of performing well in noisy conditions than any other variant considered (or any other perceptron-like decision strategy that I have found in the literature), it was decided to simulate it to see just how well it does. The details and results of the simulation are presented in Chapter 7.

Chapter 4

CLASSIFICATION USING STATISTICAL DECISION TECHNIQUES

Statistical decision techniques are often used as the basis of a pattern-classifying system. One of the advantages of this is that the adaptation of the system can be accomplished simply by estimation of the appropriate probabilities (or probability distributions), and this process is amenable to theoretical treatment. Consequently we deal in this chapter only with the decision procedures, and the learning part is considered separately in Chapter 5.

The problem of statistical classification is usually formulated in terms of the loss function (cost function) of decision theory. The loss function is defined on the Cartesian product of the set of pattern classes (possibly augmented by a reject class or a "no signal present" class) and represents the cost of deciding that a query vector is in class $F^{(i)}$ when in fact it belongs to class $F^{(j)}$. This gives a generality which, from the point of view of this thesis, is rather vacuous: although the loss function may be both non-trivial and known for certain commercial applications, it is usually either trivial or unknown and assumed trivial for convenience. In experimental situations, where the decision process is used mainly to test (in order to improve) the feature extraction, a trivial loss function is invariably used. After formulating the classification problem in a very general way using the statistical decision model, Chow (1957) remarked that "an optimum system may prove to be too expensive for mechanization". His opinion is confirmed by the

simplifications and approximations used in the various implementations of these decision methods (including his own: see Chow, 1962), and only techniques which have been used in or seriously proposed for practical machines are considered here.

The simplest and by far the most frequently used statistical classification technique is the maximum likelihood decision rule:

Given a query vector Φ, choose the pattern class whose *a posteriori* probability is greatest, i.e.

Choose class i if

$$\Pr[F^{(i)}|\Phi] \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j \qquad \ldots \text{(A)}$$

This rule is obtained from the general decision theory model if the loss function is "symmetric", that is, if correct decisions cost nothing and incorrect decisions all cost the same amount (Nilsson, 1965). For any classification system which has access only to the query vector Φ and the various conditional probabilities associated with the vectors and pattern classes, the above rule minimizes the number of mistakes made (Chow, 1957). It is in this sense that the decision is often called "optimal" (Minsky & Papert, 1969), but it should be emphasized that this optimality depends on correct assessment of the conditional probabilities. These probabilities can in principle be estimated to any desired accuracy if the training period is sufficiently long and the training patterns are sufficiently representative, but the amount of space necessary for their storage is prohibitively large, and so approximations are used which of course destroy the optimality of the decision.

A variant of this simple statistical classification rule concerns situations where the decisions may be taken on data resulting

from noise alone (Middleton, 1960). An additional class, $F^{(noise)}$, is introduced here, and the decision rule is:

Choose class i if

$$\Pr[F^{(i)}|\Phi] \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j, \qquad \ldots \text{(B)}$$

and $\Pr[F^{(i)}|\Phi] \geq \Pr[F^{(noise)}|\Phi]$;

decide that noise alone is present if

$$\Pr[F^{(noise)}|\Phi] \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j.$$

In most classification systems the problem of deciding if a signal is present or not is assigned to the feature extraction phase rather than to the decision process, and this extra "noise" pattern class is not used--even in speech recognition applications where determining if a signal is present or not is a difficult problem (Reddy, 1967).

One often wishes to reject a query vector if the recognizer is uncertain about its class. Chow (1957) showed that the following decision rule minimizes the error rate for a given rejection rate:

Choose class i if

$$\Pr[\Phi|F^{(i)}]\Pr[F^{(i)}] \geq \Pr[\Phi|F^{(j)}]\Pr[F^{(j)}] \quad \text{for all } j \qquad \ldots \text{(C)}$$

and $\Pr[\Phi|F^{(i)}]\Pr[F^{(i)}] \geq \beta.\Sigma\, \Pr[\Phi|F^{(k)}]\Pr[F^{(k)}]$;

reject Φ if

$$\beta.\Sigma\, \Pr[\Phi|F^{(k)}]\Pr[F^{(k)}] \geq \Pr[\Phi|F^{(j)}]\Pr[F^{(j)}] \quad \text{for all } j.$$

The small positive constant β controls the rejection rate. This rule corresponds to the decision theory model with a loss function for which

a) all correct decisions cost nothing,

b) all incorrect decisions cost the same, c_1,

c) all rejections cost the same, c_2;

where of course $c_1 > c_2$. The rule is equivalent to

Choose class i if

$$\Pr[F^{(i)}|\Phi] \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j$$

and $\Pr[F^{(i)}|\Phi] \geq \beta . \Sigma \Pr[F^{(k)}|\Phi]$.

Note that

$$\Sigma \Pr[F^{(k)}|\Phi] = 1,$$

provided that the pattern classes cover the space of query vectors (this is almost always true if the feature extractors eliminate "no signal present" query vectors, since the pattern classifier is expected to classify even unusual patterns by generalization). Hence a query vector is rejected if and only if the *a posteriori* probability of every class is less than the constant β. With this in mind, the decision rule becomes:

Choose class i if

$$\Pr[F^{(i)}|\Phi] \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j \qquad \ldots \text{(D)}$$

and $\Pr[F^{(i)}|\Phi] \geq \beta$;

reject Φ if

$$\beta \geq \Pr[F^{(j)}|\Phi] \quad \text{for all } j.$$

This decision rule is equivalent to the introduction of a "noise" class if $\Pr[F^{(\text{noise})}|\Phi]$ is considered to be independent of Φ. Also, there is no explicit provision for adaptation of the threshold (noise probability), since reject decisions cannot be reinforced whereas "noise only" decisions can. Adaptation of the threshold could be introduced during the training period by *ad hoc* methods.

To implement any of these classification schemes properly, we must store $\Pr[F^{(j)}|\Phi]$ for each pattern class j and each query vector Φ. Unfortunately the space required for storing these probabilities is, in general, extremely large. If Φ is n-dimensional and each

component ϕ_i can assume one of r values, the distribution $Pr[F^{(j)}|\Phi]$ is specified by r^n-1 values for each j, and hence $(m-1)(r^n-1)$ values are required for complete storage if there are m pattern classes. Suppose, to take a modest numerical example, r=n=m=10. Then around 10^{11} quantities are required to specify all the probabilities (Marill & Green, 1960).

A simplifying assumption is that of independence of the ϕ_i's relative to the pattern classes, i.e. assume

$$Pr[\Phi|F^{(i)}] = \Pi\ Pr[\phi_k|F^{(i)}].$$

Then we can write

$$Pr[F^{(i)}|\Phi] = Pr[\Phi|F^{(i)}]Pr[F^{(i)}]/Pr[\Phi]$$

$$= \frac{Pr[F^{(i)}]}{Pr[\Phi]} .\Pi\ Pr[\phi_k|F^{(i)}] \qquad \ldots (4.1)$$

This decreases storage requirements considerably. The price we pay for this reduction, the independence assumption, is discussed at length in Chapter 6; it is generally acknowledged in the literature only with a passing warning that it is a "strong" condition. Suffice it to say here that it seems to be a fundamental stumbling-block to the statistical classification methods, and one that will not be overcome except by *ad hoc* methods applicable only to restricted classes of problems.

For the decision rule (A), where we choose the pattern class whose *a posteriori* probability is greatest with no restrictions on the absolute sizes of the probabilities, we can ignore the common factor $1/Pr[\Phi]$ in the expression (4.1) to get the standard rule:

Choose class i if

$$Pr[F^{(i)}].\Pi\ Pr[\phi_k|F^{(i)}] \geq Pr[F^{(j)}].\Pi\ Pr[\phi_k|F^{(j)}] \quad \text{for all } j \qquad (4.2)$$

Only n.r.m quantities are required now for storage of the probabilities. The introduction of a "noise" pattern class, as in (B), requires a simple amendment to the rule; the additional probabilities are estimated in exactly the same way as those for the other pattern classes.

Chow's decision rule (C) also does not require storage of $Pr[\Phi]$; it effectively calculates this using the relation

$$Pr[\Phi] = \Sigma\ Pr[\Phi|F^{(k)}]Pr[F^{(k)}] \qquad \ldots\ (4.3)$$

However, if the independence assumption is used to determine $Pr[\Phi|F^{(k)}]$, errors are caused both by incorrect assessment of the values of $Pr[\phi_j|F^{(k)}]$ (caused by inadequacy of the training set) and by the inevitable invalidity of the independence assumption. Hence the summation in (4.3) leads to an accumulation of errors which will almost certainly be quite large in normal circumstances, and this will cause inconsistent and probably rather arbitrary rejection. I have found no practical work reported in the literature which uses this rejection criterion--in fact few experimental workers in this field use a reject class at all.

The decision rule (D) requires knowledge of the absolute values of $Pr[F^{(j)}|\Phi]$. To use the simple Bayesian inversion and the independence assumption above, as in (4.1), we must effectively store $Pr[\Phi]$, which requires around r^n values for complete storage. Alternatively we can assume another form of independence: that the components of Φ are statistically independent. This is much stronger than the original assumption and its effect in real situations can only be to increase the rate of misclassifications for a given rejection rate.

Another method of implementing the rule (D) is to note that maximizing $\Pr[F^{(j)}|\Phi]$ is equivalent to maximizing $\Pr[F^{(j)}|\Phi]/\Pr[\overline{F}^{(j)}|\Phi]$ over the j's, and that

$$\Pr[F^{(j)}|\Phi] \geq \beta \quad \text{if and only if}$$
$$\Pr[F^{(j)}|\Phi]/\Pr[\overline{F}^{(j)}|\Phi] \geq \beta/(1-\beta).$$

Now

$$\frac{\Pr[F^{(j)}|\Phi]}{\Pr[\overline{F}^{(j)}|\Phi]} = \frac{\Pr[\Phi|F^{(j)}]\Pr[F^{(j)}]}{\Pr[\Phi|\overline{F}^{(j)}]\Pr[\overline{F}^{(j)}]}$$
$$= \frac{\Pr[F^{(j)}]}{\Pr[\overline{F}^{(j)}]} \cdot \Pi \frac{\Pr[\phi_k|F^{(j)}]}{\Pr[\phi_k|\overline{F}^{(j)}]},$$

using our original independence assumption. Following Good (1965), we define the *weight of evidence in favour of X provided by Y*:

$$W[X:Y] = \log \frac{\Pr[Y|X]}{\Pr[Y|\overline{X}]}.$$

Then, writing

$$R^{(j)}_{\text{prior}} = \log(\Pr[F^{(j)}]/\Pr[\overline{F}^{(j)}]),$$
$$R^{(j)}_{\text{post}}(\Phi) = \log(\Pr[F^{(j)}|\Phi]/\Pr[\overline{F}^{(j)}|\Phi]);$$

we assign Φ to the class $F^{(i)}$ which maximizes

$$R^{(i)}_{\text{post}}(\Phi) = R^{(i)}_{\text{prior}} + \Sigma\, W[F^{(i)}:\phi_k].$$

This gives a rather natural interpretation of the discriminatory functions in terms of summing weights of evidence--or in fact in terms of summing the self-information provided by the Φ-components to the pattern class, since as Good (*op. cit.*) pointed out,

$$W[X:Y] = I[X:Y] - I[\overline{X}:Y],$$

where I denotes the information function.

We may decide to reject the query vector Φ if

$$\Pr[F^{(j)}|\Phi] \leq 1/2 \quad \text{for all } j,$$

i.e. if

$$R^{(j)}_{post}(\Phi) \leq 0.$$

This, or rather the converse, that Φ is certainly not rejected if for some j,

$$R^{(j)}_{post}(\Phi) > 0,$$

is exactly the result obtained by Maron (1962) as the hypothesized condition for a neuron's firing.

A price must be paid for the rejection threshold, for now both $\Pr[\phi_k|F^{(j)}]$ and $\Pr[\phi_k|\overline{F}^{(j)}]$ must be stored, together with the *a priori* probabilities, instead of merely the former as before. However, these need only be stored during the training period since on termination of training, $\Pr[\phi_k|F^{(j)}]/\Pr[\phi_k|\overline{F}^{(j)}]$ can be computed and stored instead. Alternatively an algorithm could be used for estimating weights of evidence directly, but I know of no such procedure.

For the purposes of pattern recognition, decision techniques are very often used in conjunction with query vectors whose components are binary-valued. This follows from the fact that the feature measurements usually serve to denote the presence or absence of some attribute, rather than the degree to which it occurs. The statistical classification method turns out to have a particularly simple form for binary features because $\Pr[\phi_i|F^{(j)}]$ is represented by a single number rather than by a probability distribution or density function. However, a few remarks on the non-binary case are in order here before proceeding to treat binary features.

In situations where the query vector components may take on continuous or pseudo-continuous range of values, one normally assumes a probability density function which depends on certain parameters such as the mean, variance, and possibly higher moments, which are to be estimated. It can be shown (Highleyman, 1961) that the optimum decision surface between Gaussian distributions with equal *a priori* probabilities is a hyperplane. If the covariances are not equal, however, the maximum likelihood boundary is non-linear (Cooper, 1963). I will not concern myself with cases where the distribution is considered to be continuous.

In the discrete case, if the number r_i of the values $\psi_{i1},\psi_{i2},\ldots,\psi_{ir_i}$ which may be taken on by ϕ_i is reasonably small, it should be possible to estimate the probabilities $Pr[\phi_i=\psi_{im}|F^{(j)}]$ separately. The decision surface is in general non-linear and assumes quite complicated shapes.

A limited series of experiments was made to determine if either of the following two methods of treating a discrete non-binary query vector space is significantly better than the other:

a) estimate the quantities $Pr[\phi_i=\psi_{im}|F^{(j)}]$ separately;

b) encode the non-binary features into the positional binary notation and use the binary features so generated.

(As noted in Chapter 6, it is probably better to use a binary encoding which is highly redundant but preserves the topological properties of the environment. Method (b) is, in this sense, perhaps unfair to the binary feature system.) The various probabilities were calculated exactly by a frequency count using each point of the environment in turn. The environments used were mostly linearly separable, but

because of the invalidity of the independence assumption, mistakes were normally made by both methods.

Neither method seemed to be significantly more powerful than the other; the discriminatory surfaces for (b), although linear in the binary hyperspace (see below), assumed as complicated and seemingly arbitrary shapes when re-encoded into the original non-binary feature space as those for (a). It appears from this that if the query space is to be treated as discrete, no significant loss is suffered by considering the binary encodings of the original query vector components as new features, and the binary method may prove superior if an appropriate redundant but "helpful" coding scheme is used.

If the features ϕ_i are binary, and if the independence assumption is used, the maximum likelihood pattern classification becomes linear. This was shown by Minsky & Selfridge (1960), and the following is based on their proof.

Define

$$p_{ki} = \Pr[\phi_k=1 | \Phi \varepsilon F^{(i)}], \quad q_{ki} = 1 - p_{ki};$$
$$p_i = \Pr[F^{(i)}].$$

The discriminatory functions are (see equation 4.2)

$$g^{(i)}(\Phi) = \Pr[F^{(i)}].\Pi \Pr[\phi_k | F^{(i)}],$$

assuming independence. Since log is a monotonically increasing function, we may use the amended discriminatory functions

$$\begin{aligned} f^{(i)}(\Phi) &= \log\{\Pr[F^{(i)}].\Pi \Pr[\phi_k | F^{(i)}]\} \\ &= \log(p_i) + \Sigma \log\{(p_{ki})^{\phi_k}.(q_{ki})^{(1-\phi_k)}\} \\ &= \Sigma \phi_k.\log(p_{ki}/q_{ki}) + \log(p_i) + \Sigma \log(q_{ki}) \\ &= \Sigma \phi_k w_{ki} + \theta_i; \end{aligned}$$

where

$$w_{ki} = \log(p_{ki}/q_{ki}),$$

$$\theta_i = \log(p_i) + \Sigma \log(q_{ki}).$$

Defining

$$W^{(i)} = (w_{1i}, w_{2i}, \cdots, w_{ni}, \theta_i),$$

the decision rule takes the form of maximizing $\Phi'.W^{(i)}$ by choice of i.

In trying to gain some feeling for this decision rule, we shall ignore rigour and imagine the n-dimensional binary query space as being continuous. Consider the decision surface between the classes $F^{(i)}$ and $F^{(j)}$. This has equation

$$\Phi'.(W^{(i)} - W^{(j)}) = 0.$$

This hyperplane is normal to the join of the points $(w_{1i}, w_{2i}, \cdots, w_{ni})$ and $(w_{1j}, w_{2j}, \cdots, w_{nj})$ in our n-dimensional space. Note that the "centre of gravity" of the class $F^{(i)}$ (mean value of $\{\Phi | \Phi \varepsilon F^{(i)}\}$) is represented by the point

$$C^{(i)} = (p_{1i}, p_{2i}, \cdots, p_{ni}).$$

This shows that the discriminatory hyperplane is normal to the join of the images of $C^{(i)}$ and $C^{(j)}$ under the transformation

$$x \rightarrow \log\{x/(1-x)\}$$

applied to each of the co-ordinates. This should be contrasted with the perceptron learning scheme which is in general sensitive to the outer boundaries of the pattern classes rather than to their interiors. As pointed out by Nilsson (1965), the equation of the hyperplane depends in a reasonable way on the probabilities involved. As p_{mi} increases with p_{mj} constant, w_{mi} and hence $(w_{mi} - w_{mj})$ increases. This favours an $F^{(i)}$ response for query vectors with

$$\phi_m = 1.$$

On the other hand, the m'th component of Φ is ignored if and only if

$$w_{mi} - w_{mj} = 0,$$

i.e. if and only if

$$p_{mi} = p_{mj},$$

in which case that component contributes nothing to the discrimination between $F^{(i)}$ and $F^{(j)}$. The *a priori* probabilities of the pattern classes affect only the thresholds θ, and if $F^{(i)}$ becomes less likely then θ_i decreases and the decision surface moves toward $F^{(j)}$.

With a finite number of samples in the training set it may happen that some p_{ij} or q_{ij} becomes zero. Usually, though, the conditional probabilities are smeared, partly because they are often estimated by an iterative process which rarely gives zero or one (except in storage limited cases where only a small set of values is available for probabilities), and partly because of noise perturbations. If the independence assumption is valid and the conditional probabilities are known for *separated* patterns in hyperspace, then the maximum likelihood classification becomes trivial since many of the conditional probabilities will assume their extreme values of zero or one, causing all but one $\Pr[\Phi|F^{(j)}]$ to vanish for any Φ (see Chapter 6). However, if one wishes to preserve the formalism in these cases and use log probabilities in the form of weights of evidence, there is theoretical justification for replacing the usual frequency estimate $H_i^{(j)}/N^{(j)}$ for $\Pr[\phi_i=1|F^{(j)}]$, where $H_i^{(j)}$ is the number of occurrences of Φ's in $F^{(j)}$ with i-component 1 out of a sample of $N^{(j)}$, by $\{H_i^{(j)} + 1\}/\{N^{(j)} + 2\}$, thus avoiding the problem of zero probabilities (Good, 1965).

Chapter 5

STANDARD PROBABILITY ESTIMATION TECHNIQUES, AND SOME COMPLICATIONS ARISING FROM STORAGE LIMITATIONS

I regard this chapter, or at least the first part of it, as a necessary evil. Techniques for probability estimation are well known and have been used in learning machines for some time now, and I feel that little if any improvement can be made to these. However, our discussion of the maximum likelihood decision method is of little consequence unless these techniques are described; in addition to this it is difficult to find a complete treatment of more than one probability estimation procedure in any one place in the literature. An exception to this is provided by Minsky & Papert (1969), and the first part of this chapter is based on their exposition, with some alterations and additions of my own. It was not thought worthwhile to carry the argument to several decimal places, and means, variances, and limits are assumed to exist whenever this is convenient.

One often requires learning machines to repeatedly estimate the probability of "favourable" events in some continuing process. Normally this cannot be calculated directly, since it is by definition a limit, and so one must find *estimators*. The simplest way to estimate a probability in situations of this kind is to find the ratio h/n of the number h of favourable events to the total number of events so far experienced.

Define

ϕ_n = ⌜the n'th event is favourable⌝;

let p be the true probability that an event is favourable;

p_n be the estimate of p after n trials.

Then the formula

$$p_n = (1 - 1/n)p_{n-1} + (1/n)\phi_n \qquad \ldots (5.1)$$

computes the frequency ratio

$$p_n = \frac{1}{n} \cdot \Sigma\, \phi_i = h/n.$$

Making the usual assumption that successive events are independent, and noting that ϕ_i is binomially distributed, we have

$$E[p_n] = p, \qquad \ldots \text{(expectation)}$$

$$\mathrm{Var}[p_n] = p(1-p)/n. \qquad \ldots \text{(variance)}$$

Note that p_0 is truly arbitrary, since the value of p_n $(n \geq 1)$ is independent of p_0.

The estimate $(h+1)/(n+2)$ was mentioned at the end of the last chapter; this avoids some problems which can arise from zero probabilities. The formula

$$p_n = (1 - \frac{1}{n+2}) \cdot p_{n-1} + \frac{1}{n+2} \cdot \phi_n , \qquad \ldots (5.2)$$

$$p_0 = 1/2;$$

computes

$$p_n = \frac{1}{n+2} \cdot (2p_0 + \Sigma\, \phi_i) = \frac{h+1}{n+2} .$$

It behaves as the previous procedure would if one ϕ=1 and one ϕ=0 were observed in two moves, before starting to observe the ϕ's proper.

Consider the alternative estimator

$$p_n = (1-\theta)p_{n-1} + \theta\phi_n, \quad 0<\theta<1. \qquad \ldots (5.3)$$

This is the *exponentially weighted past average* (EWPA) procedure, and has the advantage that the current value of n need not be stored and appropriately incremented. The solution of the recurrence relation is

$$p_n = (1-\theta)^n p_0 + \theta \Sigma\ (1-\theta)^{n-i} \phi_i \qquad (n>0),$$

so $E[p_n] = (1-\theta)^n p_0 + p(1 - (1-\theta)^n)$

$$\rightarrow p \quad \text{as} \quad n \rightarrow \infty, \quad \text{for all } p_0.$$

Also,

$$\text{Var}[p_n] = p(1-p).\theta(1 - (1-\theta)^{2n})/(2-\theta)$$

$$\rightarrow \frac{\theta}{2-\theta} .p(1-p).$$

It can be seen that recency outweighs experience, for this estimation procedure, since the coefficients of ϕ_i decay exponentially with time. This is an advantage for many pattern recognition systems, since the feature extractors may be changed (for example by varying thresholds) in a gradual but unpredictable manner, depending on the performance of the classifier. In the limit for large n, the expected value of p_n is p, and the variance can be made arbitrarily small by choosing θ small enough.

Following Minsky, we can "equate" the variances of procedures (5.1) and (5.3):

$$\frac{p(1-p)}{n} = \frac{\theta(1 - (1-\theta)^{2n})}{2 - \theta} .p(1-p) \simeq \frac{\theta}{2-\theta} .p(1-p) ,$$

so $n \sim 2/\theta$.

Thus the variance of procedure (5.3) is about the same as that obtained by averaging the last $2/\theta$ observations. We can think of $1/\theta$ as a time-constant for "forgetting". For small θ, there is slow adaptation but the variances are small and the final estimate is reliable. For large θ adaptation is fast but the limiting variance is large. Initially the situation is as though the probability had been estimated at p_0 on the basis of $\sim 1/\theta$ trials. Hence for small θ the influence of the arbitrary p_0 is present for some time, and for large

θ we run the risk of violent oscillations at the beginning. Samuel (1959) used an ingenious compromise in his checker-playing program. He set $p_0 = 1/2$ and used

$$p_{n+1} = (1 - 1/N)p_n + (1/N)\phi_{n+1} ,$$

where

$$N = \begin{cases} 16 & \text{if } n < 32 \\ 2^m & \text{if } 32 \leq n < 256 \text{ and } m \text{ is an integer with } 2^m \leq n < 2^{m+1} \\ 256 & \text{if } 256 \leq n . \end{cases}$$

This ensures stability at around 1/2 in the early stages, approximates uniform averaging in the middle, and finally settles down to an EWPA to ensure adaptation to changing circumstances.

Minsky & Selfridge (1960) used an adaptation rule which is only trivially different from the EWPA estimator:

$$p_n = (1-\theta)(p_{n-1} + \phi_n), \quad 0<\theta<1. \qquad \ldots (5.4)$$

Let $q_n = \theta p_n/(1-\theta)$.

Then

$$(1-\theta)q_n/\theta = (1-\theta)q_{n-1}.(1-\theta)/\theta + (1-\theta)\phi_n,$$

so $q_n = (1-\theta)q_{n-1} + \theta\phi_n,$

which is the same as rule (5.3). Hence the limiting expectation of p_n is $(1-\theta)p/\theta$,

and the limiting variance is

$$\frac{(1-\theta)^2}{\theta(2-\theta)} .p(1-p) .$$

One sometimes wishes to estimate the *likelihood ratio* $p/(1-p)$ directly. This can be done using

$$p_n = (1-\theta)p_{n-1} + \theta(1 + p_{n-1})\phi_n , \quad 0<\theta<1. \qquad \ldots (5.5)$$

Write

$$E_n = E[p_n]; \qquad E = Lt(E_n) \text{ as } n \to \infty;$$

$$\chi_n = E[p_n^2]; \qquad \chi = Lt(\chi_n) \text{ as } n \to \infty.$$

Then

$$E_n = \theta p + (1 - \theta + \theta p)E_{n-1},$$

so $$E_n = \theta p. \sum_{0}^{n-1} (1 - \theta + \theta p)^i + (1 - \theta + \theta p)^n . p_0$$

$$= \frac{p}{1-p} . (1 - (1-\theta+\theta p)^n) + p_0 . (1-\theta+\theta p)^n ,$$

provided

$$p \neq 1.$$

Hence

$$E = \frac{p}{1-p} , \text{ as required.}$$

Now $$p_n^2 = (1-\theta+\theta\phi_n)^2 . p_{n-1}^2 + 2\theta(1-\theta+\theta\phi_n)\phi_n . p_{n-1} + \theta^2 . \phi_n^2 .$$

We have

$$E[(1-\theta+\theta\phi_n)^2] = p + (1-\theta)^2 . (1-p) = (1-\theta)^2 + \theta(2-\theta)p ;$$

$$E[(1-\theta+\theta\phi_n) . \phi_n] = p;$$

and $E[\phi_n^2] = p.$

Hence

$$\chi_n = \{(1-\theta)^2 + \theta(2-\theta)p\} . \chi_{n-1} + 2\theta p E_{n-1} + p\theta^2 ,$$

so $$\chi = \{(1-\theta)^2 + \theta(2-\theta)p\} . \chi + 2\theta p^2/(1-p) + p\theta^2 .$$

This gives

$$\chi = \frac{p(2p+\theta-\theta p)}{(2-\theta)(1-p)^2} .$$

The limiting variance of p_n is given by

$$V = \chi - E^2$$

$$= \frac{\theta}{2-\theta} \cdot \frac{p}{(1-p)^2} .$$

Thus the limiting variance may be made as small as we please by choosing θ sufficiently small.

The effect of limited storage for probabilities on EWPA estimation is now considered by means of an example. Suppose the probabilities are stored as integer percentages, i.e. write

$$q_n = 100.p_n,$$

where the q's are stored as integers. The EWPA recurrence relation is

$$p_n = (1-\theta)p_{n-1} + \theta\phi_n$$
$$= p_{n-1} + \theta(1 - p_{n-1})\phi_n - \theta p_{n-1}.(1 - \phi_n) .$$

In terms of the q's, this is

$$q_n = q_{n-1} + \theta(100 - q_{n-1})\phi_n - \theta q_{n-1}.(1 - \phi_n). \quad \ldots (5.6)$$

Let the value of θ used be 2%. An acceptable digitization of (5.6) is (see Figure 5.1)

$$q_n = q_{n-1} + \Delta_i(q_{n-1}).\phi_n + \Delta_d(q_{n-1}).(1 - \phi_n),$$

where

$$\Delta_i(x) = \begin{cases} +2 & \text{if } x \leq 50 , \\ +1 & \text{if } x > 50 ; \end{cases} \quad \ldots (5.7)$$

$$\Delta_d(x) = \begin{cases} -1 & \text{if } x < 50 , \\ -2 & \text{if } x \geq 50 . \end{cases}$$

A digitization which sticks closer to the required line is

$$\Delta_i(x) = \begin{cases} +2 & \text{if } x \leq 25 , \\ +1 & \text{if } 25 < x \leq 75 , \\ 0 & \text{if } 75 < x ; \end{cases}$$

and similarly for $\Delta_d(x)$, but this suffers from the disadvantage that if p_n is in the range [25,75] then p_m (m>n) can never be less than 24

or greater than 76. Hence we use the digitization (5.7).

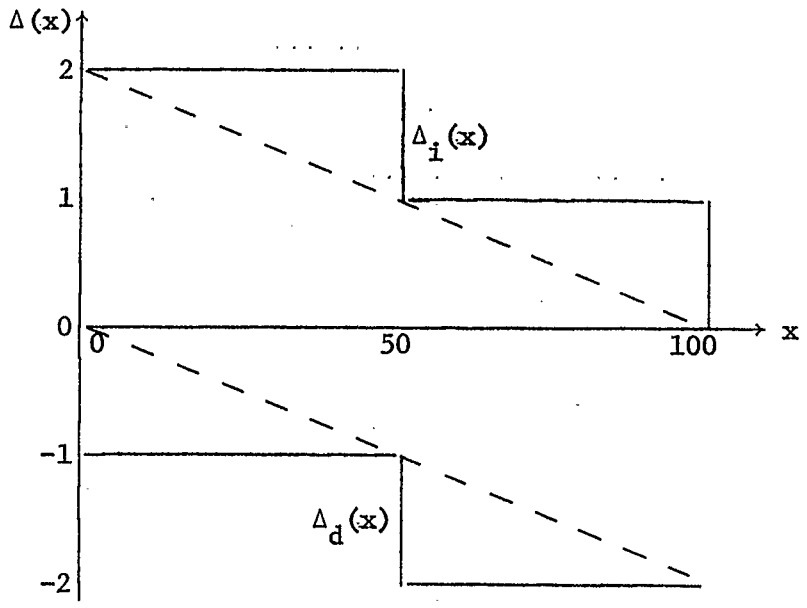

Figure 5.1

The digitization of the Δ-functions.

Now

$$E[q_n] = E[q_{n-1}] + p.E[\Delta_i(q_{n-1})] + (1-p).E[\Delta_d(q_{n-1})],$$

where p, as before, is $Pr[\phi_i=1]$. Assuming for convenience that q_n always remains above 50,

$$E[q_n] = E[q_{n-1}] + p - 2(1-p),$$

so if

$$p > 2(1-p),$$

i.e. if

$$p > 2/3,$$

then q_n is expected to increase steadily until it reaches the upper limit.

This is clearly an undesirable state of affairs. Moreover, the phenomenon is not confined to the example given; it will be present to a greater or lesser extent in any system which digitizes the probabilities and adjusts them in a straightforward manner. The example is not an extreme one which is unlikely to occur in a practical situation--on the contrary, the existence of storage limitation complications was brought home to me while experimenting with a machine which was a realization of just this example.

We have seen that the EWPA procedure (or in fact any other conventional probability estimation procedure) is spoiled by digitization unless the parameter θ is large compared with the precision to which the probabilities are stored. The only explicit reference I have seen in the literature to the following technique for overcoming this difficulty is due to Andreae (1969), although Uhr & Vossler (1963) used a similar technique without comment.

I propose using this system for incrementation and decrementation in the above example:

$$\Delta_i(x) = \begin{cases} 2 & \text{with probability } (100 - x)\%, \\ 0 & \text{" } \quad \text{" } \quad x\%, \end{cases}$$

$$\Delta_d(x) = \begin{cases} -2 & \text{" } \quad \text{" } \quad x\%, \\ 0 & \text{" } \quad \text{" } \quad (100 - x)\%. \end{cases}$$

Hence Δ_i and Δ_d can be interpreted as random variables dependent on the argument x, whose expected values, given x, are the same as the values of the Δ-functions given earlier. For a more general treatment, let us revert to the p's and re-define Δ_i and Δ_d by

$$p_n = p_{n-1} + \Delta_i(p_{n-1}).\phi_n + \Delta_d(p_{n-1}).(1 - \phi_n). \qquad \ldots (5.8)$$

We choose the Δ-functions to be

$$\Delta_i(x) = \begin{cases} \theta & \text{with probability } (1-x), \\ 0 & \text{" } \quad \text{" } \quad x, \end{cases}$$

$$\Delta_d(x) = \begin{cases} -\theta & \text{" } \quad \text{" } \quad x, \\ 0 & \text{" } \quad \text{" } \quad (1-x); \end{cases}$$

where θ is a positive constant, less than 1, chosen to be a multiple of the precision to which the p's are stored. We calculate the expectation and variance of the p's: these are assumed to exist.

Let $E_n = E[p_n]$; $\quad E = Lt(E_n)$ as $n \to \infty$.

We need the following facts:

$E[\phi_n] = p$; $\quad E[1 - \phi_n] = 1-p$;

$E[\Delta_i(p_n)] = E[E[\Delta_i(p_n)|p_n]]$, by a well-known theorem of probability theory (see for example Gnedenko, 1962);

so $E[\Delta_i(p_n)] = E[\theta(1 - p_n)]$

$= \theta(1 - E_n)$.

Similarly,

$E[\Delta_d(p_n)] = -\theta E_n$.

Hence (5.8) becomes, in terms of expectations,

$$E_n = E_{n-1} + \theta p(1 - E_{n-1}) - \theta(1-p)E_{n-1}$$

$$= (1-\theta)E_{n-1} + \theta p.$$

Also,

$$E_0 = p_0.$$

Hence

$$E_n = (1-\theta)^n.p_0 + p(1 - (1-\theta)^n),$$

so $E = p$.

This result is exactly the same as that obtained from the conventional EWPA procedure. For the variance of the p's, we use

$$p_n^2 = p_{n-1}^2 + \{\Delta_i(p_{n-1})\}^2.\phi_n^2 + \{\Delta_d(p_{n-1})\}^2.(1-\phi_n)^2$$
$$+ 2p_{n-1}\{\Delta_i(p_{n-1}).\phi_n + \Delta_d(p_{n-1}).(1-\phi_n)\}$$
$$+ 2.\Delta_i(p_{n-1}).\Delta_d(p_{n-1}).\phi_n.(1-\phi_n) \quad \ldots (5.9)$$

Write

$$\chi_n = E[p_n^2]; \quad \chi = Lt(\chi_n) \text{ as } n \to \infty;$$

these are assumed to exist. We need the following:

$$E[\phi_n^2] = p; \quad E[(1-\phi_n)^2] = 1-p; \quad E[\phi_n.(1-\phi_n)] = 0;$$

$$E[(\Delta_i(p_n))^2] = \theta^2.(1-E_n); \quad E[(\Delta_d(p_n))^2] = \theta^2.E_n;$$

$$E[p_n.\Delta_i(p_n)] = E[\theta p_n(1-p_n)] = \theta(E_n - \chi_n);$$

$$E[p_n.\Delta_d(p_n)] = E[-\theta.p_n^2] = -\theta\chi_n.$$

In terms of expectations, (5.9) becomes

$$\chi_n = \chi_{n-1} + p\theta^2(1-E_{n-1}) + \theta^2(1-p).E_{n-1}$$
$$+ 2p\theta(E_{n-1} - \chi_{n-1}) - 2(1-p)\theta.\chi_{n-1}$$
$$= (1-2\theta)\chi_{n-1} + \theta(\theta + 2p(1-\theta)).E_{n-1} + p\theta^2.$$

Hence

$$\chi = (1-2\theta)\chi + \theta(\theta + 2p(1-\theta))p + p\theta^2,$$

so $\chi = p(\theta + p - p\theta)$.

Thus the limiting variance of p_n as $n \to \infty$ is

$$\chi - E^2 = \theta p(1-p).$$

Note that this is slightly larger than the variance

$$\frac{\theta}{2-\theta}.p(1-p)$$

of the conventional EWPA procedure. This is only to be expected since the additional random element in the Δ-functions introduces a new degree of possible variation, so to speak. However for small θ the

difference in the variances is negligible, and these probabilistic incrementation and decrementation techniques provide an acceptable method for overcoming difficulties arising from storage limitations.

Chapter 6

THE INDEPENDENCE ASSUMPTION

The single failing of the maximum likelihood decision strategy (in its usual form) is that the independence assumption is almost never true. While wandering through numerous papers on this decision strategy and associated topics in preparation for this thesis, I was forcibly struck by the lack of space devoted to this assumption and its implications. Although passing references are made to the difficulty at several places in the literature (see for example Minsky, 1961; Nagy, 1967), the problem is usually dismissed in a sentence or two. Having given the subject considerable thought, I am now of the opinion that in fact not much can be said about it; one must accept the assumption if one wishes to build a practical machine embodying a maximum likelihood decision strategy, and features should be chosen in such a way as to help the decision. For this reason the present chapter is cautionary rather than constructive, and a large part is devoted to an examination of some implications of the independence assumption.

It seems that the problems of implementing a maximum likelihood decision scheme using a weaker form of the independence assumption are almost insurmountable. Lewis (1959) developed a method of successively approximating to probability distributions; his first-order approximation is equivalent to assuming independence, and higher-order approximations are given which refine the estimate of the probability distribution at the expense of increased storage

requirements and considerable additional complexity. As far as I know, however, these higher-order approximations have never been embodied in practical machines. Lewis himself, when simulating an experimental pattern classification machine, used only his first-order approximation (Lewis, 1962), and it is instructive to examine his reasons for doing so:

> It [the independence assumption] was made . . . because
> 1) the assumption yields a simple realization for the recognition system,
> 2) there are a great many situations for which such an assumption is adequate,
> 3) a study of this simple case furnishes a first step in the study of more general situations.

(1) appears to me to be the most binding consideration as far as designers of practical machines are concerned. Note that in (2), Lewis claims only that the independence assumption is adequate, rather than valid, for many situations. It is shown below that the assumption is valid only for a greatly restricted class of situations; unfortunately its adequacy is rather more difficult to investigate, and the evidence for (2) is presumably that maximum likelihood classifiers using the independence assumption have been shown to function reasonably well. Concerning (3), the conspicuous absence in the literature of investigations of more general situations seems to indicate the difficulty of implementing higher-order approximations.

It is possible to implement weaker forms of the independence assumption in an *ad hoc* manner depending on the particular features used. An example of this is provided by Chow (1962), who simulated a machine for recognition of hand-printed characters using the binary matrix representation of a character as its feature vector. Approximate size normalization and registration had already been

performed. Chow assumed a "nearest-neighbour" dependence:

$$\Pr[\Phi|F^{(k)}] = \Pi \Pr[\phi_{ij}|\phi_{i,j-1};\ \phi_{i-1,j};\ F^{(k)}],$$

where

ϕ_{ij} = ⌜the (i,j)'th matrix square is occupied by part of the character⌝.

This form of dependence is justified by intuition and depends strongly on the particular kind of features used, and on the ordering of features. The dependence is on the north and west neighbours: the other two neighbours are not explicitly needed. The number of probabilities to be stored is about four times that required if the usual independence assumption is used. Recognition was achieved with 97% success (on previously seen samples taken from the set of ten numerals), which represents a considerable improvement over the 80% success achieved using the usual independence assumption.

We next look at some of the implications of the independence assumption. Our starting-point is the following definition of independence:

The events E_1 and E_2 are *independent* if $\Pr[E_1|E_2] = \Pr[E_1]$;

The events E_1 and E_2 are *independent with respect to the event A* if $\Pr[E_1|E_2 \ \& \ A] = \Pr[E_1|A]$.

Thus E_1 and E_2 are independent w.r.t. (with respect to) A if given A, E_2 tells us nothing further about E_1. It is easy to show that this implies that E_2 and E_1 are independent w.r.t. A, as it should. We deal for the moment with only two events for the sake of simplicity: all results generalize easily.

When using independence to simplify the maximum likelihood decision scheme, one assumes that

$$\Pr[\phi_1 \ \& \ \phi_2|F] = \Pr[\phi_1|F].\Pr[\phi_2|F]. \qquad \ldots (6.1)$$

This is exactly equivalent to assuming that

$$\phi_1 \text{ and } \phi_2 \text{ are independent w.r.t. F.} \qquad \ldots (6.2)$$

For, (6.1) is true if and only if

$$\Pr[\phi_1|F].\Pr[\phi_2|F] = \Pr[\phi_1|\phi_2 \ \& \ F].\Pr[\phi_2|F],$$

which is equivalent to (6.2) (provided $\Pr[\phi_2|F] \neq 0$, which must be true if $\Pr[\phi_1|\phi_2 \ \& \ F]$ is to have meaning).

Let us examine a simple situation where the independence assumption is not true. Consider the features given by two rectangular Cartesian co-ordinates; let A be the set of points (see Figure 6.1)

$$A = \{(0,0) \ , \ (0,1) \ , \ (1,1)\},$$

where the points are encountered with equal frequencies, i.e.

$$\Pr[\phi_1=0, \ \phi_2=0|A] = 1/3,$$

and similarly for the other points in A. (The event $(\phi_1,\phi_2)\varepsilon A$ is denoted by A where this will cause no confusion.)

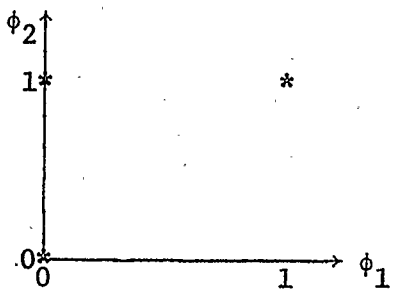

Figure 6.1

A situation where the independence assumption is not true.

Then

$$\Pr[\phi_1=0|A] = \frac{\text{number of points in A with } \phi_1=0}{\text{number of points in A}} = 2/3,$$

and $\Pr[\phi_2=0|A] = 1/3.$

Independence implies

$$\Pr[\phi_1=0,\ \phi_2=0|A] = \Pr[\phi_1=0|A].\Pr[\phi_2=0|A]$$
$$= (2/3).(1/3) = 2/9,$$

and this is not true.

Note that

$$\Pr[\phi_1=0|\phi_2=0\ \&\ A] = 1,$$

so $\Pr[\phi_1=0,\ \phi_2=0|A] = \Pr[\phi_1=0|\phi_2=0\ \&\ A].\Pr[\phi_2=0|A]$
$$= 1.(1/3) = 1/3,$$

which is as it should be.

Suppose now that we have another pattern class B which contains the point (1,0), and such that

$$\Pr[\phi_1=1,\ \phi_2=0|B] < 1/9.$$

This is easily accomplished by taking

$$B = \{(1,0)\ ,\ (2,0)\ ,\ (3,0)\ ,\ \ldots\ ,\ (10,0)\}.$$

Then the features are independent w.r.t. B, and

$$\Pr[\phi_1=1,\ \phi_2=0|B] = 1/10.$$

If we assume independence for A, the point (1,0) will be categorized wrongly, since

$$\Pr[\phi_1=1,\ \phi_2=0|A] = \Pr[\phi_1=1|A].\Pr[\phi_2=0|A]$$
$$= (1/3).(1/3) = 1/9$$
$$> \Pr[\phi_1=1,\ \phi_2=0|B].$$

Note that if we do not assume independence for A, then

$$\Pr[\phi_1=1,\ \phi_2=0|A] = \Pr[\phi_1=1|\phi_2=0\ \&\ A].\Pr[\phi_2=0|A] = 0,$$

since no point $(\phi_1,\phi_2)\varepsilon A$ with $\phi_2=0$ has $\phi_1=1$.

Whether or not the independence assumption holds for a pattern class F depends on the frequencies with which points of F are encountered. This can be seen by considering the set of points

$A = \{(0,0)\ ,\ (0,1)\ ,\ (1,0)\ ,\ (1,1)\}$.

If all the points in A have equal frequencies, then the independence assumption clearly holds. However, if the frequencies are

0.25 , 0.25 , 0.49 , 0.01

respectively, then

$\Pr[\phi_1=1|A] = 0.50$,

$\Pr[\phi_2=1|A] = 0.26$,

so $0.01 = \Pr[\phi_1=1,\ \phi_2=1|A] \neq \Pr[\phi_1=1|A].\Pr[\phi_2=1|A] = 0.13$,

so the independence assumption does not hold. Thus a pattern class F may be said to consist of a set A of points, each with frequencies attached, where the set A and the frequencies are defined by

$A = \{X|\Pr[\Phi=X|F]>0\}$;

percentage frequency$(X) = 100.\Pr[\Phi=X|F]$ for all $X\varepsilon A$.

The next result gives a characteristic which must be possessed by all sets of points corresponding to pattern classes which satisfy the independence assumption, irrespective of the associated frequencies. It is proven here for the case where the set of points is discrete; the extension to the continuous case can be shown similarly but is not relevant to this thesis.

Let F be a pattern class which satisfies the independence assumption, and denote by A its associated set of points as defined above. Suppose the query vectors are n-dimensional and discrete. We assume that all points in A have integral (not necessarily binary)

co-ordinates--this is easily arranged since the feature space is discrete. Let A_i be the projection of A on the i'th feature axis, i.e.

$$A_i = \{x \mid \text{there exist } x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \text{ with } (x_1, x_2, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \in A\}, \quad 1 \leq i \leq n.$$

By the definition of A,

$$\Pr[\phi_i = x_i \text{ for all } i,\ 1 \leq i \leq n \mid \Phi \in F] > 0 \text{ if and only if } (x_1, x_2, \ldots, x_n) \in A.$$

Also,

$$\Pr[\phi_i = x_i \mid \Phi \in F] > 0 \quad \text{if and only if} \quad x_i \in A_i.$$

The independence assumption for F is

$$\Pr[\phi_i = x_i \mid \phi_j = x_j \text{ for all } j \in B, \ \& \ \Phi \in F] = \Pr[\phi_i = x_i \mid \Phi \in F],$$

for any subset B of $\{1, 2, \ldots, i-1, i+1, \ldots, n\}$ and all i. Note that this is stronger than merely *pairwise* independence of the co-ordinates of Φ w.r.t. the event $\Phi \in F$. The independence assumption for F implies that

$$\begin{aligned} &\Pr[\phi_i = x_i \text{ for all } i,\ 1 \leq i \leq n \mid \Phi \in F] \\ &\quad = \prod \Pr[\phi_i = x_i \mid \phi_j = x_j \text{ for all } j,\ i < j \leq n, \ \& \ \Phi \in F] \\ &\quad = \prod \Pr[\phi_i = x_i \mid \Phi \in F]. \end{aligned}$$

Hence

$$\begin{aligned} (x_1, x_2, \ldots, x_n) \in A \quad &\text{if and only if} \quad \prod \Pr[\phi_i = x_i \mid \Phi \in F] > 0 \\ &\text{if and only if} \quad \Pr[\phi_i = x_i \mid \Phi \in F] > 0 \text{ for all } i \\ &\text{if and only if} \quad x_i \in A_i \text{ for all } i. \end{aligned}$$

Hence

$$A = \{(x_1, x_2, \ldots, x_n) \mid x_i \in A_i \text{ for all } i,\ 1 \leq i \leq n\};$$

that is, A is exactly the Cartesian product of its projections on each of the axes. The best way I can describe figures satisfying this restriction is to call them striated rectangular figures, with

striations parallel to the feature axes. An example in 2-space is given in Figure 6.2.

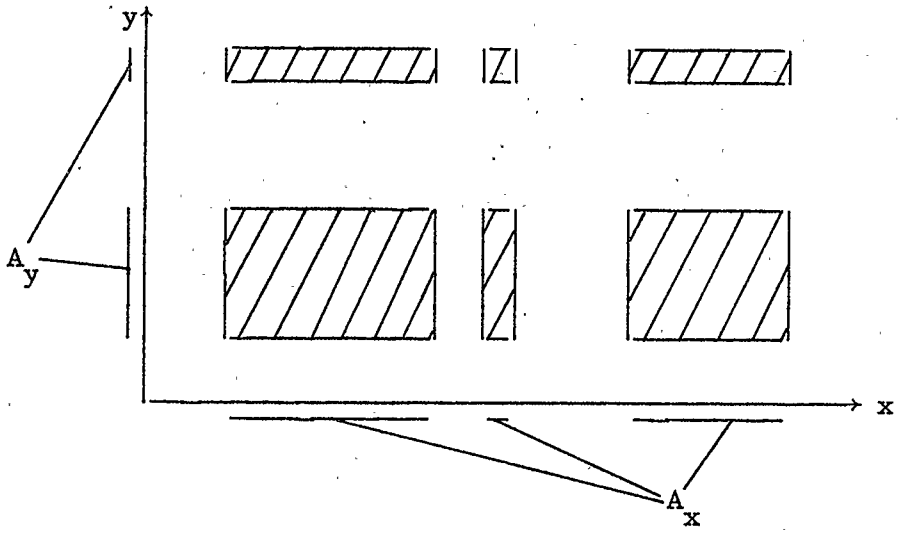

Figure 6.2

An example of a striated rectangular figure.

We have shown that if a pattern class satisfies the independence assumption, then its associated set of points must have the form of a striated rectangular figure, as described above. It is easy to see that if a pattern class F does not satisfy the independence assumption, and its associated set of points does not have this form, then a maximum likelihood classification scheme which assumes independence will assign all points in the smallest striated rectangular figure containing A to F with a non-zero probability. For, suppose there exists a point in A, for each i, with i-component x_i. Then

$$\Pr[\phi_i = x_i \mid \Phi \in F] > 0 \quad \text{for all } i,$$

so $\Pr[\Phi = (x_1, x_2, \ldots, x_n) \mid \Phi \in F] > 0.$

Hence there is a sort of generalization of the pattern class to the

smallest striated rectangular figure containing A. An example is given in Figure 6.3.

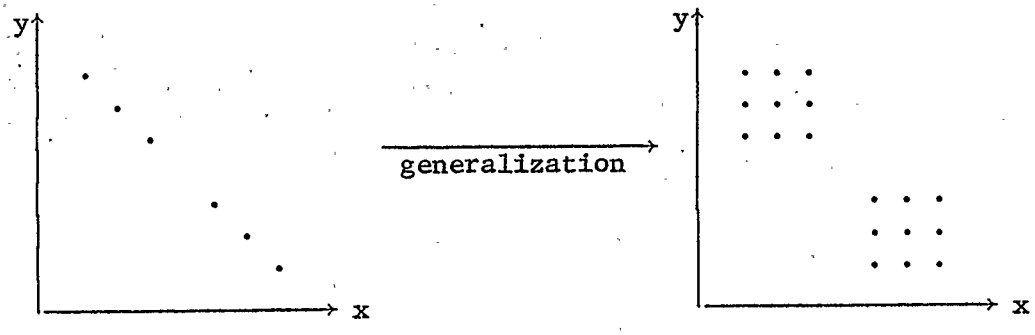

Figure 6.3

Generalization resulting from invalidity of the independence assumption.

A further consequence of the independence assumption is that if it is really true, and if the pattern classes are separated (i.e. do not intersect), then a maximum likelihood decision can be accomplished by a form of exact matching of the query vector with templates associated with the pattern classes. This is shown for the case where the features are binary-valued; the extension to many-valued features is obvious. Using the terminology of Chapter 4, and assuming independence,

$$\Pr[F|\Phi] = \frac{\Pr[F]}{\Pr[\Phi]} \cdot \prod p_j^{(\phi_j=1)} \cdot (1 - p_j)^{(\phi_j=0)} .$$

Now $\Pr[F|\Phi] > 0$ if and only if $\Phi \epsilon F$,

since the classes are separated. Take any particular $\Phi \epsilon \overline{F}$, the complement of F.

$$\Pr[F|\Phi] = 0,$$

so there exists j such that

$$p_j^{(\phi_j=1)} \cdot (1 - p_j)^{(\phi_j=0)} = 0.$$

Hence either $p_j=0$ and $\phi_j=1$ or $p_j=1$ and $\phi_j=0$. In either case, it follows that the j'th bit (feature) is the same for all query vectors in F. Define

$$f = \{j | \phi_j \text{ is the same for all } \Phi \varepsilon F\}, \text{ a non-empty set.}$$

Take any particular $\Phi^* \varepsilon F$. Then

$$\Phi \varepsilon F \quad \text{if and only if} \quad \Pr[F|\Phi] > 0$$

$$\text{if and only if} \quad \phi_j = \phi_j^* \text{ for all } j \varepsilon f.$$

Hence one can tell if a query vector Φ is in any particular pattern class by exact matching of certain bits (depending, of course, on the pattern class) of the vector Φ with any representative of that class. All that need be stored is a Φ in that class, together with pointers to the bits of Φ which are important for that class.

In the light of the above discussion, it is evident that the independence assumption is actually valid only for an extremely restricted class of situations. Fortunately it is adequate for pattern classification purposes in a rather larger class of situations. Each pattern class can be viewed as competing for any given query vector Φ--we assign Φ to the class $F^{(j)}$ which maximizes $\Pr[F^{(j)}|\Phi]$, without requiring that

$$\Pr[F^{(j)}|\Phi] > 0 \quad \text{and} \quad \Pr[F^{(i)}|\Phi] = 0 \quad \text{for all } i \neq j.$$

I have found no general way of examining the adequacy of the assumption except by experiment.

At the beginning of this chapter it was mentioned that since one virtually has to accept the rather dubious independence assumption to implement a maximum likelihood decision, the features should be

chosen in such a way as to help the decision. Although a discussion of the feature extraction process is beyond the scope of this thesis, a brief consideration of the important special case where a binary encoding of measurements is used to provide binary features is in order here, since this encoding process can affect the independence property. The next result shows by an example that independence can be lost just by encoding numerical measurements into the usual positional binary notation. It is interesting to recall that encoding features into the positional binary notation actually enhances the possibility of linear separability (see Chapter 2).

Let A be the set

$A = \{(0,1) , (0,2)\},$

and let F be the pattern class associated with A, where the points in A are encountered with equal frequency. Then F evidently satisfies the independence assumption. Consider

$A' = \{(0,0,0,1) , (0,0,1,0)\},$

obtained from A by a two-bit positional binary encoding of the features. Let F' be the pattern class associated with A'.

Then

$Pr[\phi_3=1 | \Phi \varepsilon F'] = 1/2,$

but $Pr[\phi_3=1 | \phi_4=0 \ \& \ \Phi \varepsilon F'] = 1.$

Hence F' does not satisfy the independence assumption.

Andreae (1969) discusses a situation where the feature extraction process consists merely of coding the points of a 10 × 10 matrix. Although he is primarily concerned with STeLLA-like decisions (see Chapter 8), I feel that his remarks are also relevant to maximum likelihood decisions. He suggests that the kind of input encoding

scheme used is vital to the performance of an adaptive pattern classifier, and recommends the use of a SIG (snake in the grass) code (see Figure 6.4). For Andreae's particular situation, each point is

Number	Code
1	00000
2	00001
3	00011
4	00111
5	01111
6	11111
7	11110
8	11100
9	11000
10	10000

Figure 6.4

A SIG code.

represented by ordinary rectangular Cartesian co-ordinates, each converted into the corresponding SIG code. This, he points out, has the advantage of preserving environmental continuity.

> The input coding [see above] . . . is particularly helpful to the machine because it reflects the natural topology of the environment. The Hamming distance (number of digits in opposite state) between code words reflects quite accurately the proximity of the lattice points in the 2-dimensional input space.

He goes on to compare this with a positional binary encoding scheme which, of course, preserves environmental continuity only to a very limited extent, if at all.

SIG coding also has the effect of introducing a high degree of redundancy. This is vital in noisy situations, but has disastrous implications for the independence assumption when little or no noise is present. Nevertheless, I believe that the adequacy of the

independence assumption will not be greatly impaired by the use of this kind of redundant input coding, although its validity will certainly be destroyed. Unfortunately I can offer no concrete evidence for this conjecture.

Chapter 7

SOME EXPERIMENTS WHICH ILLUSTRATE THE DIFFERENCES BETWEEN THE PERCEPTRON AND MAXIMUM LIKELIHOOD DECISION STRATEGIES

It was mentioned in Chapter 1 that few attempts to compare the performances of different decision strategies have been reported in the literature. I feel that the following reasons account, at least in part, for this regrettable fact:

1) Adaptive decision techniques are specifically intended for situations where incomplete and possibly unreliable information is available to the decision taker. Hence if comparison is attempted using environments taken from real life situations, these environments are unsuitable for well-controlled experiments, especially when reason (2) is considered. If, on the other hand, abstract environments are used, one runs the very real risk of "favouritism"--see (3) and (4).

2) While the emphasis in adaptive machine design is usually on economical hardware realization, machines are usually simulated by digital computer in the experimental stage. This simulation is rather costly in terms of computer time.

3) Different decision strategies have different characteristics which render fair comparison difficult.

4) There is no way of grading or comparing the complexity of environments except by way of the decision strategies which are to be evaluated.

At first sight, (3) looks a little out of place since after all, decision strategies are designed to perform roughly the same tasks.

For me, one of the main benefits which came from attempting an experimental comparison of decision strategies was that I was forced to consider their different characteristics in order to make the comparison fair. This is one of the chief interests of the present chapter, and will become apparent in the following pages.

Normally when one tests a pattern-classifying machine, it is the particular features used that are under test, rather than the decision strategy itself. Such tests are reported fairly well in the literature, although comparisons of the effect of different features on the same data are rather more difficult to come by. (A notable exception to this is provided by Bledsoe & Bisson, 1962; in connection with this see also Chow, 1963). As mentioned in Chapter 2, the performance of different decision strategies will depend critically on the features used; thus a comparison of decision strategies would be a gargantuan task if no analytic techniques were used. For this reason the experiments reported here are intended to illustrate some points made in previous chapters; they are presented to augment the arguments, not to carry them.

The decision data for the experiments were specially chosen in order to investigate the following phenomena:

1) the influence of threshold size on convergence time for a threshold perceptron;

2) the behaviour of a threshold perceptron in noisy conditions;

3) convergence time of a maximum likelihood classifier using the independence assumption;

4) behaviour of a maximum likelihood classifier in noisy conditions.

(1) evidently requires an environment which is linearly separable.

Moreover, for (3), the environment must be such that the independence assumption is adequate, though not necessarily valid. To investigate (2) and (4), some mechanism for introducing noise in a controllable manner is required.

The environment used is illustrated in Figure 7.1. The blank "don't care" points in the 8 × 8 two-dimensional matrix are never *presented* to the decision machine, but they may be *received* by it because of noise corruption. Features were obtained from any

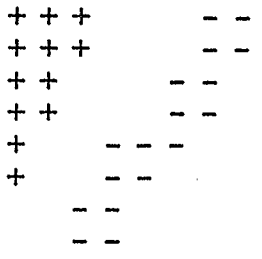

Figure 7.1

The standard environment used in all experiments.

particular point by a positional binary encoding of the two rectangular co-ordinates, giving a 6-component binary feature vector. A further component was added to each query vector Φ to obtain the augmented query vector Φ': this component was always 1. Using these features, the two pattern classes F^+ and F^- are linearly separable. In addition, the independence assumption, while not valid, was adequate for discrimination between the classes.

Noise was added to the environment by corrupting the components of each query vector with a specified probability in an independent manner thus:

Replace ϕ_i by $1-\phi_i$ with probability p ($1 \leq i \leq 6$).

The last component of each augmented query vector was never corrupted, since its use is merely a notational trick to simplify writing and programming. The probability p is referred to as the *noise level* and is expressed as a percentage. Thus if the noise level is 10%, the probability that any particular query vector is not corrupted is

$(1 - (1/10))^6 = 0.53$.

For all experiments, points in $F^+ \cup F^-$ were chosen at random. The process of choosing and corrupting a query vector can be stated as follows:

START: Select a point P at random from the 8 × 8 matrix;
If P is not classified as + or - then go to START;
If P is classified as + then TYPE = +, else TYPE = -;
Compute the uncorrupted binary Φ'-vector from the co-ordinates of P;
For i = 1 step 1 until 6 then
replace ϕ_i by $1-\phi_i$ with probability (noise level/100);
Present the corrupted query vector to the decision strategy for recognition, together with TYPE.

Figure 7.2 shows the relationship between convergence time for a threshold perceptron and threshold size, when no noise is present. The vertical axis indicates the number of mistakes made before convergence was reached. The initial weight vector for the perceptron was randomly chosen with length 1. All query vectors were normalized to length 1 before being added to the weight vector. For each threshold value the perceptron was run ten times, each run being terminated when convergence was reached. Each run took place with a

different initial weight vector, and with the pseudo-random number generator in a different state (this ensured that the sequence of points examined was different for each run). Figure 7.2 shows the mean number of mistakes before convergence, plotted against the

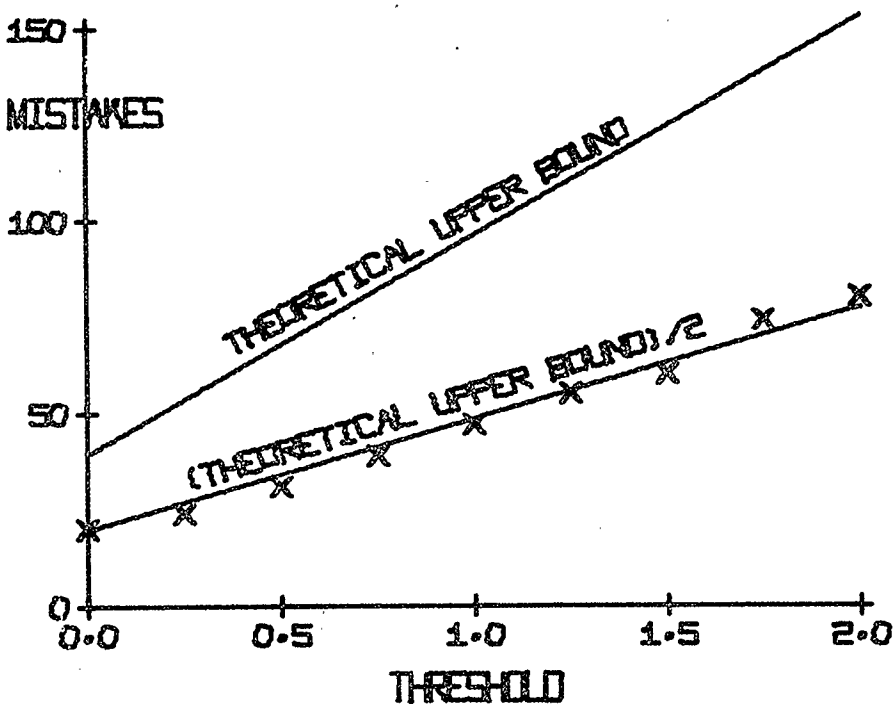

Figure 7.2

Convergence time for a threshold perceptron.

threshold. Larger threshold values were not used because of the amount of computing time required, but a smaller number of runs with thresholds 3 and 4 indicated that the linear relationship continues at least up to threshold 4. The theoretical upper bound shown is calculated from the formula obtained in the proof of the convergence theorem for the threshold perceptron (Theorem 2):

$$n = \frac{1 + 2d + 2\delta}{\delta^2} .$$

The value of δ which was used was the maximum δ found in the simulation runs:

$$\delta = 0.187 .$$

The results obtained in the simulation are strikingly close to half the theoretical upper bound--a further line is given in Figure 7.2 to emphasize this.

In Figure 7.3 the ranges of convergence times are shown for perceptrons with various thresholds and for the maximum likelihood (independence assumed) classifier. Note that the vertical axis gives the total number of cycles to convergence, and not merely the number of misclassified points as in Figure 7.2. Ten simulation runs were made for each method, and the maximum and minimum convergence times were deleted in an attempt to eliminate exceptional cases. The range between the maximum and minimum of the amended set is shown. The results are not very reliable--they depend rather critically on the particular sequence of points used for adaptation. Nevertheless, it can be seen that the maximum likelihood classifier can be expected to converge in roughly the same length of time as a perceptron with zero threshold, for this particular environment.

The question of convergence time in noisy conditions now arises. We provisionally define this to be the time taken for the adaptive machine to reach a state where it correctly classifies all noiseless points, since clearly the machine cannot be expected to classify all noisy points correctly. Certain types of decision scheme, however, improve their performance after they reach the stage where

they can correctly classify all noiseless points--the maximum likelihood decision strategy is an example of this. Hence convergence time as defined above is not necessarily an indication of the length of training period required for a classification machine if noise is present.

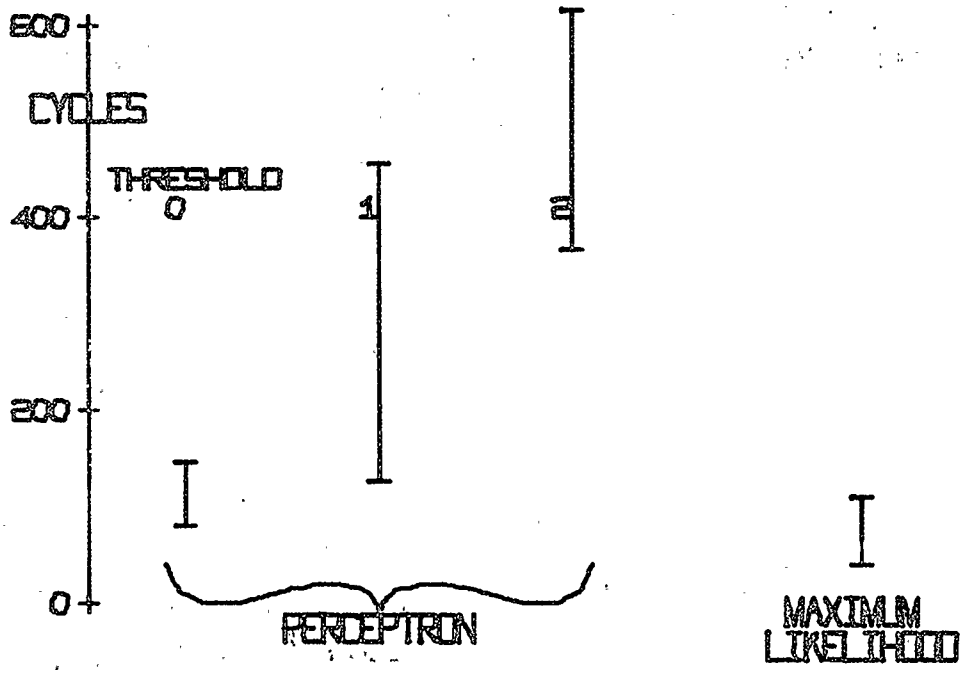

Figure 7.3

Variations in convergence time for various pattern classifying methods with a standard noiseless environment.

In cases where the details of the environment and the noise statistics are known, the optimum performance of a maximum likelihood classifier (both with and without the independence assumption) can be calculated, and one useful measure of the length of training period

required for this type of decision strategy is the mean time taken to reach this optimum performance. This is discussed later.

For perceptron-like machines, which do not eventually settle down to a fairly constant performance level, convergence time as defined above has little meaning, since the performance may deteriorate considerably after the machine has "converged". I can propose no satisfactory measure of convergence time for perceptron-like machines in noisy conditions, although clearly the confusing effect of noise means that the higher the noise level the longer the training period required. Some rather inconclusive experiments were made to investigate the expected time taken by perceptrons to reach a state which correctly classifies all noise-free points: it was found that the times taken on different occasions under the same conditions varied so much that the results were rather meaningless. For these reasons, the following investigations of the perceptron's performance in noisy conditions were conducted after the perceptron had converged in conditions of no noise, since experience of the noise during convergence would not have helped the perceptron.

In order to compare the performances of threshold perceptrons with different thresholds in noisy conditions, points corrupted by noise were ignored in the success count for the perceptron. These points had an implicit influence on the success count because adaptation continued throughout the experiments, and all misclassified points--noisy or not--modified the weight vector. The effect of the perceptron's futile attempts to adapt itself to the noisy points was sought, and since it cannot be expected to classify noisy points correctly (except by chance), the inclusion of these in the frequency

count would only cloud the issue. Figure 7.4 shows the percentage misclassification for points which were uncorrupted by noise, plotted against threshold values, for various noise levels. As predicted in

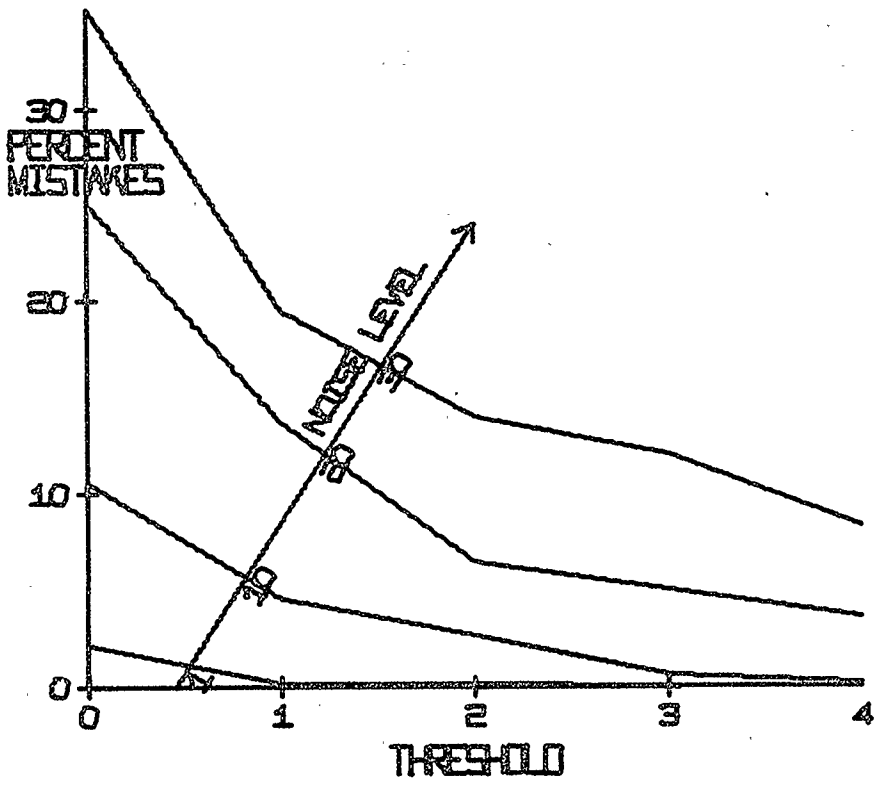

Figure 7.4

Success of a threshold perceptron in noisy conditions.

Chapter 3, misclassification of noise-free points can be almost completely eliminated by using a perceptron with a threshold of three or four times the length of the query vectors (the query vectors in these simulations were normalized to length 1), provided the noise level is low. If the noise level is high, in our case greater than

about 15%, it is much more difficult to eliminate the perturbing effect of noise on the weight vector. It should be pointed out that no decision strategy works well in conditions where a large amount of noise is present--one of the purposes of feature extraction is to reduce the amount of noise passed to the decision stage--and so, in normal conditions, it will be highly beneficial to use a threshold perceptron rather than one of the basic variety.

One of the advantages of using an artificial environment with controlled noise is that it is possible to determine the expected performance of a true maximum likelihood (not assuming independence) classifier by exact calculation. To do this, one first determines the machine's correct strategy for classification of each query vector Φ, using the relation

$$\Pr[F^{(i)}|\Phi] = \Sigma\, f(\Phi',\Phi).\lceil \Phi' \varepsilon F^{(i)} \rceil,$$

where the sum is taken over all possible vectors Φ', and

$f(\Phi',\Phi)$ = Pr[Φ' is transformed to Φ by noise corruption].

$f(\Phi',\Phi)$ can be calculated using the number of unlike bits of Φ' and Φ, and the noise level. The machine's correct strategy for classifying a query vector is now known, *viz.* choose i such that $\Pr[F^{(i)}|\Phi]$ is greatest. Naturally the "don't care" class (blank points in Figure 7.1) is never chosen, although its probability may be greater than the others. For low noise levels (up to 40% in fact), the machine's correct strategy differs from the true dichotomy, as in Figure 7.1, only in the classification of the "don't care" points. Having determined this optimum classification, the probability of a mistake being made can be calculated:

$$\Pr[\text{mistake}] = \Sigma_{\Phi'} \{\Pr[\Phi' \text{ chosen}] \cdot \Sigma_{\Phi} \{f(\Phi',\Phi) \cdot [\text{true classification of } \Phi' \neq \text{machine's classification of } \Phi]\}\},$$

where the summations are taken over Φ' and Φ respectively.

This probability of error is plotted in Figure 7.5 as line E.

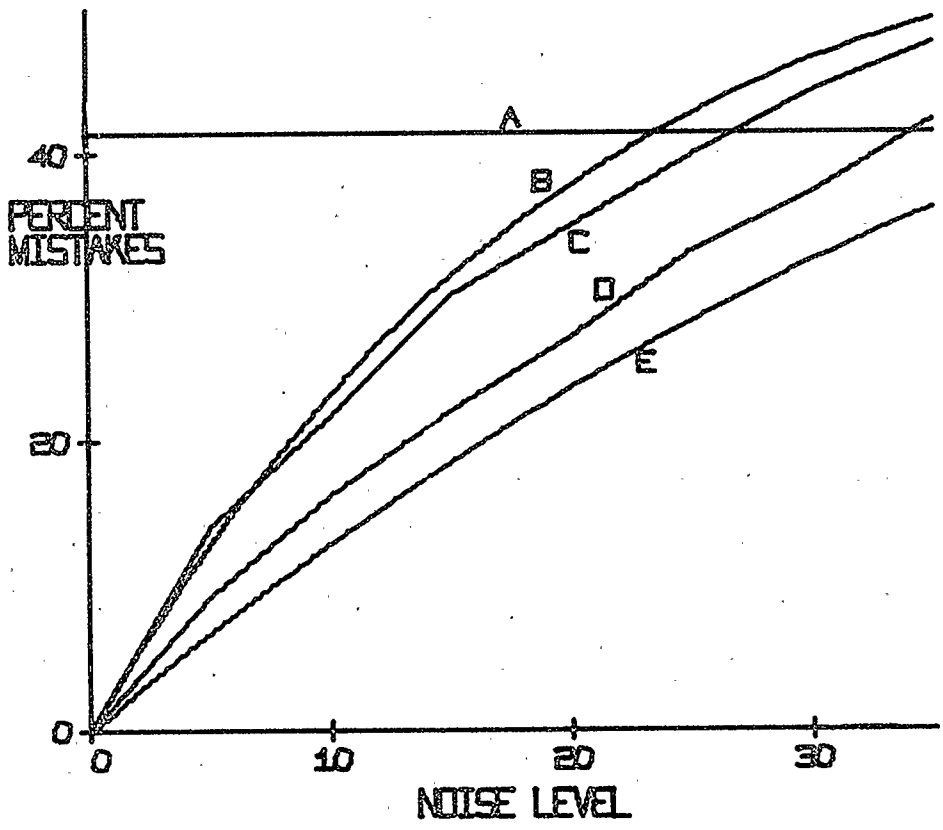

Figure 7.5

Percentage error for different decision strategies with varying noise levels.

Line B shows the probability of error if all noise-free points are classified correctly, but all "don't care" points in Figure 7.1 are given the class opposite to that dictated by the optimal policy. This

provides an upper bound to the number of mistakes made by machines which correctly classify all noiseless points. A further decision strategy is always to decide the class which has the highest *a priori* probability, without reference to the query vector. The percentage of mistakes made if this strategy is used is unaffected by noise; it is shown as A in Figure 7.5.

Lines C and D in Figure 7.5 show the performances achieved experimentally by perceptrons with thresholds 0 and 4 respectively. These simulations took place under the same conditions as those reported earlier, except that the percentage probability of any mistake is given, rather than the percentage probability of a mistake being made on a point uncorrupted by noise.

Figure 7.6 is a distortion of Figure 7.5 designed to show more clearly both the relationships between the lines, and the perturbations in lines C and D which give some indication of the experimental error. It is obtained as follows: in 100 trials, decision strategy A (for example) will make A(N) mistakes at noise level N, where

$$y = A(x)$$

is the equation of line A in Figure 7.5. At the same noise level, E, the optimal strategy, will make E(N) mistakes. Thus there are only

$$100 - E(N)$$

trials in which A could possibly be expected to identify the query vector correctly, and of these,

$$A(N) - E(N)$$

mistakes occur. Thus

$$\frac{A(N) - E(N)}{100 - E(N)} \times 100$$

represents the percentage of needless mistakes made by decision strategy A. It is this which is plotted as A in Figure 7.6, and similarly for B, C, and D. The line obtained from E evidently coincides with the horizontal axis, as shown.

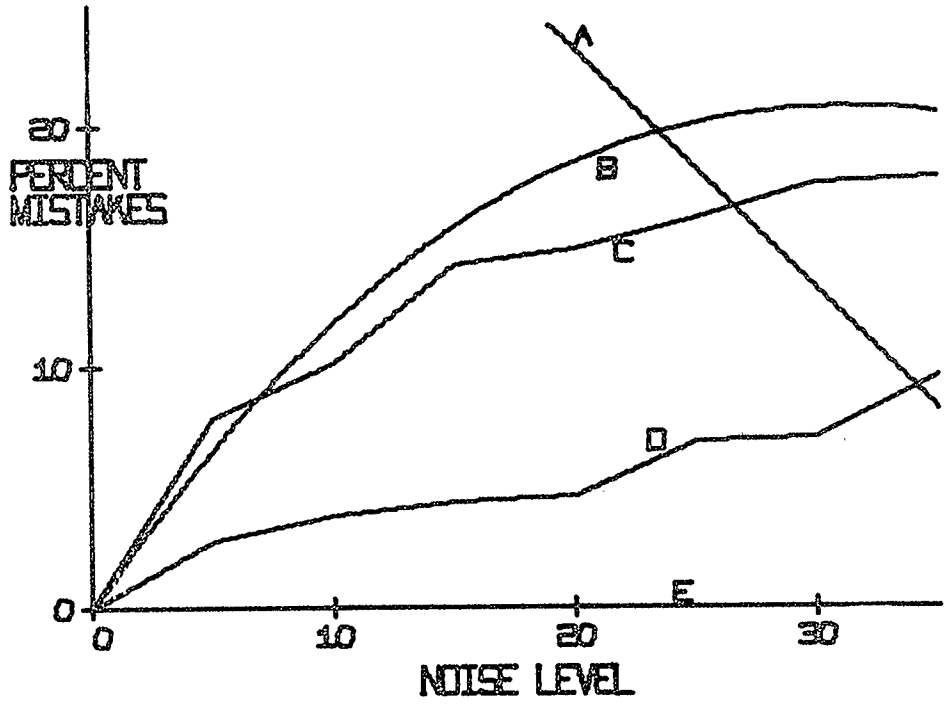

Figure 7.6

Needless errors as a percentage of points that could have been classified correctly, for different decision strategies.

One can deduce from Figure 7.6 that if the noise level is greater than about 27%, a perceptron with zero threshold performs so badly that the chance of error is reduced by ignoring the query vectors and always deciding the class whose *a priori* probability is

greatest. Although this result evidently depends on the particular environment used, it indicates rather strikingly just how badly standard perceptrons behave in noisy conditions. The vast improvement in performance which results from using a perceptron with a threshold of about 4 is also apparent.

So far no mention has been made of the performance of the usual maximum likelihood decision strategy in noisy conditions. Although the environment was chosen so that the independence assumption was adequate for discrimination between the noiseless pattern classes, the assumption is not in fact valid for this environment. It was thought that this would detract considerably from the performance of the usual maximum likelihood strategy in noisy conditions; however, this was not so.

It is possible to calculate the expected error for the usual maximum likelihood strategy in a manner similar to that described for the true maximum likelihood strategy. It was found that the difference in performance resulting from assuming independence was less than 0.2%, for all noise levels between zero and 35%. Thus an error curve for the maximum likelihood strategy with independence assumed would be almost indistinguishable from E in Figures 7.5 and 7.6.

Since probability estimation is a stable process, a practical machine embodying a maximum likelihood decision (independence assumed) will reach within an arbitrary latitude of the above expected performance level, given sufficient training time. It was found experimentally that after 700 training cycles, the error probability was within 0.8% of that indicated above; that is, a line in Figure 7.5 indicating the performance attained experimentally by a maximum

likelihood classifier with 700 training cycles would lie less than one vertical unit above E.

Thus there is no doubt that for the environment shown in Figure 7.1, a maximum likelihood strategy is superior to all perceptron-like strategies that have been considered, whether or not noise is present. It would be interesting to know if adequacy of the independence assumption for discrimination between the noiseless pattern classes guarantees near-optimal performance for the usual maximum likelihood strategy in noisy conditions, or whether the phemonemon occured here because of a lucky choice of environment. As far as I am aware, no investigation of this question has been reported.

Chapter 8

STeLLA-LIKE DECISION TECHNIQUES

8.1 Introduction.

We have seen that neither the perceptron nor the maximum likelihood (with the independence assumption) decision strategy can be described as a general purpose classification technique. Each behaves unsatisfactorily in some kinds of environment, but is very competent under certain circumstances: the perceptron strategy guarantees discrimination between patterns provided they are linearly separable but does not work well in noisy conditions, whereas the independence assumption, so vital to the implementation of the maximum likelihood decision strategy in any practical situation, almost never holds and is very often violated flagrantly enough to ruin the classification. If, however, the independence assumption is valid then the maximum likelihood decision strategy is optimal, even in noisy conditions.

This chapter is devoted to a discussion of a compromise decision strategy, one which combines the characteristics of the perceptron and maximum likelihood schemes. If W^* is such that

$$W^*.\Phi > \delta \quad \text{for all } \Phi \varepsilon F^+,$$

$$W^*.\Phi < -\delta \quad \text{for all } \Phi \varepsilon F^-;$$

and if X is a vector with

$$|X| < \delta/\alpha,$$

where α is the maximum size of the query vectors Φ; then

$$(W^* + X).\Phi > \delta - |X|.|\Phi|$$

$$> \delta - (\delta/\alpha).\alpha = 0 \quad \text{for all } \Phi \varepsilon F^+,$$

and similarly,

$$(W^* + X).\Phi < 0 \quad \text{for all } \Phi \in F^-;$$

so $(W^* + X)$ also discriminates between the pattern classes. Among these vectors which discriminate between the classes, some must behave better than others in noisy situations. The perceptron strategy is content with any vector which discriminates between the pattern classes: the decision strategy discussed here attempts to find a discriminating vector which gives good behaviour in noisy conditions. The strategy we will consider is derived from STeLLA, a learning machine whose rules were chosen on an empirical basis. It is remarkable how such rules give rise to a decision scheme with the very characteristics we seek.

STeLLA is a general purpose learning machine, described by Andreae (1964, 1969) and Gaines & Andreae (1966). She sees her environment at any one time as a binary input pattern, and selects one of a specified set of actions. The taking of this action changes the state of the environment, and the new state is reflected in a new input pattern which is presented to STeLLA. Thus by selecting various actions and observing the input patterns she attempts to build an internal model of her environment. Some input patterns are considered to be desirable states and this is communicated to STeLLA by a reward system. Her goal is to get reward as often as possible. The *control policy* is responsible for choosing the best sequence of actions with respect to the goal, and it calls on a neutral *predictor* for aid. The predictor is that part of the machine which models the behaviour of the environment, independently of rewarded states; while the control policy models the environment as it relates to goal achievement.

Adaptive decision procedures occur in several places in STeLLA, and the details of adaptation vary according to the purpose[1]. Because of this, it was decided to treat the problem in as general a manner as possible. The discriminatory functions described below are representative of the various forms of discriminatory function which STeLLA uses, and the form of the adaptive rules given is designed so that every set of rules used by her can be considered as a special case. To facilitate this, the quantitative amounts of adaptation are left unspecified as far as possible. The resulting model is a flexible tool for theoretical and experimental investigation of adaptive pattern recognition techniques, and I shall refer to it in all that follows as *the STeLLA method*, or some such term. Unfortunately the STeLLA method sacrifices elegance for generality. We have seen how the perceptron and maximum likelihood decision strategies are embedded in a precise mathematical framework: I have not been able to build such a mathematical edifice for STeLLA-like strategies, although a few bricks appear here and there in the following pages. Hence this

---

[1] For example, the predictor in more recent versions models the environment by partitioning the input patterns into sets called *clumps*, and examining the effect, in terms of reaching other clumps, of each action at each clump. The association of input patterns with clumps is accomplished using adaptive pattern classification techniques. A predicted clump is used to determine the next prediction, and, since there are two reinforcable steps here, the reinforcement is spread over the two moves. A further complication arises from the fact that both clumps and elements of the control policy are competitive in the sense that the total number of each is strictly limited. This means that unused or infrequently used clumps or policy elements will be forced out of existence to make way for new ones, and the adaptation procedures used are tailored accordingly. Unfortunately most of these details of STeLLA's operation are unpublished.

chapter takes on a rather vague and imprecise character in parts, and most solid theoretical results are applicable only to certain special cases of the decision strategy.

We consider the case where just two pattern classes, F^+ and F^-, are present; the results and methods generalize easily to multi-class cases. The discussion is restricted to query vectors with binary components--as indeed is STeLLA. The query vectors are assumed to contain a completely redundant component so that the augmented query vectors Φ' are not needed. To each pattern class is assigned a *prototype*: a query vector, generally contained in the class, which is assumed to be representative of that class. The prototype of class F^+ is denoted by P^+, and similarly for F^-. Each binary component of the prototype has an associated *pattern digit weight* (PDW) which represents the danger of overlooking a disparity between the k'th bit of a query vector Φ and the prototype's k'th component, when assigning Φ to the pattern class associated with that prototype. PDW's are constrained to lie in the interval [0,1], even if one of the adaptation rules below attempts to take the PDW out of the range.

8.2 The discriminatory functions.

The discriminatory functions used by the STeLLA method are

$$f^+(\Phi) = \prod (1 - p_k^+)^{\left(\phi_k \neq P_k^+\right)} ,$$

and similarly for f^-, where the p_k's ($1 \leq k \leq n$, where n is the dimensionality of Φ) are the appropriate PDW's. This function takes into account the dissimilarities of the prototype and the query vector. Its value is 1 if the query vector is exactly the same as the prototype. If differences exist the function takes a value less than

1, the value being smaller if many unlike components exist or if the unlike components are important ones.

There are two ways of interpreting these discriminatory functions heuristically, corresponding to situations in which the maximum likelihood strategy works well and situations in which the perceptron strategy works well. In each case, the prototype P^+ is considered to be a typical member of the class F^+.

Firstly, suppose the class F^+ consists of only one element whose components may be corrupted by noise in an independent manner. This element shall be chosen for the prototype. Let us take our null hypothesis to be that the query vector Φ belongs to F^+. Evidence against this hypothesis is provided by the bits of Φ unlike the corresponding bit of P^+, and these represent features which were obscured by noise. In this case we interpret

$$p_k^+ = \Pr[\text{feature k is not obscured by noise}],$$

and this is certainly a measure of the danger of overlooking a disparity of the k'th bit of Φ when assigning Φ to the class F^+.

The second interpretation of the discriminatory function concerns cases where there is no noise present but the class F^+ consists of many members. We describe it by means of an example: Suppose

$$P^+ = 1111 \quad (n=4),$$

and the query vector

$$\Phi = 1001$$

is presented. Since the PDW's represent the danger of neglecting the digit, we can write

$$p_1^+ = \Pr[0111 \notin F^+],$$
$$p_2^+ = \Pr[1011 \notin F^+],$$
$$p_3^+ = \Pr[1101 \notin F^+],$$
$$p_4^+ = \Pr[1110 \notin F^+].$$

Hence

$$f^+(1001) = \Pr[1011 \in F^+].\Pr[1101 \in F^+].$$

Since we know that

$$P^+ = 1111 \in F^+,$$

the discriminatory function can be interpreted as a measure of our confidence that

$$1001 \in F^+,$$

provided that the environment is "continuous" (to some degree), that is, provided that

$$1111 \in F^+, \quad 1011 \in F^+, \quad 1101 \in F^+$$

provides evidence for the proposition

$$1001 \in F^+.$$

It is suggested that this requirement of environmental continuity explains why helpful coding schemes, discussed in Chapter 6, are especially important for STeLLA-like decisions, although they would probably help most maximum likelihood schemes as well (Andreae, 1969; Gaines & Andreae, 1966).

These two heuristic interpretations, while admittedly vague and rather unsatisfactory, illustrate the compromise which was made in STeLLA between the maximum likelihood and perceptron decision procedures. As we have seen, the maximum likelihood decision is linear if binary features are used, and could in principle be used with "compromise" adaptation rules like those described below.

However, this is difficult in practice because the linear form of this decision is rather involved, and the omission of query vector components which are like the corresponding prototype components from the discriminatory functions, as above, reduces the "compromise" adaptation rules from an interesting theoretical possibility to a practical proposition. Obtaining practical approximations to ideal schemes is, after all, one of the main themes of this thesis.

8.3 Qualitative aspects of the adaptation procedure.

Although facilities exist in STeLLA for generating and adapting prototypes, they will not be discussed here. Our concern is with the generalized adaptation process for PDW's. This is governed qualitatively by the rules below, which are suggested by common sense. If a query vector $\Phi \varepsilon F^+$ is found with

$$f^+(\Phi) < e^d . f^-(\Phi),$$

where d is a non-negative constant[2], so that $e^d \geq 1$ (the exponentiation is used for later convenience), then Φ is incorrectly assigned to class F^- (or rather Φ is not assigned to class F^+ with sufficient confidence), and the adaptation rules are:

decrease PDW's of P^+ corresponding to Φ-components unlike P^+ (8.1)

increase PDW's of P^- corresponding to Φ-components unlike P^- (8.2)

If $\Phi \varepsilon F^+$ is such that

$$f^+(\Phi) > e^d . f^-(\Phi),$$

then Φ is correctly assigned to class F^+, and the rules are:

---

[2] In the STeLLA machine, this threshold was always taken to be zero. It is introduced here for generality.

decrease PDW's of P^+ corresponding to Φ-components unlike P^+ (8.3)

increase PDW's of P^+ corresponding to Φ-components like P^+ (8.4)

Similar rules are used for query vectors in F^- (change all +'s to -'s, and vice versa, in the above). Rules (8.2) and (8.3) alone were used in the adaptation of STeLLA's control policy elements.

Rules (8.1) and (8.2) are used when the query vector Φ is either misclassified or correctly classified by only a small margin. Only PDW's corresponding to Φ-components unlike the respective prototype bits are adjusted since only these PDW's affect the categorization of Φ. Rule (8.1) increases $f^+(\Phi)$; rule (8.2) decreases $f^-(\Phi)$. Rules (8.3) and (8.4) come into play only if the query vector Φ is correctly classified. Rule (8.3) generalizes on the basis of the justifiably ignored Φ-components, and incidentally strengthens the association of Φ with F^+. Rule (8.4) is intended to balance the effects of (8.3) and prevent the PDW's from constantly decreasing if no mistakes are made. Note that these last rules involve only the pattern class which contains Φ: in a sense the effect of the adaptation is local if the query vector is correctly classified (by a sufficiently large margin), whereas for incorrect classifications the rules have a global effect--they alter the values of some PDW's of all the pattern classes. This also accords with common sense, for the effect of the adaptation should be rather more drastic when mistakes are being made than when the machine is functioning correctly, when delicate adjustments are required in order to improve but not disturb the balance of weights.

The quantitative amounts of adaptation used in the STeLLA method are governed by the *increment/decrement functions*; these have

as their only argument the PDW currently under consideration. For reasons mentioned earlier, we shall deal with the rules in terms of general increment/decrement functions as far as possible, but shall rapidly be forced to specialize in many ways since the STeLLA method is mathematically rather intractable. It is as well to mention here that the problem of instability can occur; for example the machine, when started in a correctly discriminating state, may run away and end up in a situation where all PDW's are constantly bouncing off their lower bound. This cannot happen with the perceptron (provided its environment is noiseless and linearly separable) since the convergence theorem guarantees that a discriminating weight vector will be found and no adaptation takes place after such a vector is found; nor can it happen with the maximum likelihood decision (provided that the query vectors are reasonably representative) since the probability estimation procedures discussed earlier force the probabilistic weights to settle down eventually. Both these procedures, when started in a "correct" state, will retain this, with minor variations in the case of the maximum likelihood decision, indefinitely in the absence of noise. For STeLLA-like decisions, however, this is not necessarily true: stability depends on the exact form of the increment/decrement functions used. It is possible that instability could be environment-dependent--although I consider this to be rather unlikely for non-pathological environments--and if so, to suggest that any particular increment/decrement functions are "the best" or even "fairly good" would be rather presumptious. Whenever increment/decrement functions are suggested below, they are intended as plausible specializations whose purpose is to give a *lower* bound to

the power and versatility of the STeLLA method.

The increment/decrement functions are denoted as follows:

$$\left.\begin{matrix}\tilde{\Delta}_i(x) \\ \tilde{\Delta}_d(x)\end{matrix}\right\} \quad \text{for rules (8.1) and (8.2),}$$

$$\left.\begin{matrix}\Delta_i(x) \\ \Delta_d(x)\end{matrix}\right\} \quad \text{for rules (8.3) and (8.4);}$$

where the argument x is the PDW currently under consideration ($0 \leq x \leq 1$). We use the convention that the decrement functions are such that

$$\Delta_d(x) \leq 0, \qquad \tilde{\Delta}_d(x) \leq 0 \qquad \text{for all } x \ (0 \leq x \leq 1);$$

so that a decrement is added to the PDW, rather than an increment being subtracted.

8.4 Perceptron-like behaviour of the STeLLA strategy.

Any query vector Φ is assigned to class F^+ or F^- according as

$$\Pi\ (1 - p_k^+)^{(\phi_k \neq P_k^+)} \gtrless \Pi\ (1 - p_k^-)^{(\phi_k \neq P_k^-)} ,$$

i.e. according as

$$\Sigma\ \{(\phi_k \neq P_k^+).\log(1 - p_k^+) - (\phi_k \neq P_k^-).\log(1 - p_k^-)\} \gtrless 0 .$$

Unfortunately this decision is not, in general, linear in Φ. It can, however, be made linear by assuming that

$$P_k^+ = P_k^- = 0 \quad \text{(or 1)} \quad \text{for all } k \ \ (1 \leq k \leq n); \qquad \ldots \text{(A)}$$

suitable rules for prototype adaptation will ensure that if a linear dichotomy is required, this condition is eventually satisfied. Note that prototypes which satisfy (A) cannot be considered to be representative of their respective classes. The somewhat hopeful assumption is made that since non-representative prototypes are, in a sense, "unfair" to the decision strategy, better performance will be

obtained without restriction (A), so that by using this assumption we run the risk of underestimating rather than overestimating the power of the decision strategy.

The process of adaptation to successfully classified query vectors (rules (8.3) and (8.4)) is now considered to be suspended for the time being, so that we can concentrate on the machine's attempts to improve itself when confronted with query vectors which it classifies incorrectly (or which it classifies correctly but only by a small margin). For reference purposes this is called assumption (B). We also assume that there are no attempts to make out-of-range adaptations (C).

The discriminatory functions are now

$$f^{+}(\Phi) = \Pi\ (1 - p_k^{+})^{\phi_k}\ ;$$

and Φ is assigned to F^{+} or F^{-} according as

$$\Sigma\ \phi_k W_k \gtrless 0,$$

where

$$W_k = \log(1 - p_k^{+}) \ - \ \log(1 - p_k^{-})\ .$$

The adaptation rules are as follows:

Suppose a query vector $\Phi \epsilon F^{+}$ is encountered with

$$\Pi\ (1 - p_k^{+})^{\phi_k} \ < \ e^{d}.\Pi\ (1 - p_k^{-})^{\phi_k}\ ;$$

i.e. with

$$\Sigma\ \phi_k W_k < d\ . \qquad \ldots\ (8.5)$$

Then p_k^{+} and p_k^{-} are changed to $p_k^{+'}$ and $p_k^{-'}$ respectively, where

$$p_k^{+'} \ = \ p_k^{+} + \tilde{\Delta}_d(p_k^{+}).\lceil \phi_k = 1 \rceil,$$

$$p_k^{-'} \ = \ p_k^{-} + \tilde{\Delta}_i(p_k^{-}).\lceil \phi_k = 1 \rceil.$$

Now

$$\log(1 - x - \Delta(x)) = \log(1 - x) + \log\{1 - \Delta(x)/(1-x)\}$$
$$\simeq \log(1 - x) - \Delta(x)/(1-x),$$

provided

$$\Delta(x) << |1 - x|. \qquad \ldots (D)$$

We consider for the moment only increment/decrement functions which satisfy (D); it is shown later that this condition can be relaxed.

Now

$$W_k' = \log(1 - p_k^{+'}) - \log(1 - p_k^{-'})$$
$$= W_k - \{ \frac{\tilde{\Delta}_d(p_k^+)}{1 - p_k^+} - \frac{\tilde{\Delta}_i(p_k^-)}{1 - p_k^-} \}.\lceil \phi_k=1 \rceil .$$

It is clear that the following increment/decrement functions will simplify the problem considerably:

$$\tilde{\Delta}_i(x) = \mu(1-x) = -\tilde{\Delta}_d(x) , \qquad \ldots (E)$$

for some constant μ. Using these functions,

$$W_k' = W_k + 2\mu.\lceil \phi_k=1 \rceil ;$$

so $W' = W + 2\mu\Phi$. $\qquad \ldots (8.6)$

Similarly, if $\Phi \varepsilon F^-$ and

$$e^d.\Pi\ (1 - p_k^+)^{\phi_k} > \Pi\ (1 - p_k^-)^{\phi_k} ,$$

then

$$W' = W - 2\mu\Phi. \qquad \ldots (8.7)$$

Combining (8.5), (8.6), (8.7); and writing

$$G^+ = \{2\mu\Phi \,|\, \Phi \varepsilon F^+\},$$

$$G^- = \{2\mu\Phi \,|\, \Phi \varepsilon F^-\};$$

we can write the adaptation rules in this restricted case in the form of a program:

TEST: Choose $\Psi \epsilon\ G^+ \cup G^-$;

If $\Psi \epsilon G^+$ and $W.\Psi < 2\mu d$ then replace W by $W+\Psi$;

If $\Psi \epsilon G^-$ and $W.\Psi > -2\mu d$ then replace W by $W-\Psi$;

Go to TEST.

This is exactly the procedure used by a perceptron with threshold $2\mu d$. Hence we can apply Theorem 2, using as upper limit for the length of query vectors

$$\alpha = 2\mu\sqrt{n} \geq |\Psi| \quad \text{for all } \Psi \epsilon\ G^+ \cup G^- ,$$

where n is the dimensionality of the query vectors Φ:

If there exists a unit n-vector W* and some $\delta > 0$ with

$$\Phi \in F' = \{\Phi \mid \Phi \epsilon F^+\} \cup \{-\Phi \mid \Phi \epsilon F^-\} \quad \text{implies} \quad W^*.\Phi > \delta,$$

so that

$$\Psi \in G' = \{2\mu\Phi \mid \Phi \epsilon F'\} \quad \text{implies} \quad W^*.\Psi > 2\mu\delta;$$

then if W is initially chosen to be an n-vector of length λ, the above program will change W at most

$$\frac{4\mu^2 n + 4\mu d + 4\lambda\mu\delta}{4\mu^2\delta^2} = \frac{\mu n + d + \lambda\delta}{\mu\delta^2}$$

times.

Thus a STeLLA-like pattern classifier behaves like a threshold perceptron if

A) $P_k^+ = P_k^- = 0$ for all k $(1 \leq k \leq n)$;

B) no adaptations are made for correctly classified (by a large enough margin) query vectors;

C) there are no attempts to make out-of-range adaptations;

D) $\mu << 1$, so that the approximations in the expansions of $\log(1 \pm \mu)$ are valid;

E) the functions $\tilde{\Delta}_i$ and $\tilde{\Delta}_d$ have the form given.

Theorem 4 (see Appendix A) was begun in an attempt to generalize restriction (E): upper and lower bounds were sought for the function

$$\eta(x,y) = \log(1 - \frac{\tilde{\Delta}_d(x)}{1-x}) - \log(1 - \frac{\tilde{\Delta}_i(y)}{1-y}),$$

so that the theorem would remain true. It was found, however, that it was necessary to assume that the upper and lower bounds were rather close together; the allowable latitude depended on δ in such a way that the bounds must actually be equal if the theorem was to hold for all linearly separable environments. Thus

$\eta(x,y)$ = constant,

so if p_k^+ and p_k^- vary independently, which they can do if rules (8.3) and (8.4) are brought back, then the increment/decrement functions must be of the form (E). This does not of course prove that the increment/decrement functions necessarily have the form (E), it merely shows that I was unable to prove the convergence theorem without this assumption. I conjecture, however, that it is indeed true that the convergence theorem holds in general only if (E) is assumed, but I have not been able to prove this.

Two interesting corollories come out of Theorem 4: firstly that restriction (D) is not necessary, and secondly that if the components of the discriminating weight vector W* are all positive then the only condition that need be placed on η is

$\eta(x,y) > 0$ for all x,y.

This condition is always true if the increment/decrement functions satisfy

$\tilde{\Delta}_i(x) > 0$; $\tilde{\Delta}_d(x) \leqq 0$ for all x.

8.5 Use of the increment/decrement functions for probability estimation.

In order to consider the behaviour of the STeLLA pattern classification method for classifications which are correct (by a large enough margin), we can dispense with all the restrictions used above: all that is assumed is that rules (8.1) and (8.2) are never applied. The difficulty here is not the mathematical intractability of the general problem that troubled us in the last few pages; rather it is the vagueness of the probabilistic interpretation of the PDW's. It shall be assumed in the following that the PDW's are required to estimate

$$\Pr[\phi_k = P_k^+ | \Phi \varepsilon F^+],$$

but it is clear that many alternative strategies could be adopted, and as usual the increment/decrement functions given below are intended as examples rather than as recommendations.

Let us write $p_k^+(t)$ for the value of the k'th PDW of class F^+ at time t: suppose that rules (8.1) and (8.2) are never applied at times $t \geq 0$. If Φ is correctly identified in F^+ (by a sufficiently large margin) at time t, then according to the increment/decrement rules,

$$p_k^+(t+1) = p_k^+(t) + \Delta_i(p_k^+(t)).\lceil \phi_k = P_k^+ \rceil + \Delta_d(p_k^+(t)).\lceil \phi_k \neq P_k^+ \rceil .$$

Let $p = \Pr[\phi_k = P_k^+ | \Phi \varepsilon F^+]$; $q = 1 - p$.

Suppose

$$p_k^+(t) = p,$$

i.e. the value of p_k^+ at time t equals the quantity which p_k^+ is required to estimate. Then the expected value (over all Φ's) of $p_k^+(t+1)$ should

be the same, namely p. Hence

$$p = E[p_k^+(t+1)] = p_k^+(t) + p.\Delta_i(p_k^+(t)) + q.\Delta_d(p_k^+(t))$$
$$= p + p.\Delta_i(p) + q.\Delta_d(p) ;$$

where E denotes statistical expectation. (This is not strictly true. The expectation should be taken over all Φ such that

$$\Pi\ (1 - p_k^+)^{\lceil \phi_k \neq P_k^+ \rceil} > e^d.\Pi\ (1 - p_k^-)^{\lceil \phi_k \neq P_k^- \rceil} ,$$

rather than over all Φ such that

$$\Phi \varepsilon F^+$$

as above. It is assumed that the discrepancy between these will make very little difference, especially if d is small, and this point is ignored in the following. Note however that the discrepancy exists even for d=0, if the machine is not in a correctly discriminating state.) Hence the increment/decrement functions must satisfy

$$p.\Delta_i(p) = -(1-p).\Delta_d(p) .$$

Consider the functions

$$\Delta_i(x) = \nu.(1-x); \qquad \ldots (8.8)$$
$$\Delta_d(x) = -\nu.x;$$

where ν is a positive constant with $\nu<1$. These give us no trouble with attempts to make out-of-range adaptations, for

$0 \leq x \leq 1$ implies

$$x + \nu(1-x) = \nu + x(1-\nu) \leq \nu + 1 - \nu = 1;$$

and $x - \nu x = (1-\nu)x \geq 0;$

since

$$\nu < 1.$$

Using these functions,

$$p_k^+(t+1) = p_k^+(t) + \nu(1 - p_k^+(t)).^{\lceil \phi_k = P_k^+ \rceil} - \nu p_k^+(t).^{\lceil \phi_k \neq P_k^+ \rceil} ;$$

$$\text{so} \quad p_k^+(t+1) = \begin{cases} (1-\nu).p_k^+(t) + \nu & \text{with probability } p, \\ (1-\nu).p_k^+(t) & \text{with probability } q. \end{cases}$$

This is an EWPA formula (see Chapter 5), and so

$$E[p_k^+(t)] = p(1 - (1-\nu)^t) + (1-\nu)^t.p_k^+(0) ,$$

$$Var[p_k^+(t)] = p(1-p)(1 - (1-\nu)^{2t}).\nu/(2-\nu) ;$$

and the limiting expectation and variance of $p_k^+(t)$ for large t are

$$p \quad \text{and} \quad p(1-p).\nu/(2-\nu)$$

respectively. Hence these increment/decrement functions ensure that if all classifications are correct (by a sufficiently large margin), then the expected value of the k'th PDW of class F^+ approaches

$$Pr[\phi_k = P_k^+ | \Phi \epsilon F^+];$$

and its variance can be made as small as we please by choosing a small enough ν.

8.6 Discussion.

Both the probabilistic and the perceptron-like aspects of the STeLLA decision strategy have now been investigated to a limited extent. I have found no way of systematically investigating the combined effect of both, but the following remarks give a qualitative picture of what should happen.

Unless the PDW's are specially chosen initially, most classifications will be incorrect at first, and rules (8.1) and (8.2) will be used most of the time. This period of perceptron-like behaviour can be prolonged by increasing the threshold d, but too large a threshold may stall the learning procedure. For, if d is large then the size of the final weight vector must be large (see

Chapter 3), and hence some components of the final weight vector must be large in size. This means that

$$\left| \log \frac{1 - p_k^+}{1 - p_k^-} \right|$$

is large, and so

$$\frac{1 - p_k^+}{1 - p_k^-}$$

is either large or small, for some k. Hence the PDW's take values at or near the ends of their ranges. Now if the increment/decrement functions are like those in (E), it is inevitable that the PDW's will make continual frustrated attempts to adapt out of range, causing erratic and perhaps seemingly irrational behaviour. But if the increment/decrement functions are like those in (8.8), which by their very nature cannot attempt out-of-range adaptations, the changes made in the PDW's will be very small if the latter are near the ends of their ranges. This in itself may stall the learning procedure.

After the initial period of frequent misclassifications is over, but before the machine has converged to a correctly discriminating state, both sets of rules will be used fairly often. This invalidates the convergence theorem because the intervention of rules (8.3) and (8.4) could nullify the converging effects of rules (8.1) and (8.2). This seems unlikely, though, if rules (8.3) and (8.4) are well chosen, since both sets of rules are designed to achieve the same end (improving the performance of the machine) and it is probable that they will help rather than hinder each other. This of course depends on the exact form of the increment/decrement functions.

In the final stages of learning, rules (8.3) and (8.4) will be used most often in an attempt to refine the machine's discrimination. No matter how well these rules work, it is inevitable that in a long sequence of correct classifications the PDW's will wander away from their optimum positions (provided that the patterns are not presented in a systematic order), unless the functions Δ_i and Δ_d are identically zero. Hence rules (8.1) and (8.2) must be invoked sometimes during this stage, and the purpose of the threshold d is to prevent misclassifications occurring while these rules put the machine back on the right track.

The above arguments can hardly be called watertight or indisputable, and are natural targets for criticism in the form of counter-examples. It is possible that the STeLLA strategy, in attempting to combine the advantages of the perceptron and maximum likelihood methods, in fact turns out to combine their disadvantages instead. One cannot point to STeLLA for confirmation and say "she works"--she does, but it is difficult to isolate the success of just the pattern classification part, for it forms only a small sub-section of a rather complex machine. Further mathematical and experimental investigation of this pattern-classification technique is required, and because of the many variables involved this could form a complete research topic. It was for this reason that the STeLLA technique was not simulated, as were the perceptron and maximum likelihood decision schemes.

Chapter 9

CONCLUSION

The principal goal of this work has been the development of an understanding of the characteristics of the perceptron and maximum likelihood decision strategies. We shall discuss here the results of the investigation for each method in turn.

The perceptron was found to behave perfectly provided the conditions are favourable, that is, provided the environment is noiseless and linearly separable (note that this is not true of the maximum likelihood decision strategy if independence is incorrectly assumed). It was found experimentally that, for the environment used, the mean number of mistakes made during training was approximately half the smallest upper bound obtainable from the perceptron convergence theorem. Since a lower bound to the number of mistakes made is zero (the arbitrarily chosen initial weight vector may itself discriminate between the pattern classes), the mean number of mistakes to convergence found experimentally was the same as the mean of the upper and lower bounds for this quantity. Unfortunately a discriminating weight vector must be known before the upper bound can be calculated; nevertheless the very existence of this bound--even though it may not be known--must surely provide some comfort to trainers of perceptrons.

In noisy conditions, the perceptron tends to be misled by query vectors which are unavoidably misclassified. This is basically because the perceptron is sensitive to the outer bounds of the "clouds"

representing the pattern classes in feature space, rather than to the centres of gravity of these clouds. After some deliberation, a variant of the perceptron, which I called the "threshold perceptron", was defined. Heuristic arguments were used to show that this appears to have a better chance of behaving well in noisy conditions than the standard perceptron, especially when the noise level is low, and this was confirmed by experiment. As predicted, the behaviour of the threshold perceptron deteriorated rapidly as the noise level increased. This deterioration can be partially restrained by increasing the threshold, but additional cost is entailed here since convergence time increases linearly with threshold size.

The second adaptive decision strategy investigated, the maximum likelihood method, is optimal (in a precise sense) under all conditions, but unfortunately its implementation is impractical unless restrictive assumptions about statistical independence of features are made. This prompted an examination of situations for which the independence assumption is valid, and it was found that the assumption holds only for a greatly restricted class of environments. It is clear, however, that adequacy rather than validity of the assumption is the key factor here. This opens a new field for investigation which is hardly touched upon in this thesis. I feel that some theory of adequacy of the assumption would prove extremely useful to designers of practical machines. The main difficulty that such a theory would have to face is that adequacy is goal dependent--adequate for what?--whereas validity can be investigated without consideration of goals. One surprising result that came out of the experimental work is that for the environment used, adequacy of the assumption for

the noiseless pattern classes ensured adequacy (or near-adequacy--there was a small error rate) when independent noise was added.

The work on the independence assumption showed that if a pattern classification situation is such that

1) the features used are binary (in fact this restriction is not necessary),

2) the independence assumption is valid,

3) the pattern classes are separated in hyperspace;

then one might as well forget weighted decisions and use exact matching procedures based on the features indicated as important. (This is equivalent to a STeLLA-like decision with binary weights.) If most of the features are important for all of the classes, then an efficient way of implementing this decision is to store a representative query vector for each class, and use as discriminatory functions the negative of the Hamming distance between the query vector and the stored representatives. This is in fact a *voting scheme*[1], and I suggest that this accounts for the fact that voting schemes have often proved as successful as weighted decisions (see Chapter 1).

The gap between adequacy and validity of the independence assumption is particularly striking here. If we replace condition (2) above by

2) the independence assumption is assumed to be adequate,

---

[1] Voting schemes can be similarly defined for non-binary features: instead of the Hamming distance one uses the number of query vector components whose value is different from the corresponding component of the stored representative. Bobrow & Klatt (1968) provide a practical example of this. The arguments given are easily modified to cover this case.

then I have not shown, and indeed I do not believe, that a voting scheme is as powerful as the resulting maximum likelihood (independence assumed) decision strategy. A theory of adequacy of the assumption would enable us not to close this gap but at least to chart it.

The secondary objective of this research was to provide a basis for theoretical investigation of the two main decision strategies with a view to combining their virtues. Such a basis was found in the STeLLA method, a decision scheme which combines the various strategies which exist in the learning machine STeLLA. It was found that an appropriate specialization of the STeLLA strategy is almost equivalent to the threshold perceptron method. Another version of the STeLLA strategy can be shown to behave in a manner similar to the maximum likelihood (independence assumed) method. It is not possible to find a version which behaves in exactly the same manner as the maximum likelihood strategy because it was found necessary to use somewhat simplified discriminatory functions. Further investigation of the STeLLA method was not undertaken because of the enormity of the problem, but I feel that this may prove to be a fruitful topic for both theoretical and experimental investigation.

However, although this thesis has been concerned with some very basic adaptive decision mechanisms, for reasons given in Chapter 1, it would be wrong to omit mention of several more advanced topics, apart from the few mentioned above, which are of vital importance to designers of practical decision machines.

General ways of implementing non-linear decisions have long been sought. The inclusion of logical combinations of features as new features has been used fairly successfully (see for example Uhr &

Vossler, 1963), but the number of possible combinations is extremely large, especially if more than pairwise interactions are envisaged. One compromise method is proposed by Samuel (1967). Another way of realizing a general non-linear decision is to use *layered machines* (Nilsson, 1965). These consist of hierarchies of linear decision machines, the output of one layer being used as input to the next. In view of the proven effectiveness of hierarchical solutions to problems in artificial intelligence generally, layered machines seem worthy of detailed investigation. One difficulty is the problem of deciding which part of a complex machine is to be "rewarded" (reinforced) for a correct decision--the "credit assignment problem". Very little is known about layered machines; the only operational example I can give is Widrow's "Madaline" (1962, 1963).

A further topic relevant to adaptive decision strategies is the feature *selection* problem. At present, feature selection is usually undertaken by the designer of the machine. It is conceivable, however, that the decision strategy could provide some assistance here by evaluating the usefulness (to the decision) of each feature, and perhaps generating new features comprising logical combinations or random mutations of useful features. Uhr & Vossler's machine (1963) attempts this in a primitive manner. Some theoretical work has been done on the problem of feature evaluation and selection by Lewis (1962) and Kamentsky & Liu (1963).

As decision machines become more complicated, the problem of instability will arise. This has already been mentioned in connection with both bootstrapping machines (Chapter 2) and the STeLLA method (Chapter 8). It seems highly probable that it will occur to a much

greater extent in layered machines. Ihre Pohl contends that there is an optimum amount of information storage ability for learning machines, above which they begin to break down (as far as I know, this is unpublished). This appears to be an interesting topic for research.

BIBLIOGRAPHY

Agmon, S. (1954) The relaxation method for linear inequalities. *Can. J. Math.* 6(3): 382-392.

Andreae, J.H. (1964) STeLLA: A scheme for a learning machine. In *Automatic and Remote Control*, ed. by Broida, V., 407-502. Proc. 2nd IFAC Congress. London: Butterworths.

Andreae, J.H. (1969) Learning machines: A unified view. In *Encyclopaedia of Information, Linguistics, and Control*, ed. by Meetham, A.R. & Hudson, R.A., 261-270. Oxford: Pergamon.

Andrew, A.M. (1963) Pre-requisites of self-organization. In Tou & Wilcox (1964): 381-391.

Bledsoe, W.W. & Bisson, C.L. (1962) Improved memory matrices for the n-tuple pattern recognition method. IRE Trans. Electron. Comps. (corresp.) EC-11(3): 414-415; June.

Bobrow, D.G. & Klatt, D.H. (1968) A limited speech recognition system. Proc. Fall Joint Comp. Conf. 33, Part 1: 305-318.

Bobrow, D.G., Hartley, A.K., & Klatt, D.H. (1969) A limited speech recognition system II. BBN Rept. No. 1819, Job No. 11254, under Contract NAS 12-138, Cambridge, Mass..

Chow, C.K. (1957) An optimum character recognition system using decision functions. IRE Trans. Electron. Comps. EC-6(4): 247-254; December.

Chow, C.K. (1962) A recognition system using neighbour dependence. IRE Trans. Electron. Comps. EC-11(5): 683-690; October.

Chow, C.K. (1963) An experimental result on character recognition. IEEE Trans. Electron. Comps. (corresp.) EC-12(1): 25; February.

Cooper, P.W. (1963) Hyperplanes, hyperspheres, and hyperquadrics as decision boundaries. In Tou & Wilcox (1964): 111-138.

Craik, K.J.W. (1952) The nature of explanation. Cambridge (England) Univ. Press.

Efron, B. (1963) The perceptron correction procedure in non-separable situations. In Stanford Res. Inst. *Applied Physics Laboratory Research Note*; August.

Feigenbaum, E.A. & Feldman, J. (Eds.) (1963) Computers and thought. McGraw Hill.

Gaines, B.R. & Andreae, J.H. (1966) A learning machine in the context of the general control problem. Proc. 3rd IFAC Congress, paper 14B. London: Inst. Mech. Engrs..

Gnedenko, B.V. (1962) The theory of probability. New York: Chelsea.

Gold, B. (1959) Machine recognition of hand-sent Morse code. IRE Trans. Info. Theory IT-5(1): 17-24; March.

Good, I.J. (1965) The estimation of probabilities. Res. Mono. No. 30, M.I.T. Press.

Griffin, J.S., King, J.H., & Tunis, C.J. (1963) A pattern-identification device using linear decision functions. In Tou & Wilcox (1964): 169-193.

Guzman, A. (1968) Decomposition of a visual scene into bodies. Proc. Fall Joint Comp. Conf. 33, Part 1: 291-304.

Highleyman, W.H. (1961) Linear decision functions with application to pattern recognition. PhD dissertation, Elec. Engrg. Dept., Polytech. Inst. Brooklyn, N.Y.; June.

Highleyman, W.H. (1962) Linear decision functions with application to pattern recognition. Proc. IRE 50(6): 1501-1514; June.

Hill, D.R. (1969) An ESOTerIC approach to some problems in automatic speech recognition. *Intern. J. Man-Machine Studies* 1(1): 101-121; January.

Hill, D.R. & Wacker, E.B. (1969) ESOTerIC II--An approach to practical voice control: Progress report 69. In *Machine Intelligence 5*, ed. by Meltzer, B. & Michie, D., 463-493. Edinburgh Univ. Press.

Hubel, D.H. & Wiesel, T.N. (1962) Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *J. Physiol.* 160: 106-154.

Kamentsky, L.A. & Liu, C.N. (1963) A theoretical and experimental study of a model for pattern recognition. In Tou & Wilcox (1964): 194-218.

Koestler, A. (1964) The act of creation. London: Hutchinson.

Lettvin, J.Y., Maturana, H., McCulloch, W.S., & Pitts, W. (1959) What the frog's eye tells the frog's brain. Proc. IRE 47(11): 1940-1951. Also in McCulloch (1965): 230-255.

Lewis, P.M. (1959) Approximating probability distributions to reduce storage requirements. Info. & Control 2(3): 214-225; September.

Lewis, P.M. (1962) The characteristic selection problem in recognition systems. IRE Trans. Info. Theory IT-8(2): 171-178; February.

McCarthy, J., Earnest, L.D., Reddy, D.R., & Vicens, P.J. (1968) A computer with hands, eyes, and ears. Proc. Fall Joint Comp. Conf. 33, Part 1: 329-338.

McCulloch, W.S. (1965) Embodiments of mind. M.I.T. Press.

Marill, T. & Green, D.M. (1960) Statistical recognition functions and the design of pattern recognizers. IRE Trans. Electron. Comps. EC-9(4): 472-477; December.

Maron, M.E. (1962) Design principles for an intelligent machine. IRE Trans. Info. Theory IT-8(5): S179-S185; September.

Middleton, D. (1960) Introduction to statistical communication theory. New York: McGraw Hill.

Minsky, M. (1958) Some methods of artificial intelligence and heuristic programming. In *Mechanisation of thought processes*, Vol. 1: 5-27. London: HMSO (1961).

Minsky, M. & Selfridge, O.G. (1960) Learning in random nets. In *Information Theory*, ed. by Cherry, C., 335-347. Proc. 4th London Symposium. London: Butterworths (1961).

Minsky, M. (1961) Steps toward artificial intelligence. Proc. IRE 49(1): 8-30; January. Also in Feigenbaum & Feldman (1963): 406-450.

Minsky, M. & Papert, S. (1969) Perceptrons--an introduction to computational geometry. M.I.T. Press.

Nagy, G. (1967) Prospects in hyperspace: State of the art in pattern recognition. IBM Res. Paper RC-1869; June. The first part of this report was published under the title "Classification algorithms in pattern recognition" in IEEE Trans. Audio and Electro-Acoustics AU-16(2): 203-212; June, 1968.

Newell, A., Shaw, J.C., & Simon, H.A. (1957) Empirical explorations with the logic theory machine: A case study in heuristics. Proc. Western Joint Comp. Conf. 15: 218-239. Also in Feigenbaum & Feldman (1963): 109-133.

Nilsson, N.J. (1965) Learning machines: Foundations of trainable pattern-classifying systems. McGraw Hill.

Papert, S. (1960) Some mathematical models of learning. In *Information Theory*, ed. by Cherry, C., 353-363. Proc. 4th London Symposium. London: Butterworths (1961).

Reddy, D.R. (1967) Computer recognition of connected speech. *J. Acoust. Soc. Am.* 42(2): 329-347; August.

Rosenblatt, F. (1957) The perceptron: A perceiving and recognizing automaton. In *Cornell Aeron. Lab. Rept.*, No. 85-460-1; January.

Rosenblatt, F. (1962) Principles of neurodynamics. New York: Spartan Books.

Samuel, A.L. (1959) Some studies in machine learning using the game of checkers. *IBM J. of Res. & Develop.* 3(3): 211-229; July. Also in Feigenbaum & Feldman (1963): 71-105.

Samuel, A.L. (1967) Some studies in machine learning using the game of checkers II--recent progress. *IBM J. of Res. & Develop.* 11(6): 601-617; November.

Sebestyen, G.S. (1962) Decision-making processes in pattern recognition. New York: MacMillan.

Selfridge, O.G. (1958) Pandemonium: A paradigm for learning. In *Mechanisation of thought processes*, Vol. 1: 513-526. London: HMSO (1961).

Selfridge, O.G. & Neisser, U. (1963) Pattern recognition by machine. In Feigenbaum & Feldman (1963): 237-250.

Tou, J.T. & Wilcox, R.H. (Eds.) (1964) Computer and information sciences. Washington: Spartan Books.

Uhr, L. & Vossler, C. (1963) A pattern recognition machine that generates, evaluates, and adjusts its own operators. In Feigenbaum & Feldman (1963): 251-268; and Uhr (1966): 349-364.

Uhr, L. (Ed.) (1966) Pattern recognition. New York: Wiley.

Widrow, B. (1962) Generalization and information storage in networks of Adaline "neurons". In *Self-Organizing Systems*, ed. by Yovits, Jacobi, & Goldstein, 435-461. Washington: Spartan Books.

Widrow, B. & Smith, F.W. (1963) Pattern-recognizing control systems. In Tou & Wilcox (1964): 288-317.

Appendix A

PROOFS OF THEOREMS QUOTED IN THE TEXT

Theorem 1: The convergence theorem for a perceptron whose weight vector is of a given initial length.

This simple extension of the convergence theorem is given here for completeness. The result is directly applicable to the two class case and generalizes easily if there are more than two pattern classes, as shown by Minsky & Papert (1969).

THEOREM 1.

Let F be a set of real n-vectors with

$\Phi \varepsilon F$ implies $|\Phi| \leq \alpha$.

Suppose there exists a unit n-vector W* and some $\delta > 0$ with

$\Phi \varepsilon F$ implies $W^*.\Phi > \delta$.

Then if W is initially chosen to be an n-vector of length λ, the program

TEST: Choose $\Phi \varepsilon F$;

If $W.\Phi \leq 0$ then assign $W+\Phi$ to W;

Go to TEST;

will change W at most $(\alpha^2 + 2\lambda\delta)/\delta^2$ times.

Proof. Define

$G(W) = W^*.\hat{W} \leq 1$.

Consider the behaviour of G(W) on successive performances of the assignment statement.

$G(W^{t+1}) = W^*.\hat{W}^{t+1} = W^*.(W^t + \Phi)/|W^t + \Phi|$.

Now $W^*.(W^t + \Phi) \geq W^*.W^t + \delta$;

so after the m'th execution of the assignment statement,

$$W^*.(W^{m-1} + \Phi) \geq m\delta - \lambda.$$

Also,

$$|W^t + \Phi|^2 \leq |W^t|^2 + \alpha^2, \quad \text{since} \quad W^t.\Phi \leq 0;$$

so $|W^m|^2 \leq m\alpha^2 + \lambda^2$.

Hence

$$1 \geq G(W^m) \geq (m\delta - \lambda)/(m\alpha^2 + \lambda^2)^{1/2},$$

$$m\alpha^2 + \lambda^2 \geq m^2\delta^2 - 2\lambda\delta m + \lambda^2,$$

so $m \leq (\alpha^2 + 2\lambda\delta)/\delta^2$.

Theorem 2: The convergence theorem for a threshold perceptron.

THEOREM 2.

Let F be a set of real n-vectors with

$\Phi \varepsilon F$ implies $|\Phi| \leq \alpha$.

Suppose there exists a unit n-vector W* and some $\delta > 0$ with

$\Phi \varepsilon F$ implies $W^* . \Phi > \delta$.

Then if W is initially chosen to be an n-vector of length λ, the program

TEST: Choose $\Phi \varepsilon F$;

If $W . \Phi \leq d$ then assign $W + \Phi$ to W;

Go to TEST;

will change W at most $(\alpha^2 + 2d + 2\lambda\delta)/\delta^2$ times for any $d \geq 0$.

Proof. Using the same notation as in the previous theorem, we find

$W^* . (W^{m-1} + \Phi) \geq m\delta - \lambda$ as before,

but $|W^m|^2 \leq m(\alpha^2 + 2d) + \lambda^2$ since $W^t . \Phi \leq d$.

Hence

$$1 \geq G(W^m) \geq (m\delta - \lambda)/(m(\alpha^2 + 2d) + \lambda^2)^{1/2},$$

so $m \leq (\alpha^2 + 2d + 2\lambda\delta)/\delta^2$.

Theorem 3: Concerning the size of the final weight vector of a perceptron.

The quoted result concerning the length of the final weight vector of a perceptron whose initial weight vector is constrained to a certain size is proved here.

THEOREM 3.

If the weight vector of a perceptron is set initially to length λ and allowed to vary according to the usual perceptron adaptation procedure, then for any $\varepsilon>0$ there exists a non-trivial environment for which the final weight vector W^t satisfies

$$|W^t| < 1 + \varepsilon,$$

provided an unlucky choice is made for the initial weight vector.

Proof. Let F consist of unit vectors clustered around some vector $\Phi^* \varepsilon F$, i.e.

$$\Phi \varepsilon F \text{ implies } \Phi^*.\Phi > 1 - \theta, \text{ for some small } \theta.$$

Let the initial choice of W be

$$W^0 = -\lambda\Phi^*.$$

Denote the weight vector after the k'th mistake has been made during training by W^k; let W^t be the final weight vector.

Then

$$W^k = \Phi^k + W^{k-1} \quad (t \geq k \geq 1), \text{ where } \Phi^k \varepsilon F.$$

Hence

$$\frac{W^t - W^0}{|W^t - W^0|}.\Phi^* = \frac{(\Phi^t + \Phi^{t-1} + \ldots + \Phi^1).\Phi^*}{|(\Phi^t + \Phi^{t-1} + \ldots + \Phi^1)|}$$

$$> \frac{t(1-\theta)}{t} = 1 - \theta. \qquad \ldots \text{ (A.1)}$$

Now $(W^t - \Phi^t).\Phi^t = W^{t-1}.\Phi^t \leq 0$ since Φ^t was misclassified; hence

$$W^t.\Phi^t \leq |\Phi^t|^2 = 1.$$

Let $\chi = \Phi^t - \Phi^*$;

$$|\chi|^2 = |\Phi^t - \Phi^*|^2 \leq 2 - 2(1 - \theta) = 2\theta.$$

Hence

$$W^t.\Phi^* = W^t.(\Phi^t - \chi)$$
$$\leq 1 + \sqrt{2\theta}|W^t| \qquad \ldots (A.2)$$

Also,

$$|W^t| \leq (W^t.\Phi^*)|\Phi^*| + |W^t - (W^t.\Phi^*)\Phi^*|$$
$$\leq 1 + \sqrt{2\theta}|W^t| + |W^t - (W^t.\Phi^*)\Phi^*| \quad \text{from (A.2)} \qquad \ldots (A.3)$$

Let $v = -W^t + (W^t.\Phi^*)\Phi^*$,

$u = W^t - W^0$,

$w = W^0 - (W^t.\Phi^*)\Phi^*$.

Then u,v,w form a triangle with a right angle between v and w, and the acute angle ψ between u and w satisfies

$\cos\psi > 1 - \theta$ from (A.1).

Hence

$$\sin^2\psi = 1 - \cos^2\psi < 2\theta - \theta^2 < 2\theta,$$

so $|v| = |w|\tan\psi$

$< |W^0 - (W^t.\Phi^*)\Phi^*|.\sqrt{2\theta}/(1-\theta)$

$= (\lambda + W^t.\Phi^*).\sqrt{2\theta}/(1-\theta)$

$\leq (\lambda + 1 + \sqrt{2\theta}|W^t|).\sqrt{2\theta}/(1-\theta)$ from (A.2)

Now $W^t \leq 1 + \sqrt{2\theta}|W^t| + |v|$ from (A.3)

$< 1 + \sqrt{2\theta}|W^t| + (\lambda + 1 + \sqrt{2\theta}|W^t|).\sqrt{2\theta}/(1-\theta)$,

so $$|W^t| < \frac{1 + (\lambda+1).\sqrt{2\theta}/(1-\theta)}{1 - \sqrt{2\theta} - 2\theta/(1-\theta)}$$

$$= \frac{1 - \theta + (\lambda + 1)\sqrt{2\theta}}{(1-\theta)(1 - \sqrt{2\theta}) - 2\theta}$$

$\to 1$ as $\theta \to 0$ for any fixed λ.

Hence given $\varepsilon>0$ there exists θ with

$$|W^t| < 1 + \varepsilon.$$

Theorem 4: Generalization of the convergence theorem for STeLLA-like pattern classifiers.

Before stating the theorem we introduce some notation, and state the adaptive rules for STeLLA in a concise and tractable form. The prototype bits are assumed to be all zero, and the process of adaptation to successfully classified query vectors is considered to be suspended, as before. The rules (8.1) and (8.2) of Chapter 8 are:

$\Phi \varepsilon F^+$ and $W.\Phi < d$ implies

$$W_k' = W_k + [\log(1 - \frac{\tilde{\Delta}_d(p_k^+)}{1 - p_k^+}) - \log(1 - \frac{\tilde{\Delta}_i(p_k^-)}{1 - p_k^-})].(\phi_k = 1)$$

$$= W_k + \eta(p_k^+, p_k^-).\phi_k \qquad (1 \leq k \leq n);$$

where W' is the modified (new) weight vector, n is the dimensionality of the query vectors, and

$$\eta(x,y) = \log(1 - \frac{\tilde{\Delta}_d(x)}{1 - x}) - \log(1 - \frac{\tilde{\Delta}_i(y)}{1 - y}).$$

Similarly,

$\Phi \varepsilon F^-$ and $W.\Phi > -d$ implies

$$W_k' = W_k - [\log(1 - \frac{\tilde{\Delta}_d(p_k^-)}{1 - p_k^-}) - \log(1 - \frac{\tilde{\Delta}_i(p_k^+)}{1 - p_k^+})].(\phi_k = 1)$$

$$= W_k - \eta(p_k^-, p_k^+).\phi_k \qquad (1 \leq k \leq n).$$

We write the adaptive rules in the form of a program as follows:

TEST: Choose $\Phi \varepsilon F^+ \cup F^-$; . . . (A)

If $\Phi \varepsilon F^+$ then let $\Psi = \Phi$; else let $\Psi = -\Phi$;

If $W.\Psi < d$ then

If $\Phi \varepsilon F^+$ then replace W_k by $W_k + \eta(p_k^+, p_k^-).\psi_k$ for $1 \leq k \leq n$;

If $\Phi \varepsilon F^-$ then replace W_k by $W_k + \eta(p_k^-, p_k^+).\psi_k$ for $1 \leq k \leq n$;

Go to TEST.

We use the notation $\eta(k)$ for $\eta(p_k^+, p_k^-)$ or $\eta(p_k^-, p_k^+)$ in places where the distinction is either not important or adequately made by the context; the argument (k) is shown explicitly because the dependency of η on k is vital. The notation Ψ for Φ or $-\Phi$ depending on the classification of Φ, as above, is also used.

THEOREM 4.

Let F^+ and F^- be classes of an n-dimensional binary query space, and suppose there exists a unit n-vector W* and some $\delta>0$ such that

$\Phi \varepsilon F^+$ implies $W^*.\Phi > \delta$;

$\Phi \varepsilon F^-$ implies $W^*.\Phi < -\delta$.

Suppose not only that there are no attempts to make out-of-range adaptations (§8.4, Assumption C), but also that there exists $\Omega>0$ such that no attempt is made to take any PDW out of the range

$$[\ 1/(1 + e^{\Omega})\ ,\ 1 - 1/(1 + e^{\Omega})\ ].$$

(This condition ensures that each component of all weight vectors obtained in the course of adaptation satisfies

$$|W_k| \leq \Omega,$$

as can be readily verified.)

Let η be a real valued function defined on the Cartesian product of the interval [0,1] with itself, such that there exist ξ and ζ with

$$0 < \xi \leq \eta(x,y) \leq \zeta \quad \text{for all} \quad 0 \leq x,y \leq 1,$$

and $\zeta < \xi(1 + \delta/n)$.

Then if W is initially chosen to be an n-vector of length λ, the program (A) above will change W at most

$$\frac{n\zeta^2 + 2\zeta d + 2\lambda\xi\delta + 2n(\zeta-\xi)(\Omega-\lambda)}{[\xi(n + \delta) - n\zeta]^2}$$

times.

Proof. Define

$$G(W) = W^*.\hat{W} \leqq 1.$$

Consider the behaviour of G(W) on successive changes of the weight vector.

$$G(W^{t+1}) = W^*.W^{t+1}/|W^{t+1}|.$$

$$W^*.W^{t+1} = \Sigma\ W_k^*.(W_k^t + \eta(k)\psi_k)$$

$$= W^*.W^t + \Sigma\ (W_k^*.\psi_k + 1)\eta(k) \quad - \Sigma\ \eta(k).$$

Now $|W_k^*| \leqq 1$ since W* is a unit vector;

$|\psi_k| \leqq 1$ since the query vectors Φ are binary.

Hence

$$W_k^*.\psi_k + 1 \geqq 0.$$

So $W^*.W^{t+1} \geqq W^*.W^t + \xi.\Sigma\ (W_k^*.\psi_k + 1) \quad - n\zeta$

$$\geqq W^*.W^t + \xi W^*.\Psi + n(\xi-\zeta)$$

$$\geqq W^*.W^t + \xi\delta + n(\xi-\zeta)$$

$$\geqq W^*.W^t + \delta';$$

where

$$\delta' = \xi(n + \delta) - n\zeta$$

$$> 0 \quad \text{by the conditions of the theorem.}$$

Hence after the m'th change of W,

$$W^*.W^m \geqq m\delta' - \lambda.$$

Also,

$$|W^{t+1}|^2 = \Sigma\ (W_k^t + \eta(k).\psi_k)^2$$

$$\leqq |W^t|^2 + \zeta^2|\Psi|^2 + 2.\Sigma\ (W_k^t.\psi_k + \Omega).\eta(k) \quad - 2\Omega.\Sigma\ \eta(k)$$

$$\leqq |W^t|^2 + n\zeta^2 + 2\zeta.\Sigma\ (W_k^t.\psi_k + \Omega) \quad - 2n\Omega\xi$$

$$< |W^t|^2 + n\zeta^2 + 2n\Omega(\zeta-\xi) + 2\zeta d.$$

So $|W^m|^2 < m[n\zeta^2 + 2n\Omega(\zeta-\xi) + 2\zeta d] + \lambda^2.$

Hence

$$1 \geq G(W^m) > \frac{m\delta' - \lambda}{\{m[n\zeta^2 + 2n\Omega(\zeta-\xi) + 2\zeta d] + \lambda^2\}^{1/2}},$$

$$\text{so } m < \frac{n\zeta^2 + 2n\Omega(\zeta-\xi) + 2\zeta d + 2\delta'\lambda}{\delta'^2}$$

$$= \frac{n\zeta^2 + 2\zeta d + 2\lambda\xi\delta + 2n(\zeta-\xi)(\Omega-\lambda)}{[\xi(n+\delta) - n\zeta]^2}.$$

Corollary 1. The convergence theorem is satisfied if

$$\tilde{\Delta}_i(x) = \mu(1-x) = -\tilde{\Delta}_d(x),$$

for any constant $\mu>0$. If these increment/decrement functions are used then no restriction is required on the size of each component of all weight vectors, so one need only assume that no out-of-range PDW adaptations are attempted.

Corollary 2. If in addition to the conditions of the theorem,

$$W_k^* > 0 \quad (1 \leq k \leq n),$$

then the theorem holds even if

$$\zeta > \xi(1 + \delta/n),$$

provided none of the other conditions are violated.

Appendix B

GLOSSARY

This glossary explains important terms, abbreviations, and some notations which are used throughout this thesis. Although no attempt has been made to include every symbol used, all frequently used global symbols are given. The order of items in the glossary corresponds roughly to the order in which the ideas are introduced in the thesis, although a couple of miscellaneous notations appear at the end.

Query (feature) vector, Φ, Φ^*, n, r, α.

The query or feature vector is the input to the decision phase. Its i'th component indicates the extent to which the i'th feature is present. Query vectors are always denoted by Φ; Φ^* is sometimes used for a particular query vector. Φ is n-dimensional; ϕ_i can take on r values; and the maximum length of Φ is α.

Query (feature) space, Environment.

Query or feature space is the n-dimensional hyperspace in which query vectors lie. The environment is the set of possible query vectors, with their frequencies and classes.

Pattern class, reject and noise only classes, $F^{(i)}$, m, F^+, F^-, F'.

The purpose of a pattern classifier is to classify inputs into pattern classes. These may include a reject class (information not sufficient for a firm decision) and/or a noise only class (no pattern present). Pattern classes are denoted by $F^{(i)}$ $(1 \leq i \leq m)$, or, in the two class case, by F^+ and F^-. It is found convenient to define

$F' = \{\Phi | \Phi \epsilon F^+\} \cup \{-\Phi | \Phi \epsilon F^-\}$.

Convergence, Convergence time.

An adaptive pattern classification machine is said to have converged if it can classify all possible query vectors correctly. If noise is present we only require it to classify all noiseless query vectors correctly. The convergence time of a particular machine for a particular environment is the mean number of patterns presented before convergence is reached.

Discriminatory functions, surfaces, $f^{(i)}$, f^+, f^-.

Classification is effected by discriminatory functions $f^{(i)}$ (f^+ and f^- in the two class case), one associated with each pattern class, such that (ideally)

$f^{(i)}(\Phi) > f^{(j)}(\Phi)$ for all $j \neq i$ if and only if $\Phi \epsilon F^{(i)}$. The surfaces in feature space given by $f^{(i)}(\Phi) = f^{(j)}(\Phi)$ $(j \neq i)$ are called

discriminatory surfaces.

Weights, weight vector, $W^{(i)}$, λ.

Linear discriminatory functions are of the form

$$f^{(i)}(\Phi) = w_1^{(i)}\phi_1 + w_2^{(i)}\phi_2 + \ldots + w_n^{(i)}\phi_n + w_{n+1}^{(i)}.$$

The coefficients w are called weights, and the weight vector is

$$W^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_{n+1}^{(i)}).$$

The length of the initial weight vector (for a perceptron) is denoted by λ.

Augmented query vector, Φ'.

Linear discriminatory functions can be written as $f^{(i)}(\Phi) = W^{(i)}.\Phi'$, where the augmented query vector Φ' is such that

$$\phi'_j = \phi_j \quad \text{for } 1 \leq j \leq n; \quad \phi'_{n+1} = 1.$$

Linear separability, discriminating weight vector, W^*, δ.

The two pattern classes F^+ and F^- are said to be linearly separable if there exists a weight vector W^* and some $\delta > 0$ with

$W^*.\Phi' > \delta$ for all $\Phi \varepsilon F^+$;

$W^*.\Phi' < -\delta$ for all $\Phi \varepsilon F^-$.

W^* is called the discriminating weight vector. The notion of linear separability extends to the case where there are more than two pattern classes.

Threshold perceptron, d.

The notion of a threshold perceptron is explained in Chapter 3. d denotes the threshold of a threshold perceptron.

Pr[A], Pr[A|B], E[X], Var[X].

Pr[A] (Pr[A|B]) denotes the probability of the event A (given the event B). E[X], Var[X] denote the expectation and variance, respectively, of the random variable X.

EWPA.

Abbreviation for the exponentially weighted past average probability estimation procedure (see Chapter 5).

Prototype, P^+, P^-.

A prototype is a query vector, generally contained in the pattern class with which the prototype is associated, which is assumed to be representative of that class. It is denoted by P^+, P^- (for classes F^+ and F^- respectively).

Pattern Digit Weight, PDW, p_k^+, p_k^-.

The k'th pattern digit weight (PDW) of the i'th class is a number which represents the danger of overlooking a disparity between the k'th (binary) component of a query vector Φ and the prototype, when assigning Φ to the i'th class. PDW's are denoted by p_k^+, p_k^- (for F^+ and F^- respectively).

Increment/decrement functions, $\Delta_i(x)$, $\tilde{\Delta}_i(x)$, $\Delta_d(x)$, $\tilde{\Delta}_d(x)$.

These specify the quantitative amounts of adaptation used in the STeLLA method (see Chapter 8). Δ_i and Δ_d are also used to denote increment/decrement functions in Chapter 5.

Corners, $\ulcorner\ , \urcorner$.

The Boolean value of an expression is denoted by enclosing the expression in corners. Thus

$$\ulcorner x=5 \urcorner = \begin{cases} 1 & \text{if } x=5, \\ 0 & \text{otherwise.} \end{cases}$$

Unit vectors, $\hat{A}$.

$\hat{A}$ denotes the unit vector in the direction of the vector A, i.e. $\hat{A} = A/|A|$.