Annals of Mathematics and Artificial Intelligence, 1 (1990) 207–239

STOCHASTIC DISCRIMINATION

E.M. KLEINBERG

Department of Mathematics, State University of New York at Buffalo, NY 14214, USA

Abstract

A general method is introduced for separating points in multidimensional spaces through the use of stochastic processes. This technique is called stochastic discrimination.

Keywords: Pattern recognition, complexity theory.

1. Introduction

In this paper, we will introduce a general method for separating points in multidimensional spaces through the use of discrete stochastic processes. This technique of *stochastic discrimination* is extremely general, and has application in diverse areas ranging from pattern recognition and artificial learning where, for example, it may "train itself" to distinguish structures which contain a target pattern from structures which do not, to computational complexity theory where it might be used to identify edges which comprise an optimal tour through a given graph.

The method functions, basically, by taking as input poor solutions to a problem at hand, and using them, in concert, to create good solutions. In fact, the set of poor solutions is the only access the method is ever given to a problem at hand, yet we can rigorously prove that given an appropriate set of such poor solutions, stochastic discrimination will provide *arbitrarily good* solutions within very short (low-degree polynomial) periods of time.

It should be noted here that the notion of *appropriate* as applied to sets of poor solutions is concerned with the degree of dispersion of the solutions in the set rather than with the degree of goodness of them – the technique will create a solution which is arbitrarily close to perfection from input solutions none of which is better than ε away from the rating expected of a random guess *provided only that the input solutions are different from one another*. Of course, the concept of *dispersion* must be made mathematically precise, but on an intuitive level, one might consider this: if one were presented again and again with the same poor solution to a problem, he would have little chance of ever creating anything better than that poor solution – on the other hand, if he were presented again and again with equally poor *but different* solutions to the problem, he would at least be

getting diverse information; and in this case, stochastic discrimination will enable him to create from this diverse information an essentially perfect solution.

In order to get a sense for just how powerful the technique is, we might consider an example based on the well-known parable about producing Shakespeare's play Hamlet from a roomful of monkeys sitting at typewriters. [+] For definiteness sake, let us assume that Hamlet (written in binary) consists of a string of 0's and 1's of length s, that our monkeys are sitting at typewriters capable only of printing the digits 0 or 1, and that every time one of them produces a string of length s, his work is graded and passed on to us (without any indication of what the correct digits are or even of which digits he got right or wrong). Then as we all know, the probability that any one of these graded attempts gets a mark of 100 percent is exponentially small, and hence, given any collection of such attempts whose size is polynomial in s, the probability that a perfect string sits in the collection is exponentially small. What we show, however, is that through the use of stochastic discrimination, this situation can be completely reversed; within a low-degree polynomial amount of time, we can take any such collection of graded attempts at Hamlet and use them to produce a new collection of attempts, *and the probability that a perfect string does* ***not*** *sit in this new collection is exponentially small.* Describing this situation from a slightly different point of view, and with somewhat more specificity, we will show that given any $(16s^4 - 8s^3)$-many random papers from our roomful of monkeys, we can construct a string of 0's and 1's containing less than 1 expected error *total* in the entire play. And since it is easy to see that for large enough s, the probability that any one of our $(16s^4 - 8s^3)$-many random papers is graded better than ε is arbitrarily small, it is clear that our method here is not based on synthesizing something from better and better guesses.

Of course, this example is of little practical importance; however, it is appealing from the point of view of complexity theory since we almost seem to be accomplishing an NP task within P time. This is more fully explored when we consider the application of stochastic discrimination to the traveling salesman problem.

There is an equally appealing characteristic of our approach which is of interest in the field of pattern recognition. Namely, given training data for a solvable pattern recognition problem, if we generate sufficiently "coarse" guesses, then solutions to the problem built from these guesses by stochastic discrimination, although their complexity may increase enormously as they converge to

[+] This example, in and of itself, has very little to do with the application of stochastic discrimination to either complexity theory or pattern recognition. In either of these areas, the problems we wish to solve are not random in nature, and some care is needed in producing the so-called "set of poor solutions". However, the mathematical analogue of this example, which we will discuss in some detail later in the paper, contains the application-independent essence of stochastic discrimination. As such, we feel that this introductory discussion provides worthwhile insight.

perfection, will remain stable as one measures their accuracy on new data not present in the training set. Thus the traditional dilemma, whereby in order to fit the training data well one tends to build large, *training set specific*, models which are of little use in actual practice, can be avoided entirely – through the use of stochastic discrimination, one can build large models which fit the training data but which are not specific to it.

This paper is comprised, essentially, of two parts. In the first, we describe the method, provide its theoretical underpinnings, and prove several theorems which establish the power and generality of the technique. In the second, we consider a specific application of stochastic discrimination to a very general class of problem, but in a context where we have control over the input which we may provide to the technique. We show here how the method should operate in theory.

We refer the reader to two additional papers for discussion of the application of stochastic discrimination to well-known areas of practical importance. In [1], we consider an application involving visual pattern recognition, specifically, the problem of distinguishing among handwritten digits. Here the major difficulty is in supplying input which is sufficiently dispersed so as to provide what is known as a uniform cover of our pattern space; however, we can do well enough that the separation provided by stochastic discrimination is significantly better than that achieved by conventional means. And in [2], we consider an application to computational complexity theory, specifically, toward deriving a (polynomially timed) solution to the traveling salesman problem. Here the wrinkle is not with lack of uniform cover, but rather with accurately rating proposed sets of edges in order to determine if a given set is worthy of being provided as input to the process. Needless to say, we do not prove here that $P = NP$; however, we present what we consider to be a general technique, which operates in (low-degree) polynomial time, for attacking arbitrary NP problems.

1.1. AN OVERVIEW

The class of problem we are considering is one characterized by the following three factors:

(a) we are trying to find a perfect solution to the problem;
(b) we can quickly (polynomial time) determine if a given candidate is a perfect solution to the problem;
(c) if we simply make "unintelligent", independent guesses, then within any polynomial amount of time, the chance that any of the guesses to that point in time is a perfect solution is exponentially small.

The virtue of the independent guess approach is that it can be carried out in parallel – since there is no feedback or other interaction between the guesses they can all be generated simultaneously on a very wide machine. However, the chance of success, being exponentially small, renders the approach worthless. As a result, traditional attack on such problems has been through analysis of the underlying

data – a pattern recognition person might study numerical characteristics of digitized images in formulating sets of rules to distinguish among them, just as complexity theorists might study the spatial characteristics of grid graphs in looking for optimal tours. And although such traditional attack might well include the generation and testing of hypotheses, there is always feedback involved, the guesswork is intelligent, so to speak, and the entire process is fundamentally sequential instead of parallel.

With stochastic discrimination, the traditional approach could be set aside entirely. The notion of training data as something to "learn from" could be completely avoided, and we will show that (c) above could be replaced with

(c′) if we simply make "unintelligent", independent guesses (in parallel, if possible) *and feed them to stochastic discrimination*, then derivative solutions are generated, and within *some* polynomial amount of time, the chance that *none* of these derivative solutions is an instance of what we are looking for is exponentially small.

Thus from a conceptual point of view the traditional approach of learning from training data is replaced with a closed 2-step process, namely, *make a sufficient number of independent guesses, and then stochastically discriminate*. And what is remarkable is that we can rigorously prove that this seemingly "unintelligent" approach always succeeds, and that it succeeds extremely quickly.

In its most general form, the technique operates as follows:

Given a space containing points of two sorts * (our goal is to come up with an accurate solution to the discrimination problem, namely, to come up with a rule for accurately discriminating points of one sort from points of the other), we associate with each point q in the space a discrete stochastic process $\{M_i^q\}$. For each i and q, the random variable M_i^q is a function of the set of i-element sets of (possible) solutions, and so, for any q, by applying the variables in the process $\{M_i^q\}$ to larger and larger sets of proposed solutions, we are able to associate with q a sequence of reals $\{a_i^q\}$. Then the following two surprising phenomena occur: (a) each of the sequences $\{a_i^q\}$ converges, and it converges to only one of two possible (distinct) values; and *all sequences associated with points of one sort converge to one of these values, and all sequences associated with points of the other sort converge to the other of these values*; (b) there exists a single polynomial function which governs the *rate* of convergence of all of these sequences; and, specifically, *one can determine within a polynomial amount of time, for any given point q, just which of the two possible values* $\{a_i^q\}$, *with high probability, converges to*. Thus, within a polynomial amount of time (of looking at *possible* solutions, inaccurate though they all may be), we have come up with a solution which, with high probability, *is perfect*.

* For purely pedagogical reasons, this paper will deal only with *binary* discrimination problems. With relatively little change, however, the technique could be recast in such a way as to enable it to deal with arbitrary *n-ary* discrimination problems.

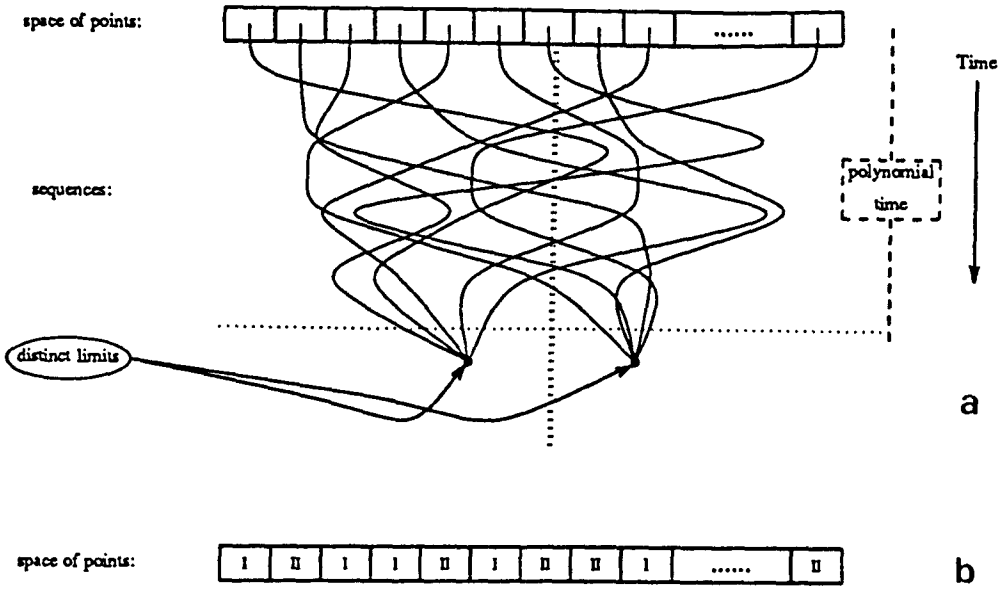

Fig. 1. Graphical representation of the dynamics of generating "stochastic solutions".

In effect, one of these accurate solutions generated by this technique (solutions which will later be referred to as *stochastic algorithms*) consists of a sufficiently large set, S, of "possible solutions" (of size n, say) along with the set of random variables M_n^q. A given structure q will then be classified by this stochastic algorithm as being of type I (type II) if the value of M_n^q on S is "close" to the type I (type II) limit point. If we consider "possible solutions" as items appearing over time, we may pictorially view the dynamics of all of this as in fig. 1a, and the classification given by any stochastic algorithm based on all those "possible solutions" generated by any time beyond that indicated by the horizontal dotted line would be as depicted in fig. 1b.

2. The underlying theory

In the realm of activity which is generally construed as "intelligent", pattern recognition problems clearly occupy a position of central importance. Many areas of application for stochastic discrimination are obvious instances of such problems. But since we will also be able to cast our subsequent work in complexity theory in terms of pattern recognition (learning, for example, to recognize which edges make up an optimal tour in a given grid graph), we will use pattern recognition as the vehicle through which to present stochastic discrimination.

2.1. THE GENERIC PATTERN RECOGNITION PROBLEM

Our entity engaged in "learning" is trying to formulate rules for distinguishing one set of objects (the "examples") from another set of objects (the "nonexam-

ples"). For maximal generality, the paradigm is constructed as follows:

(a) All objects (the "examples" as well as the "nonexamples") are of some fixed structure type. In terms of pattern recognition, the "examples" are those structures which contain a target pattern, and the "nonexamples" are those structures which do not.

(b) The learning entity has access to a *training set* of objects. Each object in this set is marked as being an "example" (henceforth known as an E-structure) or as being a "nonexample" (N-structure).

(c) The learning entity (henceforth, the LE) has access to a *test set* of objects. The objects in this set are *not* marked as E-structures or N-structures, but the LE does have access to an algorithm (which is timed polynomially in the size of the test set and the complexity of the structure type) which will grade the accuracy of any set of rules the LE may come up with. In other words, given a set of rules for deciding whether a particular structure is of type E or N, each structure in the test set can be evaluated by these rules and as such be categorized as an E or an N. The grading algorithm would simply return a value indicating how accurate the rules turned out to be in their categorizations. It is interesting to note that since the LE is never given the "answers to the questions in the test set" and is only given access to the grading algorithm as an oracle, the same test set can be used more than once in evaluating sets of rules. In other words, we have set things up so that the LE can test itself whenever it chooses.

(d) Since it would be a simple matter for an LE to rapidly code all information in the training set, the degree of accuracy of any set of rules considered as a solution to the problem at hand will only be measured through its performance on the test set – how well it may do on the training set is of no interest to us.

Of course, there are many different methods one could use to rate the accuracy of a set of rules at solving a problem, and in some sense, our work here is independent of any particular choice. However, for definiteness sake, let us settle on certain specifics.

(a) Given a set of rules (henceforth referred to as a separation algorithm) let us view that algorithm as assigning the value 1 to each structure it feels is an E-structure, and a 0 to each structure it feels is an N-structure. Notationally, $C(q) = 1$ ($C(q) = 0$) should be viewed as the separation algorithm C asserting that q is an E-structure (N-structure).

(b) In light of (a) above, let us define two real-valued (rating) functions, r_E and r_N as follows: for any algorithm C, $r_E(C)$ equals the fraction of the total number of E-structures in the test set to which C assigns the value 1, and $r_N(C)$ equals the fraction of the total number of N-structures in the test set to which C assigns the value 1. Symbolically, if s denotes the total number of E-structures in the test set and t denotes the total number of N-structures in the test set, we have

$r_E(C) = |\{q: q \text{ is an E-structure in the test set and } C(q) = 1\}|/s$

and

$r_N(C) = |\{q: q \text{ is an N-structure in the test set and } C(q) = 1\}|/t.$

For example, if C were an algorithm which simply assigned to every structure the value 1, then we would have $r_E(C) = 1$ and $r_N(C) = 1$. Similarly, if C were an algorithm which assigned a 0 to every structure, we would have $r_E(C) = 0$ and $r_N(C) = 0$. Needless to say, these trivial algorithms are of no interest. On the other hand, if C were an algorithm such that $r_E(C) = 1$ and $r_N(C) = 0$, then C is a perfect classifier for our given problem.

(c) It is sometimes convenient to combine the rating functions r_E and r_N into a single function d. Namely, given an algorithm C, let us simply consider the difference $d(C) = r_E(C) - r_N(C)$. Without loss of generality, we can assume that for all algorithms considered, this difference is always nonnegative, for, if necessary, we can replace C with $1 - C$. Clearly, the closer $d(C)$ lies to 1, the better the algorithm C is at the classification problem. Indeed, each of the trivial algorithms mentioned above has a difference of 0, and the perfect classifier has a difference of 1.

In K Square technical report 04.0288 [3], a complete theory is developed concerning the generation of separation algorithms through "study" of training sets. None of that aspect of learning will be discussed here, and in fact, this paper will concern itself not at all with training sets. Rather, we will assume, as a given, the ability to routinely achieve at least some minimal degree of success at developing separation algorithms (through such analysis of known data, through sheer guessing, or through any other means), and will parlay this ability into the creation of an essentially perfect separation algorithm.

In effect, the method we will develop here, stochastic discrimination, is a process which is designed to be carried out at a point in the development of knowledge well beyond where one usually expects to find out anything new. In other words, if we view learning as a pipeline process (fig. 2a) whose final component is an evaluation stage for separation algorithms which come down the line, we will now take the output from this evaluation stage and pipe it into our stochastic discrimination stage, a stage which will produce further separation algorithms (fig. 2b). Something seemingly remarkable will happen here, for even if each of the separation algorithms piped into our stochastic discrimination stage is of low rating, the process will be able to *quickly* construct an essentially perfect separation algorithm from them.

In order to get some sense for what we will be dealing with, let us consider a trivial instance of a rather general example. Suppose that our structure type is a simple one-field record consisting of a single nonnegative integer less than $2s$, and that the E and N records of the test set consist of two disjoint s-element sets of structures. So far as learning is concerned, let us make the conventional components of the pipeline trivial in that we will assume that it is a random process which generates separation algorithms, and that these are then fed to a

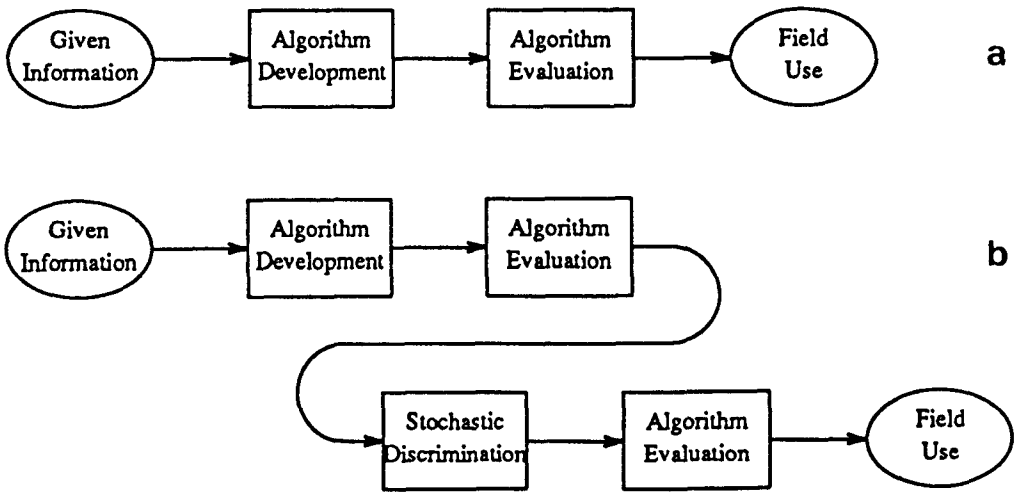

Fig. 2. The "pipeline" process of learning, (a) without and (b) with stochastic discrimination.

stage which evaluates them as described above. It is the stream of randomly generated, evaluated separation algorithms which is piped to our stochastic discrimination stage, and let us say that we are interested in developing a separation algorithm from this stream which is expected to make no more than one error, total, at the task of identifying E-structures as E-structures and N-structures as N-structures; that is, that we are interested in developing a separation algorithm with an expected d-rating of at least $1-(1/s)$.

Of course, there is a rather obvious route to take here – one need only sit at the stream checking the rating of each algorithm which passes by and simply wait until one with a rating at least $1-(1/s)$ passes by. However, given that there exist effectively

4^s

many algorithms, and that of these, only

$2s+1$

many are rated at $1-(1/s)$ or better, one could expect to spend an amount of time of order exponential in s waiting for one to appear.

Our approach is quite different. We will present an explicit 4th degree polynomial, $p(s)$, so that within an amount of time of order $p(s)$, the stochastic discrimination stage of our learning pipeline will produce a separation algorithm with an expected d-rating of at least $1-(1/s)$.

We will discuss this example in some detail later in this paper, but we might point out one further feature of interest. For any real number ε greater than 0, let P_ε represent the probability that during the stochastic discrimination stage, the stream *ever* presents the stage with a separation algorithm with a d-rating greater than ε. Then we will prove that for *any* ε greater than 0, as s increases without bound, P_ε approaches 0. In other words, even though we are able to develop an

essentially perfect separation algorithm through analysis of a stream of (lesser) algorithms, the most likely event is that, in fact, throughout the entire process, the stream never shows us anything much beyond the trivial. Thus it appears as if we are making something out of nothing in developing these algorithms.

2.2. UNIFORM COVER

In the section immediately following this one, we will be defining random variables associated with structures, and will be interested in proving that they all share the same probability density function. The technical device we will use concerns a notion of "uniform cover" as applied to sets of algorithms. That is our focus for now.

On an intuitive level, we would like to develop some sense for what it means for a set of algorithms to cover the set of structures so that among algorithms in the set with given r_E and r_N-ratings, the number giving the value 1 to a particular structure equals the number giving the value 1 to any other structure of the same type (E or N). Clearly, small sets of algorithms providing such a uniform cover are difficult, if not impossible, to construct. For example, a set consisting of two separation algorithms each of which has an r_E-rating equal to some real number x strictly between 0.5 and 1 could not possibly provide an unbiased cover since some proper subset of the collection of E-structures would necessarily be rated 1 by *both* algorithms in the set.

However, as the number of separation algorithms considered together increases, the notion of uniform cover does take on practical meaning. If A is a given set of algorithms, and x and y are two given real numbers, let us define A_{xy} to be

$$\{C \in A : r_E(C) = x \text{ and } r_N(C) = y\}.$$

We can then define A to be *E-uniform* if given any two E-structures p and q, and any two real numbers x and y,

$$\sum_{C \in A_{xy}} C(p) = \sum_{C \in A_{xy}} C(q).$$

Similarly, we have a notion of *N-uniform* using the same definition with "E-structures" replaced with "N-structures". A set of selection algorithms is said to be a *uniform cover*, if it is both E-uniform and N-uniform.

In order to approach the problem of existence of such sets of algorithms, we must first look at another, purely combinatorial, view of the notion of uniform cover. A given separation algorithm C has associated with it a specific set of E-structures, E_C, namely, those E-structures to which C assigns a value of 1. Similarly, it has an associated set of N-structures, N_C, namely those N-structures to which C assigns a value of 1. Indeed, since we will never be getting involved with the internal details of any particular algorithm (other than the detail that it

operates in polynomial time), any two algorithms C and D such that $E_C = E_D$ and $N_C = N_D$ will be viewed, for the purposes of this paper, as being identical. Thus we will feel free to equate algorithms with pairs of sets of structures, and specifically, if X and Y are the two sets equated with an algorithm C, then if $i(j)$ is the size of the intersection of $X(Y)$ with the set of all E-structures (N-structures), then $r_E(C)$ ($r_N(C)$) is equal to i/s (j/t).

Using this duality between algorithms and sets, let us consider any pair of nonnegative integers i less than or equal to s and j less than or equal to t, let us define x to be i/s and y to be j/t, and let us consider the set Q_{xy} of *all* algorithms $C = (E_C, N_C)$ such that E_C contains i-many elements and N_C contains j-many elements. Then we must have that for any particular E-structure q,

$$\sum_{C \in Q_{xy}} C(q) = \binom{s-1}{i-1}\binom{t}{j} = \left(\frac{(s-1)!}{(i-1)!(s-i)!}\right)\left(\frac{t!}{j!(t-j)!}\right), \tag{0}$$

and for any particular N-structure q,

$$\sum_{C \in Q_{xy}} C(q) = \binom{t-1}{j-1}\binom{s}{i} = \left(\frac{(t-1)!}{(j-1)!(t-j)!}\right)\left(\frac{s!}{i!(s-i)!}\right). \tag{1}$$

Naturally, this shows that the set of all algorithms constitutes a uniform cover, but given the nature of our notion of uniform cover, we have also, clearly, established that for any possible set T of d-ratings, the set of all separation algorithms C such that $d(C)$ is a member of T is a uniform cover.

2.3. THE BASE VARIABLES

Although one tends to think of the application of statistics in contexts such as this as involving sampling within the population of structures, the basis of our approach will involve sampling within the population of separation algorithms. Indeed, our key method for attacking the problem will make use of random variables associated with such sampling.

If one were to simply "guess" at separation algorithms for our problem one would obviously produce some algorithms which were better and some which were worse. Needless to say, the higher the d-ratings one required, the less frequently one would expect to find acceptable algorithms through guessing. For example, in the situation discussed above, it is only after an exponential amount of time that one might expect to find an algorithm with a d-rating greater than or equal to $1 - (1/s)$. However, it is also clear that if we lower our sights, and rather than look for algorithms C such that $d(C) \geqslant 1 - (1/s)$, just look for algorithms C such that $d(C) > 0$, we can expect to see them regularly.

We will attack this question in some detail later in section 3, but until further notice, let us assume that we have access to some polynomially (in s, t, and structure complexity) timed process for producing separation algorithms C whose

degree of goodness is not only greater than 0 but is bounded away from 0 by some positive real number $1/\nu$. In other words, we will assume that there exists a real number $1/\nu > 0$ and a polynomially timed process for producing a stream of rated algorithms sampling the population of all algorithms C such that $d(C) > 1/\nu$. (The notion of creating such a process is a most interesting area in its own right, and, as mentioned above, it is the topic discussed in [3]. And while, in some sense, it involves the half of the learning paradigm concerned with "study" of the training set, later in this paper we will return to our example, introduced earlier, which turns out to satisfy this assumption independent of any consideration of training set. The example is of special interest since, despite its simplicity, its natural solution appears to require an exponential (or at least NP) amount of time. We will produce a solution within an amount of time polynomial in s.)

Let us denote by $\mathscr{S}$ the set of all separation algorithms C such that $d(C) > 1/\nu$. Then if we consider $\mathscr{S}$, under the counting measure, as a sample space, we can associate with any structure q of our separation problem the random variable X^q, which, at an element C of $\mathscr{S}$ takes the value

$$C(q)/r_{\mathrm{E}}(C).$$

LEMMA 1

If p and q are two structures of the same category (E or N), then X^p and X^q have the same probability density functions.

Proof

Let us introduce the following notation: if r is a given real number, then $\langle r \rangle$ denotes the least integer greater than r, and $[r]$ denotes the greatest integer less than r. Then using our analysis above concerning uniform cover, and in particular, in light of eqs. (0) and (1), it follows that for *any* E-structure q, the number of points where X^q takes the value 0 is

$$\sum_{i=\langle s/\nu \rangle}^{s} \sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s-1}{i}\binom{t}{j},$$

and for every $i > s/\nu$, the number of points where X^q takes the value s/i is

$$\sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s-1}{i-1}\binom{t}{j}.$$

Similarly, for *any* N-structure q, the number of points where X^q takes the value 0 is

$$\sum_{i=\langle s/\nu \rangle}^{s} \sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s}{i}\binom{t-1}{j},$$

and for every $i > s/v$, the number of points where X^q takes the value s/i is

$$\sum_{j=0}^{[t(i/s-1/v)]} \binom{s}{i}\binom{t-1}{j-1}.$$

Since none of these expressions involves q, the lemma is proved. □

Now that we know that random variables associated with structures of a particular category all have the same probability density function, we know that there are at most two possible values for expectations of such variables. In point of fact, there are *exactly* two values, and this will be extremely important for our subsequent work. Before we prove this, however, let us explicitly state (without proof) a trivial proposition which we will need both here and later in this paper.

PROPOSITION

Let $\mathscr{X}$ be a finite sample space with counting measure v, and let $\mathbb{K}$ be a partition of $\mathscr{X}$. Then if X is a random variable defined on $\mathscr{X}$, and if, for each subset $\mathscr{U}$ of $\mathscr{X}$ in $\mathbb{K}$, $X^{\mathscr{U}}$ denotes the restriction of X to $\mathscr{U}$, then

$$E(X) = \sum_{\mathscr{U}\in\mathbb{K}} \frac{v(\mathscr{U})}{v(\mathscr{X})} E(X^{\mathscr{U}}).$$

We are now in a position to prove that E-structure random variables and N-structure random variables have *distinct* expectations.

LEMMA 2

If q is an E-structure, then $E(X^q) = 1$. If q is an N-structure, then $E(X^q) < 1 - (1/v)$.

Proof

Let $i \le s$ and $j \le t$ satisfy $i/s - j/t > 1/v$, and let $\mathscr{D}$ denote the set consisting of all algorithms in $\mathscr{S}$ with an r_E-rating of i/s and an r_N-rating of j/t. Then we immediately see that for any E-structure q, the expectation of the variable X^q restricted to the sample space $\mathscr{D}$ is

$$\frac{\binom{t}{j}\binom{s-1}{i-1}\left(\frac{i}{s}\right)^{-1}}{\binom{t}{j}\binom{s}{i}} = \frac{\left(\frac{t!}{j!(t-j)!}\right)\left(\frac{(s-1)!}{(i-1)!(s-i)!}\right)\left(\frac{s}{i}\right)}{\left(\frac{t!}{j!(t-j)!}\right)\left(\frac{s!}{i!(s-i)!}\right)} = \frac{\binom{t}{j}\binom{s}{i}}{\binom{t}{j}\binom{s}{i}} = 1.$$

Thus by the proposition above, for any E-structure q, the expectation of X^q is 1. The situation for N-structure variables is different. For if q is an N-structure, and $\mathscr{D}$ denotes the set consisting of all algorithms in $\mathscr{S}$ with an r_E-rating of i/s

and an r_N-rating of j/t, we immediately see that for any N-structure q, the expectation of the variable X^q restricted to the sample space $\mathscr{D}$ is

$$\frac{\binom{s}{i}\binom{t-1}{j-1}\left(\frac{i}{s}\right)^{-1}}{\binom{s}{i}\binom{t}{j}} = \frac{\left(\frac{s!}{i!(s-i)!}\right)\left(\frac{(t-1)!}{(j-1)!(t-j)!}\right)\left(\frac{s}{i}\right)}{\left(\frac{s!}{i!(s-i)!}\right)\left(\frac{t!}{j!(t-j)!}\right)} = \frac{\binom{t}{j}\binom{s}{i}\left(\frac{js}{it}\right)}{\binom{t}{j}\binom{s}{i}} = \left(\frac{js}{it}\right).$$

Since every C in $\mathscr{S}$ has the property that $d(C) > 1/\nu$, we must have that

$$\frac{i}{s} - \frac{j}{t} > \frac{1}{\nu},$$

and so,

$$\frac{js}{it} < 1 - \frac{s}{i\nu}.$$

Since $i \leq s$, we can say that

$$\frac{js}{it} < 1 - \frac{1}{\nu}.$$

We may thus conclude, by the proposition above, that for any N-structure q, the expectation of X^q is less than $1 - 1/\nu$. □

2.4. THE STOCHASTIC PROCESS

The way stochastic discrimination operates is as follows:
Suppose q is a given pattern structure. Then we will undertake a sequence of independent trials over our sample space, and for each i, let us denote by X_i^q the random variable corresponding to X^q associated with the ith trial. Furthermore, for each n, let M_n^q denote the random variable

$$\left(\sum_{i=1}^{n} X_i^q\right)/n.$$

Then by the Law of Large Numbers, as n increases without bound, the probability that M_n^q lies close to the expectation of X^q approaches 1. But since the expectation of X^q is 1 for E-structures q and is less than $1 - (1/\nu)$ for N-structures q, we should be able to tell, by taking n "large enough" just which of the two possible expectations M_n^q is, with "high" probability, "close to", and hence, should be able to tell, with "high" probability, whether q is an E-structure or an N-structure. In practice, if we simply take a random sample with replacement, $\mathscr{S}$, of "large enough" size n, then $\mathscr{S}$ can be viewed as representing the n-many independent random variables X_1^q, $X_2^q, \ldots, X_n^q$, and thus the sample mean can be viewed as representing M_n^q. *And by the Law of Large Numbers, if that sample mean were within* $1/2\nu$ *of* 1, *we could conclude with "high" probability that q was an E-structure, and if that sample mean were*

not within $1/2v$ *of* 1, *we could conclude with "high" probability that q was an N-structure*. Of course, all of this is heavily dependent on just what constitutes "large enough" as applied to sample size, but we will provide an explicit polynomial function such that if n is larger than the bound provided by this function, then that n is "large enough" to make the algorithm work.

There are different levels of sophistication possible in precisely formulating the algorithm described above. These levels are somewhat related to just how sophisticated a version of the law of large numbers one chooses to use. An initial result follows with a simple use of Chebyshev's inequality:

LEMMA 3

There exists a polynomial $P(x, y, z)$ such that for any positive h, k, and s, if $m \geq P(h, k, s)$,

$$Pr\left(\left| M_m^q - E(X^q) \right| < 1/k\right) > 1 - (1/h).$$

Proof

Let $E(X^q)$ and σ^2 denote the mean and variance, respectively, of X^q. Then $E(X^q)$ and σ^2/m are the mean and variance, respectively of M_m^q. By Chebyshev's inequality,

$$Pr\left(\left| M_m^q - E(X^q) \right| \geq 1/k\right) \leq \frac{\sigma^2 k^2}{m},$$

and so,

$$Pr\left(\left| M_m^q - E(X^q) \right| < 1/k\right) > 1 - \frac{\sigma^2 k^2}{m}.$$

Since the range of X^q is contained in $[0, s]$, $\sigma < s$. Thus if we let

$$P(x, y, z) =^{df} xy^2z^2,$$

then for any $m \geq P(h, k, s)$,

$$\frac{k^2\sigma^2}{m} \leq \frac{1}{h},$$

and so

$$Pr\left(\left| M_m^q - E(X^q) \right| < 1/k\right) > 1 - (1/h). \quad \square$$

To flesh all of this out somewhat, let us return to our example considered earlier where pattern structures consist of single-field records containing non-negative integers less than $2s$. In a later section, this example will be covered in detail, and in particular, we will set things up so that the expectations of E-structure base random variables and N-structure base random variables lie further than $1/(\sqrt{2s-1})$ apart. Thus by lemma 3, if n is larger than

$P(2s, 2\sqrt{2s-1}, s) = 16s^4 - 8s^3$, given *any* structure q, if we were to simply calculate the mean of any random sample with replacement having size n viewed as representing M_n^q, then if that mean were within $1/(2\sqrt{2s-1})$ of 1 the likelihood that q *was not* an E-structure would be less than 1 in $2s$, and similarly, if the mean were within $1/(2\sqrt{2s-1})$ of 1, the likelihood that q was an N-structure would be less than 1 in $2s$. *In other words, we have just described a separation algorithm for our problem which operates in polynomial time, and which has an expected d-rating greater than* $1 - (1/s)$.

2.5. THE SEPARATING DENSITY FUNCTIONS

Let us look at the dynamics of the situation. For any specific sample size, n, we have two probability density functions, one associated with E-structures (i.e., the density function for M_n^q for any E-structure q), and the other associated with N-structures (i.e., the density function for M_n^q for any N-structure q). The means of these functions are some fixed distance apart from one another, and by the Central Limit Theorem, for large enough n, the density functions are essentially normal (see fig. 3).

But as sample size (n) increases, we also have that the variances of the density functions approach 0, and in this fact lies the reason stochastic discrimination works. For this means that as sample size increases, the two density functions *separate from one another* eventually to the point where their overlap is negligible (fig. 4).

As a result, if we are dealing with a large enough sample, given the value of one of the density functions on a (random) point taken from their common domains, it is quite clear which of the functions (the E-structure function or the N-structure function) was used to produce the value.

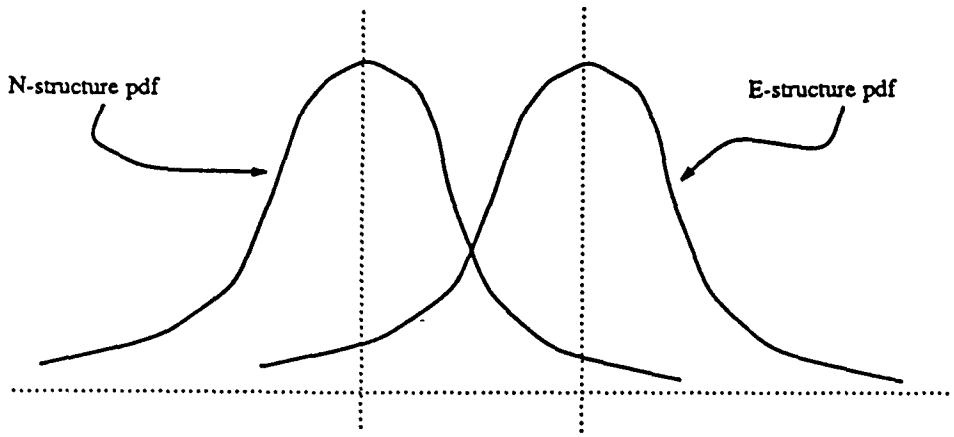

Fig. 3. Two overlapping pdf's with separated means.

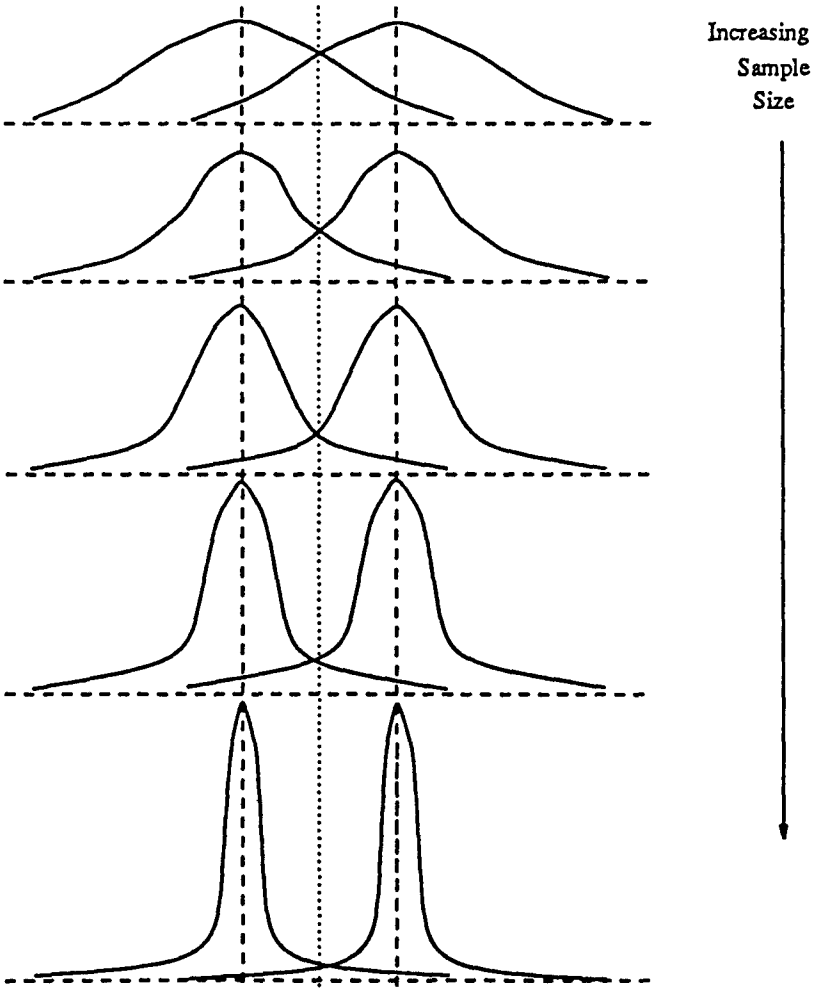

Fig. 4. With increasing sample size n the overlap of the pdf's becomes negligible.

Thus, if we define the *separating threshold, t*, to be the mean of the means of the E-structure and N-structure density functions, given a (random) sequence from our sample space of separation algorithms, we have, for each n, an instance of the stochastic algorithm, SA_n, which simply calls a structure q an E-structure if the value of M_n^q on the length-n initial segment of the sequence is greater than t. And by the lemma above, the expected accuracy of SA_n will be greater than $1-(1/h)$ if n is taken larger than some polynomial function of h (and problem complexity).

2.6. THE DUALITY LEMMA

In the previous subsection, we introduced what might be called the *stochastic algorithm* for separating points in a space, and we discussed the performance of

this algorithm in terms of "expected d-rating" for large samples. We now wish to formulate a precise picture of the actual d-rating of stochastic algorithms. This will enable us to prove a surprising, and extremely powerful result concerning the stability of stochastic algorithms as one moves from performance on training set to performance on test set to performance in field use. Basically, we will prove that if the algorithms in the sample from which a stochastic algorithm is built are all stable in their performance in going from training data to test data, then the stochastic algorithm will be stable as well. And since, given some care in constructing training data, it is easy to find stable (though, perhaps, poor) separation algorithms, the practical implications of this are enormous. And what is surprising, is that we have here a method for building arbitrarily complex solutions to problems without the usual worries of training-set specificity and the inevitable instability it creates.

Given a fixed structure q, we have been considering the random variable M_i^q, and have viewed it as being defined on an i-fold product of spaces of separation algorithms. But if we were to fix a member of this i-fold product of spaces of separation algorithms, M_i^q could alternatively be viewed as a random variable defined on the space of structures. And by casting things in this light, we have a mechanism for precisely evaluating the d-rating of the stochastic algorithm built from that (fixed) member of the i-fold product. For let R_E (R_N) denote M_i^q as a function of q defined on the sample space of E-structures (N-structures). Then it is immediate, given the definitions of r_E and r_N, that the r_E-rating (r_N-rating) of the derived stochastic algorithm is just the area under the density function of $R_E(R_N)$ to the right of the separating threshold, and hence, the d-rating of the derived stochastic algorithm is just the area to the right of the separating

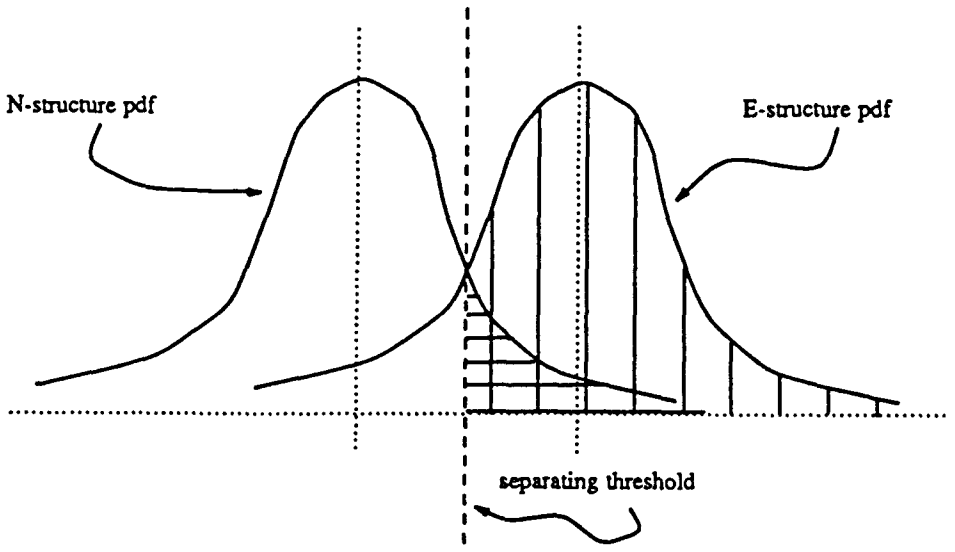

Fig. 5. Illustration of the use of the separating threshold.

threshold bounded from above by the density function of R_E and from below by the density function of R_N (see fig. 5).

In order to make use of these observations, we must come up with usable descriptions of the functions R_E and R_N. This is the role of the duality lemma.

We begin with some notation. Given $i \leq s$ and $j \leq t$, let

$$\mathscr{S}_{ij} =^{df} \left\{ C\colon r_E(C) = \frac{i}{s} \text{ and } r_N(C) = \frac{j}{t} \right\}.$$

Fix some integer $n > 0$ and sequence $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ such that for each i,

$$\frac{a_i}{s} - \frac{b_i}{t} > \frac{1}{\nu},$$

and let us define $\mathscr{Q}$ to be

$$\prod_{i=1}^{n} \mathscr{S}_{a_i b_i}.$$

Let $\mathscr{P}$ denote the space of all structures, and let us denote by M_n the map from the product $\mathscr{P} \times \mathscr{Q}$ into the reals which at any point $(q, \langle C_1, C_2, \ldots, C_n \rangle)$ takes the value

$$M_n^q(\langle C_1, C_2, \ldots, C_n \rangle).$$

LEMMA 4

Let $\mathscr{K}$ be either the set of all E-structures or the set of all N-structures, and fix points q in $\mathscr{K}$ and $\mathbf{C}$ in $\mathscr{Q}$. Let Y_q denote the restriction of M_n to $\{q\} \times \mathscr{Q}$, and let $Z_{\mathbf{C}}$ denote the restriction of M_n to $\mathscr{K} \times \{\mathbf{C}\}$. Then Y_q and $Z_{\mathbf{C}}$ have the same pdf's.

Proof

Consider a possible value in the range of M_n. It is of the form

$$\frac{s}{n} \sum_{i=1}^{n} \varepsilon_i (a_i)^{-1},$$

where, for each i, ε_i is either 0 or 1. What is the probability that Y_q takes this value? Clearly the probability that ε_i is equal to 1, that is, that s/a_i gets contributed to the sum, is just the probability that q is accepted by an algorithm in $\mathscr{S}_{a_i b_i}$. If $\mathscr{K}$ is equal to $\mathscr{E}$, this probability is

$$\frac{\binom{t}{b_i}\binom{s-1}{a_i-1}}{\binom{t}{b_i}\binom{s}{a_i}} = \frac{\left(\frac{t!}{b_i!(t-b_i)!}\right)\left(\frac{(s-1)!}{(a_i-1)!(s-a_i)!}\right)\left(\frac{s}{a_i}\right)\left(\frac{a_i}{s}\right)}{\left(\frac{t!}{b_i!(t-b_i)!}\right)\left(\frac{s!}{a_i!(s-a_i)!}\right)} = \frac{\binom{t}{b_i}\binom{s}{a_i}\left(\frac{a_i}{s}\right)}{\binom{t}{b_i}\binom{s}{a_i}}$$

$$= \frac{a_i}{s}.$$

Thus the probability that Y_q takes the value

$$\frac{s}{n}\sum_{i=1}^{n}\varepsilon_i(a_i)^{-1}$$

is

$$\prod_{i=1}^{n}\left\{\varepsilon_i\left(\frac{a_i}{s}\right)+(1-\varepsilon_i)\left(1-\frac{a_i}{s}\right)\right\}=s^{-n}\prod_{i=1}^{n}(\varepsilon_i a_i+(1-\varepsilon_i)(s-a_i)).$$

(By similar reasoning, if $\mathscr{K}$ is equal to $\mathscr{N}$, the probability that Y_q takes the value

$$\frac{s}{n}\sum_{i=1}^{n}\varepsilon_i(a_i)^{-1}$$

is

$$s^{-n}\prod_{i=1}^{n}(\varepsilon_i b_i+(1-\varepsilon_i)(s-b_i)).)$$

We now consider the variable $Z_{\mathbf{C}}$. Suppose $\mathscr{K}$ is equal to $\mathscr{E}$. Then since the ith coordinate of $\mathbf{C}$ has an r_{E}-rating of a_i/s, the probability that ε_i equals 1 in the sum above is a_i/s. Thus arguing as above, the probability that $Z_{\mathbf{C}}$ takes the value

$$\frac{s}{n}\sum_{i=1}^{n}\varepsilon_i(a_i)^{-1}$$

is

$$s^{-n}\prod_{i=1}^{n}(\varepsilon_i a_i+(1-\varepsilon_i)(s-a_i)).$$

Using r_{N}-ratings in place of r_{E}-ratings, if $\mathscr{K}$ were equal to $\mathscr{N}$, the probability that $Z_{\mathbf{C}}$ takes the value

$$\frac{s}{n}\sum_{i=1}^{n}\varepsilon_i(a_i)^{-1}$$

would be

$$s^{-n}\prod_{i=1}^{n}(\varepsilon_i b_i+(1-\varepsilon_i)(s-b_i)).$$

□

Just how does this lemma prove the stability of stochastic algorithms? Specifically, suppose that we have a given training set TR, a given test set TE, and that our stochastic algorithm is built from n-many stable separation algorithms, that is, from separation algorithms which have the same r_{E} and r_{N}-ratings when evaluated in either TR or TE. Let $a_1, a_2, \ldots, a_n$ be the sequence of such r_{E}-ratings for these separation algorithms, and let $b_1, b_2, \ldots, b_n$ be the sequence of

r_N-ratings. Then it is clear that for any E-structure q (whether it is in TR or TE), the pdf of Y_q (in the notation of lemma 4) depends only on the sequence $\{a_i\}$. Thus by lemma 4, the random variable Z_C defined on the E-structures in TR has the same pdf as the random variable Z_C defined on the E-structures in TE. But the r_E-rating of the stochastic algorithm is equal to the area under this pdf to the right of the separating threshold, and so we have shown that the r_E-rating of the stochastic algorithm is the same whether it is evaluated in TR or in TE. Arguing similarly with N-structure and the sequence $\{b_i\}$, we see that the r_N-rating of the stochastic algorithm is the same whether it is evaluated in TR or TE. Thus the stochastic algorithm is stable.

2.7. A REMARK CONCERNING CONVERGENCE TO NORMALITY

In order for the calculation of d-ratings of stochastic algorithms using the method implied by the previous section to have practical value, we must be able to explicitly work with relevant pdf's. Of course, the duality lemma presents us with a complete description of them, but it is not hard to see that the form they take is extremely difficult to work with. Fortunately, we have central limit theorems, and as a result, these distributions approach normality. The question for us, however, concerns just how quickly the distributions approach normality. For unless things happen polynomially fast, the normal approximations are of little practical use.

In this section, we prove that convergence does, indeed, happen polynomially fast; however, we will make no attempt to analyze the specific distributions at hand, nor will we make any attempt to get the sharpest estimates possible. *We are simply interested in demonstrating the* ***polynomial*** *nature of the convergence and so will rely on general limit theorems.* (A detailed analysis of the normal approximations to our specific pdf's is saved for a later work.)

The general tool we will use here is the so called Cramer–Berry–Esseen theorem:

THEOREM

If $X_1, X_2, \ldots, X_m$ are independent, identically distributed random variables with expectation 0, variance σ^2, and finite third moment, and if S_m represents the sum of the X_i, then

$$\left| Pr\left(\frac{S_m}{\sqrt{m}} \leq x \right) - \Phi_\sigma(x) \right| \leq \frac{K_{CBE} E\left(|X_1|^3 \right)}{\sigma^3 \sqrt{m}} \tag{2}$$

for some constant K_{CBE} less than 3.

For a start, our variables do not have expectation 0, and so we must use the above theorem with $E(|X_1|^3)$ replaced with $E(|X_1 - \mu|^3)$ (where μ is the

expectation of the X_i). Second of all, since we are interested in the distribution of S_m/m, and not that of $S_m/\sqrt{m}$, we must rephrase (2) as

$$\left| Pr\left(\frac{S_m}{m} \leq \frac{x}{\sqrt{m}} \right) - \Phi_\sigma(x) \right| \leq \frac{K_{\text{CBE}} E\left(| X_1 - \mu |^3 \right)}{\sigma^3 \sqrt{m}}$$

or

$$\left| Pr\left(\frac{S_m}{m} \leq y \right) - \Phi_\sigma\left(\sqrt{m}\, y\right) \right| \leq \frac{K_{\text{CBE}} E\left(| X_1 - \mu |^3 \right)}{\sigma^3 \sqrt{m}}.$$

With a simple change of variables within integrals we get

$$\Phi_\sigma\left(\sqrt{m}\, y\right) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\sqrt{m}\, y} e^{-(x^2/2\sigma^2)}\, dx = \frac{\sqrt{m}}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} e^{-((t\sqrt{m})^2/2\sigma^2)}\, dt = \Phi_{\sigma/\sqrt{m}}(y).$$

Thus we have

$$\left| Pr\left(\frac{S_m}{m} \leq y \right) - \Phi_{\sigma/\sqrt{m}}(y) \right| \leq \frac{K_{\text{CBE}} E\left(| X_1 - \mu |^3 \right)}{\sigma^3 \sqrt{m}}, \tag{3}$$

in other words, the maximum difference between the distribution of S_n/n and the normal approximation to it is bounded in absolute value by

$$\frac{K_{\text{CBE}} E\left(| X_1 - \mu |^3 \right)}{\sigma^3 \sqrt{m}} \tag{4}$$

for some constant K_{CBE} less than 3.

Thus, in order to establish that our distributions converge to normality at a polynomial rate, we need only prove

LEMMA 5

There exists a polynomial $R(x, y, z)$ such that for any s, v, w, and structure q, if $m \geq R(s, v, w)$, then

$$\frac{K_{\text{CBE}} E\left(| X^q - \mu |^3 \right)}{\sigma^3 \sqrt{m}} < \frac{1}{w}.$$

Proof

The key for us is to find an upper bound estimate, $U(s, v)$ on

$$\frac{E\left(| X^q - \mu |^3 \right)}{\sigma^3}.$$

For if we can find such a $U(s, v)$ which is polynomial in s and v, we would be able to easily construct our desired polynomial $R(x, y, z)$. So let X^q be one of

our (previously defined) variables (having expectation μ). Then arguing as we did in the proof of lemma 1, we have that

$$E(|X^q-\mu|^3)=\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\left(\binom{s-1}{i}\binom{t}{j}\mu^3+\binom{s-1}{i-1}\binom{t}{j}\left(\frac{s}{i}-\mu\right)^3\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}, \tag{5}$$

and that

$$\sigma^2=\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\left(\binom{s-1}{i}\binom{t}{j}\mu^2+\binom{s-1}{i-1}\binom{t}{j}\left(\frac{s}{i}-\mu\right)^2\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}. \tag{6}$$

Needless to say, each of these expressions, let alone the quotient

$$\frac{E(|X^q-\mu|^3)}{\sigma^3},$$

is rather difficult to evaluate, but if we employ a simple trick used earlier, we will be able to make some headway. For given a finite set T of numbers for which one wishes to find the mean, one can first partition T into some number of disjoint subsets, find the mean of the numbers in each set in the partition, and then take an average (weighted by the *size* of each set in the partition) of these means to find the average of the numbers in T. This was exactly the point of our trivial proposition of section 4. In the case of (5) and (6), we partition the space of all algorithms C such that $d(C)>1/\nu$ into those subspaces where $r_E(C)=i/s$ is constant, and using the same technique we used in the proof of lemma 2 for simplifying binomial coefficients, we have

$$\begin{aligned}\mathit{Numerator}(i)&=\frac{\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s-1}{i}\binom{t}{j}\mu^3+\binom{s-1}{i-1}\binom{t}{j}\left(\frac{s}{i}-\mu\right)^3}{\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}\\&=\frac{\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}\left(\left(\frac{s-i\mu}{i}\right)^3\frac{i}{s}+\left(\mu^3\frac{s-i}{s}\right)\right)}{\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}\\&=\frac{(s-i\mu)^3+i^2(s-i)\mu^3}{i^2s},\end{aligned} \tag{7}$$

and (working, for the moment, with the simpler expression, σ^2, rather than the true denominator σ^3)

$$Denominator(i) = \frac{\sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s-1}{i}\binom{t}{j}\mu^2 + \binom{s-1}{i-1}\binom{t}{j}\left(\frac{s}{i}-\mu\right)^2}{\sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s}{i}\binom{t}{j}}$$

$$= \frac{\sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s}{i}\binom{t}{j}\left(\left(\frac{s-i\mu}{i}\right)^2\frac{i}{s} + \left(\mu^2\frac{s-i}{s}\right)\right)}{\sum_{j=0}^{[t(i/s-1/\nu)]} \binom{s}{i}\binom{t}{j}}$$

$$= \frac{(s-i\mu)^2 + i(s-i)\mu^2}{is}. \tag{8}$$

The virtue of these expressions is this. If we can calculate a *largest* value for any *Numerator(i)*, it will provide an upper bound for $E(|X^q-\mu|^3)$, and if we can calculate a *smallest* value for any *Denominator(i)*, raising it to the 3/2 power will provide a lower bound for σ^3, *and the quotient of these two bounds will provide a desired upper bound for*

$$\frac{E(|X^q-\mu|^3)}{\sigma^3}. \tag{9}$$

In the case of *Numerator(i)*, things are quite straightforward. For since s is fixed, and i varies from $\langle s/\nu\rangle$ to s, the largest value *Numerator(i)* can take (which clearly happens when i is as small as possible, that is, when $i=\langle s/\nu\rangle$) is

$$Numerator(\langle s/\nu\rangle) = \frac{\left(s-\frac{s\mu}{\nu}\right)^3 + \left(\frac{s}{\nu}\right)^2\left(s-\frac{s}{\nu}\right)\mu^3}{\left(\frac{s}{\nu}\right)^2 s} = \frac{(\nu-\mu)^3 + (\nu-1)\mu^3}{\nu}.$$

Taking into account the fact that μ must lie between 0 and 1, we thus have that for any variable X^q,

$$E(|X^q-\mu|^3) \le \frac{\nu^3+(\nu-1)}{\nu}. \tag{10}$$

The situation with *Denominator(i)* is different. For given that s is fixed, and that i varies from $\langle s/\nu\rangle$ to s, our temptation is to say that the smallest value *Denominator(i)* can take is when i is as large as possible, that is, when $i=s$. In this case we would have

$$Denominator(s) = \frac{(s-s\mu)^2 + s(s-s)\mu^2}{s^2} = (1-\mu)^2. \tag{11}$$

However, if $\mu = 1$, which happens with random variables associated with E-structures, this denominator will be 0, and hence will be useless in forming a quotient to provide an upper bound for (9). So while, in the case of N-structure random variables, we have just shown that

$$\sigma^3 \geq (1-\mu)^3 > 0, \tag{12}$$

providing a nonzero lower bound for σ^3, what do we do for E-structure random variables, where $\mu = 1$? Well, it is easy enough to derive an estimate for the smallest *nonzero* value which *Denominator(i)* can take. This clearly happens when $i = s-1$, and we have

$$\textit{Denominator}(s-1) = \frac{(s-(s-1))^2 + (s-1)(s-(s-1))}{(s-1)s} = \frac{1}{s-1}. \tag{13}$$

But how can this be used in calculating a *nonzero* lower bound for σ^3 given that the mean over one of the sets in our partition (the set where $i = s$) is 0? We must argue as follows. Without loss of generality, we may assume that s and v are so large that

$$1 - \frac{1}{v} - \frac{2}{s} > \frac{1}{2} \quad \text{and} \quad s > 5. \tag{14}$$

Let us consider the values for the means of sets in our partition as calculated in (8) for $i = s$, $i = s-1$, and $i = s-2$. For $i = s$, the mean is 0 (this is the only set in the partition having a mean of 0) and the size of the set having this mean of 0 is

$$\sum_{j=0}^{[t(1-1/v)]} \binom{s}{s}\binom{t}{j} = \sum_{j=0}^{[t(1-1/v)]} \binom{t}{j}. \tag{15}$$

For $i = s-1$, as calculated in (13), the mean is $\textit{Denominator}(s-1) = 1/(s-1)$, and the size of the set having this mean is

$$\sum_{j=0}^{[t(1-1/v-1/s)]} \binom{s}{s-1}\binom{t}{j} = \sum_{j=0}^{[t(1-1/v-1/s)]} \binom{s}{1}\binom{t}{j}. \tag{16}$$

Finally, for $i = s-2$, the mean is $2/(s-2)$, and the size of the set having this mean is

$$\sum_{j=0}^{[t(1-1/v-2/s)]} \binom{s}{s-2}\binom{t}{j} = \sum_{j=0}^{[t(1-1/v-2/s)]} \binom{s}{2}\binom{t}{j}. \tag{17}$$

We claim that the three terms given in (15), (16), and (17) form a nondecreasing sequence. For in light of (14), even the sum with the fewest terms, namely that in (17), still has more than $t/2$-many terms. And given the symmetry of the binomial coefficients about $j = t/2$, and in particular, the fact that

for *every* j, $0 \leq j \leq t$, $\binom{t}{j} = \binom{t}{t-j}$,

we see that for every value of j which appears in (15) and not in (16), there is a term

$$s\binom{t}{t-j}$$

in (16) whose value is clearly greater than

$$\binom{t}{t-j}+\binom{t}{j}$$

(since $s>2$). This shows that the expression in (15) is no greater than the expression in (16). By a similar argument, we have that for every value of j which appears in (16) and not in (17), there is a term

$$\frac{s(s-1)}{2}\binom{t}{t-j}$$

in (17) whose value (given (14)) is clearly greater than

$$s\binom{t}{t-j}+s\binom{t}{j}$$

(since $s>5$). This shows that the expression in (16) is no greater than the expression in (17). Given that the sizes of these three sets in the partition are nondecreasing, and given that their means are 0, $1/(s-1)$, and $2/(s-2)$ (where $2/(s-2)$ is more than twice as large as $1/(s-1)$), we have shown that the mean over the union of these three sets in our partition is greater than $1/(s-1)$. Since $1/(s-1)$ is the smallest *positive* value of any mean of any set in the full partition, we have thus established the fact that the mean of the *entire* set is greater than $1/(s-1)$. Putting all of this together, we have that for E-structure random variables

$$\sigma^3 \geq (s-1)^{-3/2} > 0. \tag{18}$$

Now let us try to combine the two cases of E-structure and N-structure variables. We have shown earlier that the mean μ for N-structure variables is less than or equal to $1-(1/\nu)$, and so $(1-\mu)^3$ is at least as large as

$$\frac{1}{\nu^3}.$$

Since $1/(s-1)^{3/2}$ and $1/\nu^3$ both lie in the interval (0, 1], we thus have that for either E-structure variables or N-structure variables,

$$\sigma^3 \geq \frac{1}{\nu^3(s-1)^{3/2}} > 0. \tag{19}$$

Combining (10) and (19), we now have that for *any* variable X^q,

$$\frac{E(|X^q-\mu|^3)}{\sigma^3} \leq \left(\nu^3+(\nu-1)\right)\nu^2(s-1)^{3/2}.$$

Given this polynomial upper bound, $U(s, v)$, if we now just define R by

$$R(s, v, w) = K_{\mathrm{CBE}}^2(w+1)^2(U(s, v))^2,$$

our proof is complete. □

2.8. PERFECT STOCHASTIC ALGORITHMS

The d-rating of an algorithm basically presents one with a measure of the probability that the algorithm will correctly classify any given structure. Thus, given a particular algorithm and a particular set of structures, we can calculate the probability that the algorithm is, in fact, perfect.

Suppose we are dealing with n-many structures, and that we have built a stochastic algorithm whose probability of misclassifying any given structure is less than $1/kn$. Then the probability that this algorithm makes no errors in classifying all n-many structures is

$$\left(1 - \frac{1}{kn}\right)^n,$$

which, if n is large enough, is approximately equal to

$$e^{-(1/k)}.$$

Thus if we take k to be equal, say, to 2, then the probability this algorithm is perfect is certainly greater than $1/2$.

Suppose, now, that we were to build, independently, m-many such stochastic algorithms. Then the probability that *none of them* was perfect would be less than 2^{-m}. In other words, within an amount of time polynomial in problem complexity and m, the probability that we fail to produce a perfect stochastic algorithm is exponentially small. This observation will be exploited when we discuss complexity theory.

3. An example

Let us now return to the example considered earlier where the structure type is a simple one-field record consisting of a single nonnegative integer less than $2s$, and the E and N records of the test set consist simply of two disjoint s-element sets of structures.

In order to make things as easy as possible, let us begin with a process which "randomly" produces and rates trivial Boolean algorithms (i.e., Boolean expressions based on atoms of the form $v = k$, where v is the (unique) record-variable and k is a specific nonnegative integer less than $2s$). Clearly, it is easy to construct such processes which operate polynomially in s, and let us assume that we have set such a process in motion producing a stream of separation al-

gorithms. It is our goal to take this stream and come up with an algorithm having an expected d-rating greater than $1 - (1/s)$.

Our first step is to put a simple "wait and see" filter on the stream which only lets pass those algorithms which identify precisely s-many elements from the set of all structures. Such algorithms, which we will henceforth refer to as *symmetric algorithms*, are the ones most likely to be seen in our random stream, each of whose algorithms identifies somewhere between 0-many elements and $2s$-many elements from the set of structures. Thus, given the distribution of sizes of subsets from a set containing $2s$-many elements, we can certainly expect to see at least one symmetric algorithm out of every $2s$-many produced by the stream (much more frequently than this, actually), and so this initial filter does not slow the stream appreciably – every polynomially-in-s-many ticks of the clock, we can expect a new symmetric algorithm to appear in our filtered stream.

As discussed earlier, the likelihood of finding highly rated algorithms in the stream itself is small, but what about the likelihood of finding algorithms having some minimal d-rating?

The process of "randomly picking" trivial Boolean symmetric algorithms in this context is combinatorially identical to the process of randomly picking size s subsets from a set having $2s$-many elements in it. An algorithm is perfect if the subset picked is identical with the set of E-structures, and in general, the ratings for any such algorithm are based on the degree to which the subset picked coincides with the set of E-structures.

Specifically, suppose we pick a subset whose intersection with the set of E-structures is of size k. Then the r_E-rating of this algorithm is k/s and the r_N-rating is $(s-k)/s$. Thus the d-rating of the algorithm is $(2k-s)/s$.. If we consider the random variable which takes as values the size of the intersection of our "guessed" subsets with the set of E-structures, then the probability density function for this variable is just the hypergeometric density function with parameter $1/2$, and as such, it has mean $s/2$ and variance

$$\frac{s}{4} - \frac{2(s-1)}{4(2s-1)} = \frac{s^2}{4(2s-1)}.$$

Since there exists a central limit theorem for this distribution, for large s it is effectively normal, and hence we know that the probability that a randomly selected set of size s is at least 1 standard deviation from the population mean is greater than 0.30.

What does this say about d-ratings? Since the mean of our distribution is $s/2$, and its standard deviation is

$$\frac{s}{2\sqrt{(2s-1)}},$$

subsets whose intersection with the set of E-records has size at least

$$\frac{s}{2} + \frac{s}{2\sqrt{(2s-1)}},$$

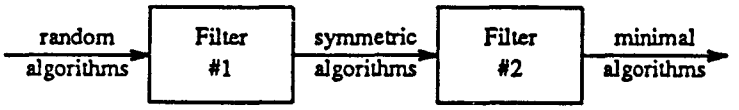

Fig. 6. Producing a minimal algorithm.

must have a d-rating at least

$$\frac{2\left(\frac{s}{2}+\frac{s}{2\sqrt{(2s-1)}}\right)-s}{s}=\frac{1}{\sqrt{2s-1}}.$$

In other words, we have proved that with probability greater than 0.15, a randomly produced algorithm has a d-rating

$$\frac{1}{\sqrt{2s-1}},$$

or greater. Let us call algorithms C such that

$$d(C)\geq\frac{1}{\sqrt{2s-1}}$$

minimal algorithms (see fig. 6).
Since our stream produces rated symmetric algorithms at a rate polynomial in s, and since

$$\frac{1}{\sqrt{2s-1}}>0,$$

we have thus shown that for some fixed real number $1/v>0$, we have a polynomial (in s) stream which randomly produces rated members of

$$\{C\colon d(C)>1/v\}.$$

We are now ready to employ the analyses of the previous two sections to produce an algorithm, C, based on stochastic discrimination. Given an arbitrary record, q, here is how to calculate $C(q)$:

Wait at the stream of minimal algorithms until you acquire

$$T(s)=P(2s,\,2\sqrt{2s-1}\,,\,s)=16s^4-8s^3 \tag{20}$$

many of them. From discussion above, the length of time spent at this task is clearly polynomial in s. Now simply calculate the value of M_n^q at this sequence of minimal algorithms so acquired (again, this requires no more than polynomial-in-s much time). If that value is within $1/(2\sqrt{2s-1}\,)$ of 1, we set $C(q)$ equal to 1; otherwise, we set $C(q)$ equal to 0.

How accurate is this algorithm? Given the number of minimal algorithms we employ, we know (by an appeal to lemma 3) that with probability greater than $1-(1/(2s))$, sample means of our random variables (E-record or N-record) lie

within $1/(2\sqrt{2s-1})$ of population means. But since the two population means are at least $1/\sqrt{2s-1}$ apart, the probability that an N-record random variable lies within $1/(2\sqrt{2s-1})$ of 1 (which is the population mean of E-record random variables) is less than $1/(2s)$. Thus the probability is less than 1 in $2s$ that C misclassifies any record; or, alternately, C has an expected d-rating greater than $1-(1/s)$.

Thus, if we view an "essentially perfect" solution to a pattern recognition problem as an algorithm expected to make less than one error, total, in classifying records, we have shown that for this general class of example, essentially perfect stochastic algorithms always exist. It is important to note that our procedure not only produces algorithms which operate at a rate polynomial in problem complexity – the procedure itself for producing the stochastic algorithm operates at a rate polynomial in problem complexity.

We had mentioned earlier that for this example, as s increases without bound, the probability of seeing better than trivially rated algorithms in the stream during the construction of our stochastic algorithm goes to 0. Specifically, for any real number ε greater than 0, let us denote by P_ε the probability that during the stochastic discrimination stage, the stream *ever* presents the stage with a separation algorithm with a d-rating greater than ε.

In the processing of the stream prior to stochastic discrimination for this example, two filters are employed (see fig. 7). The first filter only lets pass symmetric algorithms, and, as discussed earlier, one can expect to see at least one such algorithm out of every $2s$-many that appear in the stream. The second filter only lets past minimal algorithms, and since these are precisely those algorithms which (when viewed in terms of sets) are at least one standard deviation greater than the mean, one can expect to see at least one such algorithm out of every 7 that appear in the stream (since more than 30 percent of the area under a normal pdf lies further than one standard deviation from its mean). Combining these two facts, we see that out of every $14s$-many random algorithms in the original stream, at least one can be expected to make it through the second filter.

But for the moment, let us restrict our attention to what comes out of the first filter. If we look at the number of minimal algorithms needed to carry out stochastic discrimination (see (20)), and keep in mind the fact that the ratio of symmetric algorithms to minimal algorithms is about 7 to 1, we estimate that about

$$7T(s)$$

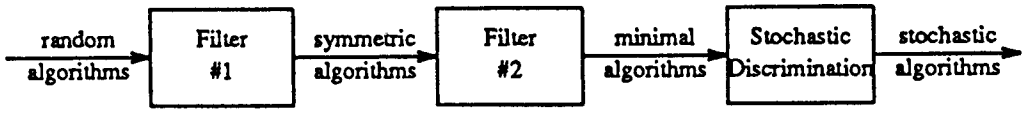

Fig. 7. Producing stochastic algorithms from minimal algorithms by using stochastic discrimination.

many algorithms output by filter no. 1 are required by our stochastic discrimination stage to derive a separation algorithm with an expected d-rating of at least $1-(1/s)$. Since there exist a total of

$$\binom{2s}{s} = \frac{(2s)!}{(s!)^2}$$

many symmetric algorithms, we thus have that the fraction of symmetric algorithms output by filter no. 1 during formation of the stochastic algorithm is

$$F(s) = \frac{7T(s)}{(2s)!(s!)^2}.$$

Clearly P_ε is some function of both $F(s)$ and the probability, $G(s)$, that a given symmetric algorithm has a d-rating greater than ε; and although we are not interested in developing just what this function is, it is clear that if both $F(s)$ and $G(s)$ are allowed to decrease to 0, P_ε must also approach 0.

So let us examine what happens as s approaches infinity. Since

$$\frac{(2s)!}{(s!)^2} = \prod_{i=0}^{s-1} \frac{2s-i}{s-i} > 2^s,$$

we clearly have that

$$F(s) = \frac{7T(s)}{(2s)!/(s!)^2} < \frac{7T(s)}{2^s}$$

approaches 0 as s approaches infinity. But what about $G(s)$? Looking back at our analysis above concerning the hypergeometric distribution, if we denote by $i(C)$ the size of the intersection of the set picked by symmetric algorithm C with the set of all E-structures, then since the mean of the distribution is $s/2$, and since its standard deviation is $s/(2\sqrt{(2s-1)})$, we know that for any positive integer n, those symmetric algorithms with d-ratings greater than

$$\frac{n}{\sqrt{2s-1}},$$

are exactly those symmetric algorithms C such that $i(C)$ is at least n standard deviations greater than the mean. But as n goes to infinity, the probability approaches 1 that given an arbitrary symmetric algorithm C, $i(C)$ is *less than* n standard deviations from the mean. In other words, given any number δ greater than 0, if n is so large that the probability is less than δ that given an arbitrary symmetric C, $i(C)$ is at least n standard deviations from the mean, and s is then chosen so large that

$$\frac{n}{\sqrt{2s-1}} < \varepsilon,$$

then for that value of s, the probability that a randomly chosen symmetric algorithm has a d-rating greater than ε is less than δ. Since δ was arbitrary, we have shown that as s approaches infinity, the probability that a randomly selected symmetric algorithm has a d-rating greater than ε approaches 0.

Combining all of the above, we have thus proved that for any ε greater than 0, as sample size increases without bound, the probability of ever seeing, through the stream fed to the stochastic discrimination stage, an algorithm with a d-rating greater than ε, approaches 0.

4. Remarks

(1) It is important to note that we have not just proved the existence of algorithms, but have actually given a procedure for producing them. In the case of our "example" the procedure is fully spelled out, but even in the abstract case considered earlier, given an externally supplied stream to carry out the production of "possible solutions" to a given problem, our procedure integrates the output of this stream, and, within polynomial time, generates an essentially perfect stochastic solution.

(2) A standard worry in solving pattern recognition problems is that the solution will be too "training set specific", and hence will not project to new cases. This is of special concern with computer generated solutions where the temptation is to use the large quantities of memory available to simply catalogue instances of structures for some (perhaps inadvertent) form of template matching. On the other hand, with complex problems it is often impossible to find "small" solutions. With stochastic discrimination, though, no matter how large and complex the solution may be, there is an apparent reduction in this danger of "training set specificity" since the algorithm development process never has access to any structures underlying the given pattern recognition problem, and hence, could never simply code templates. But even more to the point, our discussion following the proof of the duality lemma shows that if a stochastic algorithm is built from stable pieces, then *no matter how many such pieces are used, the stochastic algorithm will be stable.* Thus, when viewed from this perspective, stochastic discrimination provides a possible solution to the seemingly paradoxical problem of finding complex, *but not "training set specific"*, solutions to pattern recognition problems.

(3) The development carried out here was in terms of random variables which were, in some sense, *normalized.* At times it may be desirable to consider the random variable X^q to simply take on the value $C(q)$ at a given observation C. In this case, the development carried out above is still valid, however, the expectation of E-structure random variables is no longer equal to 1. Referring back to the proof of lemma 2, but now using this new version of the random variables, let $i \leq s$ and $j \leq t$ satisfy $i/s - j/t > 1/\nu$, and let $\mathscr{D}$ denote the

subspace of all algorithms in $\mathscr{S}$ with an r_E-rating of i/s and an r_N-rating of j/t. Then we immediately see that for any E-structure q, the expectation of X^q restricted to $\mathscr{D}$ is

$$\frac{\binom{t}{j}\binom{s-1}{i-1}}{\binom{t}{j}\binom{s}{i}} = \frac{\left(\frac{t!}{j!(t-j)!}\right)\left(\frac{(s-1)!}{(i-1)!(s-i)!}\right)\left(\frac{s}{i}\right)\left(\frac{i}{s}\right)}{\left(\frac{t!}{j!(t-j)!}\right)\left(\frac{s!}{i!(s-i)!}\right)} = \frac{\binom{t}{j}\binom{s}{i}\left(\frac{i}{s}\right)}{\binom{t}{j}\binom{s}{i}} = \frac{i}{s}.$$

Thus for any E-structure q, the expectation of X^q is

$$\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}\left(\frac{i}{s}\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}.$$

Similarly, for any N-structure q, the expectation of X^q is

$$\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}\left(\frac{j}{t}\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}.$$

Thus the difference of these two expectations is

$$\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}\left(\left(\frac{i}{s}\right)-\left(\frac{j}{t}\right)\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}.$$

However, we know that

$$\frac{i}{s}-\frac{j}{t}>\frac{1}{\nu},$$

and so the difference of the two expectations is greater than

$$\frac{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}\left(\frac{1}{\nu}\right)}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}} = \frac{\left(\frac{1}{\nu}\right)\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}}{\sum_{i=\langle s/\nu\rangle}^{s}\sum_{j=0}^{[t(i/s-1/\nu)]}\binom{s}{i}\binom{t}{j}} = \frac{1}{\nu}.$$

Thus we are left with the same situation as before, where the random variables for a large number of independent trials congregate about two poles a distance more than $1/\nu$ apart.

(4) Suppose one is confronted with a pattern recognition problem, and has access to a stream of proposed solutions, but does not have available the r_E, r_N, and d-ratings used above. Then the general process described in the introduction is still possible even if one must make an empirical determination of what constitutes an average (random) algorithm in order to filter the stream for the purpose of enriching it. For given any sequence of nontrivial algorithms, the (nonnormalized) random variables discussed just above can be used, and their means will still congregate about distinct poles. The only requirement is that the rating scheme which is associated with the algorithms in the stream in unbiased across structures of a given category (E or N), and that any solution it rates as nontrivial, would also have a d-rating which was nontrivial.

(5) As mentioned in the introduction, the process of stochastic discrimination is highly amenable to implementation on arbitrary multiprocessor machines. For given the fact that the process takes as input algorithms *generated by any means* (related only by their addressing of a common problem), these algorithms might well be generated most rapidly by independent parallel computations.

Acknowledgements

The author wishes to thank L.D. Brown from Cornell for many helpful suggestions in the presentation of this material. In particular, Professor Brown suggested a shorter proof of lemma 3 than that presented in our original manuscript – the proof included in this paper is a result of his remarks.

References

[1] E.G. Kleinberg and E.M. Kleinberg, An application of stochastic discrimination of digit recognition, in preparation.

[2] E.M. Kleinberg, Stochastic discrimination and the traveling salesman problem, in preparation.

[3] E.M. Kleinberg, A general theory of artificial learning, K.S.I. Tech. Rep. no. 04.0288, K. Square (December 1985).