On the Algorithmic Implementation of Stochastic Discrimination

Eugene M. Kleinberg

Abstract—*Stochastic discrimination* is a general methodology for constructing classifiers appropriate for pattern recognition. It is based on combining arbitrary numbers of very weak components, which are usually generated by some pseudorandom process, and it has the property that the very complex and accurate classifiers produced in this way retain the ability, characteristic of their weak component pieces, to generalize to new data. In fact, it is often observed, in practice, that classifier performance on test sets continues to rise as more weak components are added, even after performance on training sets seems to have reached a maximum. This is predicted by the underlying theory, for even though the formal error rate on the training set may have reached a minimum, more sophisticated measures intrinsic to this method indicate that classifier performance on both training and test sets continues to improve as complexity increases. In this paper, we begin with a review of the method of stochastic discrimination as applied to pattern recognition. Through a progression of examples keyed to various theoretical issues, we discuss considerations involved with its algorithmic implementation. We then take such an algorithmic implementation and compare its performance, on a large set of standardized pattern recognition problems from the University of California Irvine, and Statlog collections, to many other techniques reported on in the literature, including boosting and bagging. In doing these studies, we compare our results to those reported in the literature by the various authors for the other methods, using the same data and study paradigms used by them. Included in this paper is an outline of the underlying mathematical theory of *stochastic discrimination* and a remark concerning boosting, which provides a theoretical justification for properties of that method observed in practice, including its ability to generalize.

Index Terms—Pattern recognition, classification algorithms, stochastic discrimination, SD.

1 Introduction

1.1 General Outline of Paper

Stochastic discrimination (*SD*) is a general methodology for constructing classifiers appropriate for pattern recognition. It is based on combining arbitrary numbers of very weak components, which are usually generated by some pseudorandom process and it has the property that the very complex and accurate classifiers produced in this way retain the ability, characteristic of their weak component pieces, to generalize to new data. This phenomenon is well-understood from a theoretical point of view and has been consistently observed in applications where theoretical norms are not usually met. In fact, it is often observed that classifier performance on test sets continues to rise as more weak components are added, even after performance on training sets seems to have reached a maximum. This phenomenon is also understood mathematically; it turns out that even though the formal error rate on the training set may have reached a minimum, more sophisticated measures intrinsic to this method indicate that classifier performance on both training and test sets continues to improve as complexity increases.

Stochastic discrimination might appear to fall in the general category of pattern recognition methods based on the notion of combining classifiers. But there is a fundamental difference which epitomizes the essence of stochastic discrimination. Indeed, most combination methods deal with component classifiers which, even if they may be weak for the problem at hand, are still somewhat "expert" in that they will tend to agree with one another about the classification of at least some of the "easy" points in the feature space. However, in the case of stochastic discrimination, not only does this tend not to happen—it is essential to the success of the method that it be avoided to the greatest extent possible. We are not talking here about simply trying to maintain some degree of orthogonality among component classifiers. In the case of SD, all points of a given class in the feature space must be "viewed equivalently" by the full set of weak component classifiers. This notion, which in the theory of stochastic discrimination is known as "uniformity," captures the essence of stochastic discrimination, making it, in effect, a method for combining "classifiers" which, modulo specific points in the feature space, have no commonality whatsoever. In a sense, the weak component classifiers are not really classifiers at all—they are simply subsets of the feature space selected from a randomly generated stream of such subsets *solely* by virtue of their having an error rate for the problem at hand slightly better than the average for the stream.

Combination methods do exist which bear relationships to SD. Most of these, such as Ho's work dealing with Decision Forests (see [6], [7]), are derivatives of the original theory of stochastic discrimination (see [8], [9]), where resistance to overtraining is maintained by combining uniform streams of (slightly) enriched, projectable weak classifiers. Here, for example, one uses multiple trees built in randomly chosen subspaces of the feature space. For another example of derivative work, see [1].

• *The author is with the Department of Mathematics, the State University of New York, Buffalo, NY 14214. E-mail: kleinbrg@math.buffalo.edu.*

Manuscript received 13 July 1999; revised 14 Mar. 2000; accepted 14 Mar. 2000.
Recommended for acceptance by T.K. Ho.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 110227.

However, the combination method known as boosting (see [4]), based as it is on the notion of iteratively adjusting the distribution of training points so as to deemphasize the "easy" points, also bears a certain relationship to SD. In fact, by viewing boosting in an appropriate way, one can use the mathematical theory of SD ([8], [9]) to provide a theoretical base explaining empirically observed characteristics of boosting, including its performance on nontraining data. We do not appeal to an argument involving VC dimension (see [14]), as is done in [5]. We comment further on this later in this paper, and provide a complete, self-contained presentation in [11].

While SD may be easily structured with a preset limit on the VC dimension of the "weak models" considered for inclusion in the final (combined) classifier, another unique aspect of the underlying theory is that there is no restriction on the number of such weak models which may be included. In fact, the VC dimension of the set of classifiers built by SD in any particular context is usually infinite, and as such, our mathematical proof that classifiers built by SD generalize to nontraining data is not based on limiting VC dimension.

Stochastic discrimination has a complete theoretical base ([8], [9]) which accounts for the performance of algorithmic implementations. While historically the mathematical theory came first, in this paper we begin with a nontheoretical, algorithm-oriented, review of the method of stochastic discrimination. Using a sequence of simple, synthetic examples, we motivate the required components by observing the effect of different algorithms on carefully constructed problems. In short order, we build up to pseudocode containing all key pieces required for a full implementation.

There exist many methods for building classifiers. Some of these are quite general and do well on large classes of problems, whereas others are quite specific to certain types of applications. Some methods have strong theoretical bases, while others are quite heuristic, and their apparent success at dealing with certain problems is not well-understood. Given all of this, it has become popular in recent years to compare different methods by evaluating the performance of algorithmic implementations on standardized problems in pattern recognition. Two such sets of problems have been used extensively, namely the Statlog datasets and the University of California, Irvine datasets.

We conclude this paper with a discussion of the application of an implementation of stochastic discrimination to these datasets and compare its performance to that of other methods as reported in the recent literature under the same test conditions. In the case of the Statlog problems, we compare results to 23 different methods reported on by the Statlog project and in the case of the Irvine problems, we compare results to three underlying methods along with their boosted and bagged versions (for a total of nine methods) as reported on by Freund and Schapire in [4]. As an initial preview, we might point out here that of the 24 experiments reported on, stochastic discrimination placed first on 19 of them, second on two others, fourth on another, and fifth on the remaining two. Again, we compare our performance to results as reported in the literature by the various authors for the other methods and carried out our runs using the same data and study paradigms as that used by them. Furthermore, we basically ran our algorithm as it came "out of the box" with no special "tuning," or adjustment of parameters, to the particular problems at hand.

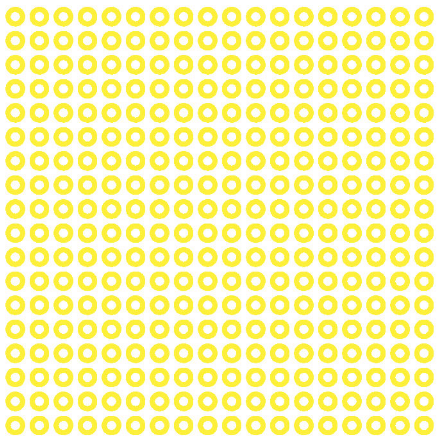
Fig. 1. Feature space.

2 Stochastic Discrimination

2.1 The Central Limit Theorem

Classification models built by the method of stochastic discrimination are created by sampling *from the space of all subsets of the underlying feature space* of the problem at hand. For the purpose of motivating the method, let us assume that our underlying feature space is the subset of Euclidean 2-space consisting of all points (n,m) such that n and m are positive integers no larger than 18. In other words, our feature space, henceforth denoted F, consists of an 18 by 18 grid of lattice points (see Fig. 1). (There is nothing special about our choice of the size of this feature space. We could just as well have used 20 by 20, 15 by 15, etc.) Given a subset S of F, recall that the characteristic function of S, denoted C_S, is the function from F into {0,1} defined by "$C_S(q) = 1$ iff q is a member of S."

For a start, assume that we carry out the following very simple process:

```
Algorithm: 1cl.pts
Procedure: INITIALIZE()
    For each $q$ in $F$
        Set $number_q := 0.0$
    end for
    Set $counter := 0$
end Procedure:INITIALIZE()
While 1
    Procedure:GENERATE.RANDOM.SUBSET()
        Generate a random subset $S$ of $F$, with the
            probability of any point of $F$ being in $S$ of 0.5
    end Procedure:GENERATE.RANDOM.SUBSET()
    Procedure:UPDATE.VARIABLES()
        For each $q$ in $F$
            Update $number_q := number_q + C_S(q)$
        end for
        Update $counter := counter + 1$
        For each $q$ in $F$
            Update $probty_q := number_q/counter$
        end for
    end Procedure:UPDATE.VARIABLES()
end while
```

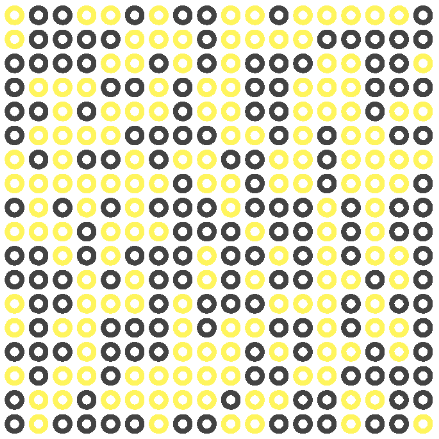
Fig. 2. Random subset of feature space.

In effect, this process proceeds to place random subsets (see Fig. 2) on top of the points in F again and again, and at any given stage, the variable $number_q$ simply counts the total number of such sets which have landed on top of q to that point in time, something we might call the total coverage of q to that point in time; and $probty_q$ measures the average coverage of q to that point in time, that is, the total number of covers which landed on top of q divided by the total number of covers produced. Note that in each iteration of this process, each point q of the space has its associated variables $number_q$ and $probty_q$ updated. Also, note that, in our generation of subsets of F, we are sampling *with replacement*.

For a given value of $counter$, we are primarily interested in the distribution of the values $probty_q$ (with q varying over F). But there are really two random variables which can be considered here. The first, as just mentioned, is the random variable $probty_q$, a function of q ranging over the discrete sample space F (assuming a fixed value for $counter$). However, we could, alternatively, choose a point q_0 in F, and consider the random variable $C_S(q_0)$, a function of S ranging over the discrete sample space, $\mathbf{F}_{.5}$, consisting of those subsets of F whose measure is equal to half the measure of F. Given the way our algorithm above carries out sampling from this space, it is clear that for any fixed values of $counter$ and q_0, the first of these random variables, $probty_q$ (defined on the sample space F), is identically distributed to the random variable

$$\frac{\sum_{k=1}^{counter} C_{S_k}(q_0)}{counter}$$

(defined on the product sample space $(\mathbf{F}_{.5})^{counter}$). On the other hand, given that our algorithm is carrying out sampling with replacement from $\mathbf{F}_{.5}$,

$$\sum_{k=1}^{counter} C_{S_k}(q_0)$$

is a sum of independent identically distributed random variables, and as such, by the Central Limit Theorem, as $counter$ approaches infinity, the distribution of

$$\frac{\sum_{k=1}^{counter} C_{S_k}(q_0)}{counter}$$

approaches a normal distribution with mean equal to that of $C_S(q_0)$, namely 0.5, and variance approaching 0.

This brief analysis is illustrated graphically in Fig. 3, where we present pictures of probability mass functions for the random variable $probty_q$ over a sequence of increasing values of $counter$ produced by an actual run of our algorithm 1cl.pts described above. As predicted by the Central Limit Theorem, the shape of the mass functions approaches the standard bell-shape; furthermore, note the change in shape of the mass functions, in particular, the evident decrease in variance, as $counter$ increases.

(For these pictures, as well as for other pictures to appear later in this paper, the "mass functions" are really smoothed approximations which we produced by partitioning the set of possible values for random variables into intervals and then calculating probabilities by simply counting numbers of points q (whose associated values $probty_q$ were) "bracketed" by these intervals. We then carried out average-smoothing (each y_i replaced with $\frac{y_{i-1}+y_i+y_{i+1}}{3}$) on the resulting functions. In the graphs themselves, the x-axis represents possible values for the random variable $probty_q$ and the y-axis represents probabilities. Furthermore, we place an x-tic, and (space permitting) associated value, at the mean of $probty_q$ and place x-tics one standard deviation to either side of the mean. We also place reference values near the extremes of each axis to provide a sense of scale.)

So far, no discrimination has taken place, but this is about to change.

2.2 Two-Class Problem

Assume now that the points in our feature space are actually of two different types. Let's refer to them as the green points and the red points, and let us assume that the green points reside in the three outer rings of the feature space, and that the reds reside in the remaining inner 12 by 12 square region (see Fig. 4). Then, of course, if we carry out the above process as before, as $counter$ approaches infinity, the mass function of the distribution of the $probty_q$ for red points, q, approaches bell-shape, with mean 0.5 and standard deviation approaching 0, and the mass function of the distribution of the $probty_q$ for green points, q, also

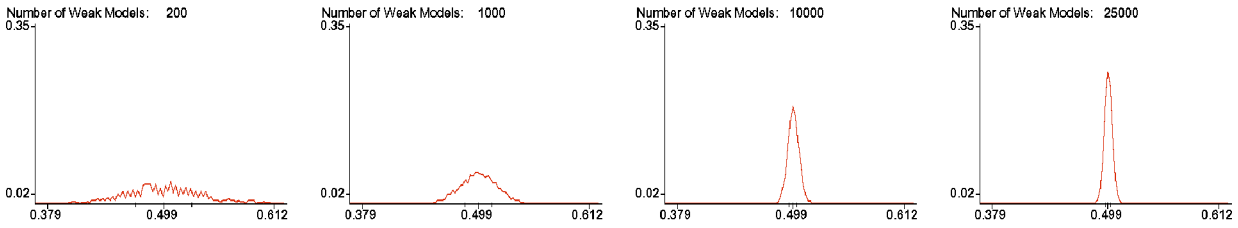

Fig. 3. One-class problem, weak model stream: random subsets.

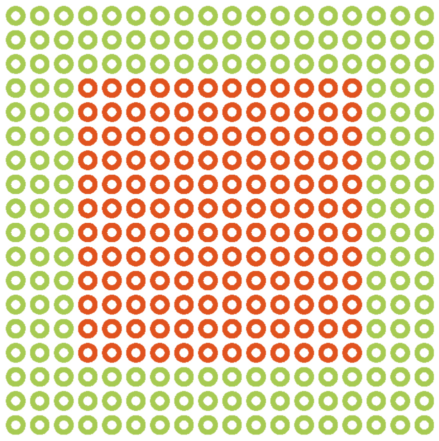
Fig. 4. Feature space, two-class problem.

approaches bell-shape with mean 0.5 and standard deviation approaching 0.

However, suppose that we now carry out the process of repeatedly covering the feature space with random subsets, but this time, rather than allowing the covering sets to treat all points equally, having the same probability of capturing any one point as any other, we require that a covering set have *slightly higher* probability of capturing red points than of capturing green points. For example, we could generate random subsets in such a way that any given red point has a 0.51 chance of being captured and any given green point has a 0.49 chance of being captured. Instead, let us use a more general algorithm. We will simply generate subsets of F at random as before, but now only "keep" those which capture a higher fraction of red points than of green points: (note that some procedures called in this code were defined previously).

```
Algorithm: 2cl.pts
INITIALIZE()
While 1
        Set $S.not.enriched := 1$
        While $S.not.enriched$
              GENERATE.RANDOM.SUBSET()
              Procedure:CALCULATE.COVERAGES($F$)
                    Set $Frac.reds$ equal to number of reds in $F$
                        covered by $S$ divided by the total number
                        of reds in $F$
                    Set $Frac.greens$ equal to number of greens in
                        $F$ covered by $S$ divided by the total
                        number of greens in $F$
              end Procedure:CALCULATE.COVERAGES($F$)
              Procedure:CHECK.ENRICHMENT()
                    If $Frac.reds > Frac.greens$
                        Set $S.not.enriched := 0$
              end Procedure:CHECK.ENRICHMENT()
        end while
        UPDATE.VARIABLES()
end while
```

There are a few things to note. First of all, given the way in which we did not use all subsets which are generated in updating the values of the $probty_q$, the distribution of the $probty_q$ for red points q is now different from that for green points q. Furthermore, while the means and standard deviations for the distributions of red and green $probty_q$ are now more difficult to calculate than before, we can easily see that, 1) the expectation of $probty_q$ for red q is greater than the expectation of $probty_q$ for green q; and 2) the central limit theorem applies to the distributions of the $probty_q$ for red q and for green q, and, in particular, as $counter$ approaches infinity, the mass functions approach bell-shape with means $mean.red > mean.green$ and standard deviations approaching 0.

In Fig. 5, we illustrate this graphically for an actual run of the algorithm described above. In this figure, and in all future figures of this general sort displaying sequences of probability mass functions, the probability mass functions of $probty_q$ for red q appear in red, and are usually located shifted to the right of the corresponding probability mass functions of $probty_q$ for green q, which appear in green. Furthermore, we draw a vertical line at the value $threshold$, a number equal to the average of the two means $mean.red$ and $mean.green$. Finally, we use the word "level" to refer to the number of enriched, random subsets used. This is the same notion we had called "number of weak models" in Fig. 3.

The idea underlying stochastic discrimination is now apparent: in order to decide if a given point q is a red or a green, simply calculate $probty_q$. If $probty_q$ is greater than $threshold$, the point is a red; otherwise it is a green.

How accurate is the classifier we just described? Clearly the points erroneously classified are those reds whose $probty_q$ lies to the left of $threshold$ and those greens whose $probty_q$ lies to the right of $threshold$, in other words, those points represented by the region in any of the graphs in Fig. 5 where the two mass functions overlap. However, as $counter$ increases to infinity, the variance of the mass functions approaches zero, and since the expectations of the mass functions remain the same distinct values, the area of the overlap region, and hence, the probability of error for the classifier, approaches zero. This is apparent by looking at the progression of graphs.

2.3 Training and Test Sets

Of course, when building classifiers in pattern recognition, we are just given a "representative" training set to work with, as opposed to the entire space of examples. Our goal

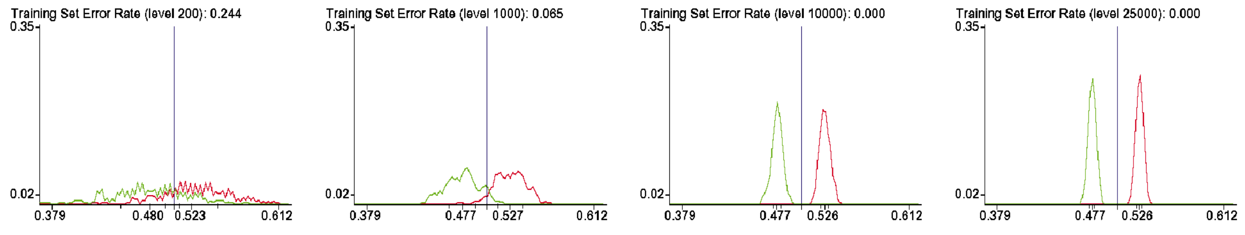

Fig. 5. Two-class problem, weak model stream: random subsets.

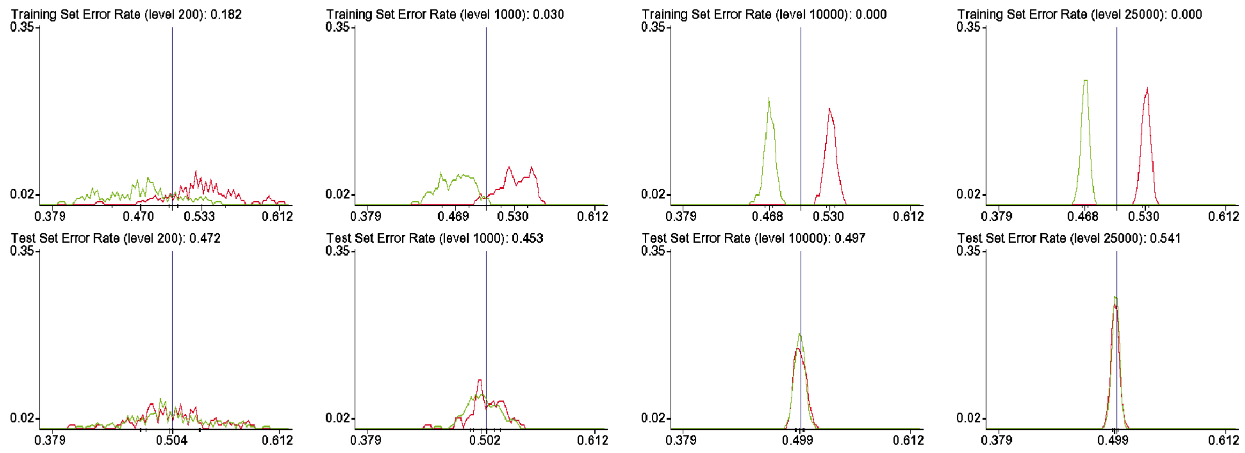

Fig. 6. Two-class problem, training and test sets, w.m.stream: random subsets.

is to carry out the above development of a classifier now using only a training set of examples, with the hope that the classifier which is created has good accuracy not only on the training examples, but on all points in the feature space.

One might note that in most "real-world" pattern recognition problems the feature space contains points which are of neither of the two classes. From the perspective of motivating our approach, however, this extra complication adds no useful insight, and so, for the sake of simplicity, we ignore it.

Let us then partition the feature space F, at pseudorandom into training and test sets, TR and TE, respectively, of approximately equal size, and run the above algorithm just as before with the one exception that the determination of whether or not a set is enriched with respect to reds is made solely based on computations involving points in the training set.

Algorithm: trte.pts

```
INITIALIZE()
While 1
    Set $S.not.enriched := 1$
    While $S.not.enriched$
        GENERATE.RANDOM.SUBSET()
        CALCULATE.COVERAGES($TR$)
        CHECK.ENRICHMENT()
    end while
    UPDATE.VARIABLES()
end while
```

In Fig. 6, we illustrate the results of our run as before, but this time, directly below each pair of probability mass functions for the greens and reds in the training set, we show the corresponding pair of mass functions for the greens and reds in the test set. (As a result, there are a total of eight images displayed in this figure rather than the four displayed in previous such figures.)

As before, the pair of mass functions for the training set behave as expected with distinct expectations and variances approaching zero; but the pair for the test set, while certainly having variances approaching zero, seem to have nondistinct expectations. What has happened here?

The answer is quite simple. Since the enrichment which took place was just with respect to the training set, the stream of subsets which were retained was not enriched with respect to the test set, and so, as far as the points in the test set were concerned, the stream was not enriched at all, resulting in no distinction between reds and greens in the test set. The net effect is exactly the same as the process we carried out initially as illustrated in Fig. 3, where we had a single mass function for all points in the feature space.

2.4 Thick Weak Models

How then, can one go about building a classifier based solely on training data which shows at least some discrimination power when evaluated on test data? The problem we had in the example above was based on the fact that any particular subset being enriched with respect to reds on training data does not imply anything about whether or not that same set is enriched with respect to reds on test data. We need a method to retain sets enriched with respect to reds on test data and we must do this without having any access to the test data.

Of course, this is, in general contexts, impossible to do. However, there is implicit in any pattern recognition problem an underlying assumption about the data provided and that is that there exists some relationship between points of a given class in the training set and in the test set. In our example given here, that relationship is that points of either class, in both training and test sets, tend to be clustered in similar regions of the feature space, that is, neighborhoods (of diameter greater than zero) of red points in the training set tend to have mostly red points (from both training and test sets), and neighborhoods (of diameter greater than zero) of green points in the training set tend to have mostly green points (from both training and test sets).

The stream of subsets of the feature space generated in our algorithm above consisted of subsets which were unions of neighborhoods of diameter zero about points in the feature space, and as such, the fact that any such set tended to contain a higher fraction of reds than of greens on the training set said nothing about the relative fractions of reds and greens from the test set captured by the generated subset. However, if we generated subsets which are unions of *positive*-diameter neighborhoods about points in the feature space, then the fact that such a subset is enriched with respect to reds as measured on the training set should translate into its being enriched with respect to reds as measured on the test set.

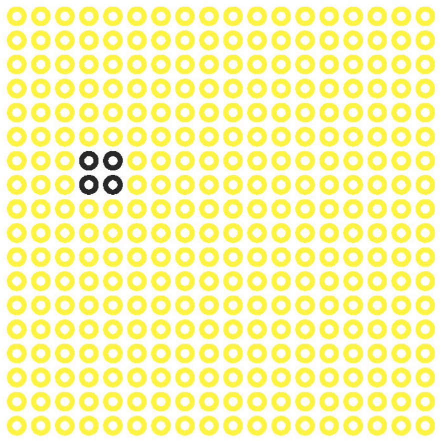
Fig. 7. Thick weak model.

So let's do exactly that. For simplicity, each of our randomly generated subsets of the feature space will consist of a single 2x2 square region (see Fig. 7). For the problem at hand, any such region, even though small, is still "thick" enough that if it tends to be enriched with respect to red points from the training set, it will probably also be enriched with respect to red points from the test set. ("Thickness" in the context of more complex classification problems will be discussed later in this paper. In particular, see the discussion at the start of Section 5.)

Here is the algorithm:

```
Algorithm: trte.2x2
INITIALIZE()
While 1
    Set $S.not.enriched := 1$
    While $S.not.enriched$
        Procedure:GENERATE.RANDOM.SQUARE()
            Generate a random subset $S$ of $F$, such that $S$
            consists of a 2x2 region in $F$; that is, choose a
            random point (i,j) in $F$ such that both i and j
            are less than 18, and let S be the set
            {(i,j),(i+1,j),(i,j+1),(i+1,j+1)}
        end Procedure:GENERATE.RANDOM.
            SQUARE()
        CALCULATE.COVERAGES($TR$)
        CHECK.ENRICHMENT()
    end while
    UPDATE.VARIABLES()
end while
```

In Fig. 8, we illustrate the results of a run of this algorithm. Note, as predicted, the expectations for the test set mass functions are distinct, and so, the classifier produced does have some predictive power when it comes to distinguishing among points in the test set. But there seems to be a new wrinkle, namely the mass functions for both the training and test sets no longer seem to be approaching bell-shaped functions. What has happened?

2.5 Uniformity

The answer lies at the heart of the method of SD. Let's simplify the example and try to isolate the problem. Namely, let's not worry about training and test sets for the moment, but let's retain our "thick"-component subspaces. First, a run with no enrichment:

```
Algorithm: Nenrich.2x2
INITIALIZE()
While 1
    Set $S.not.enriched := 1$
    While $S.not.enriched$
        GENERATE.RANDOM.SQUARE()
        Set $S.not.enriched := 0$
    end while
    UPDATE.VARIABLES()
end while
```

Notice the fact that in Fig. 9 the mass functions now appear bell-shaped. However, as we would expect from a nonenriched stream, the means of the red and green mass functions are identical.

Now consider the same algorithm, but this time with enrichment. In order to make our analysis of the output as simple as possible, we will, for this example, use a very simple scheme of enrichment, namely the requirement that the randomly generated subset contains only red points:

```
Algorithm: enrich.2x2
INITIALIZE()
While 1
    Set $S.not.enriched := 1$
```

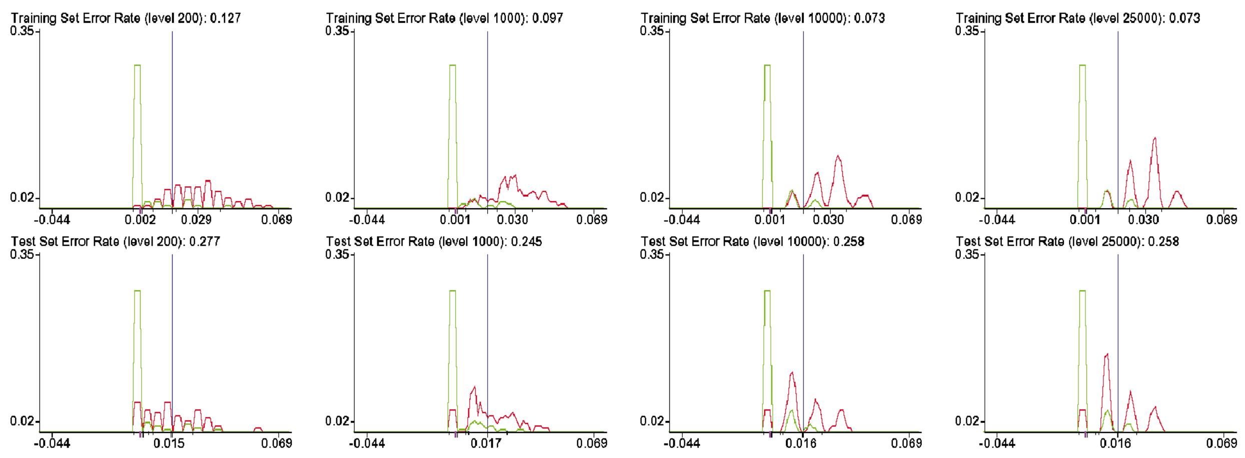

Fig. 8. Two-class problem, training and test sets, w.m.stream: 2 by 2 squares.

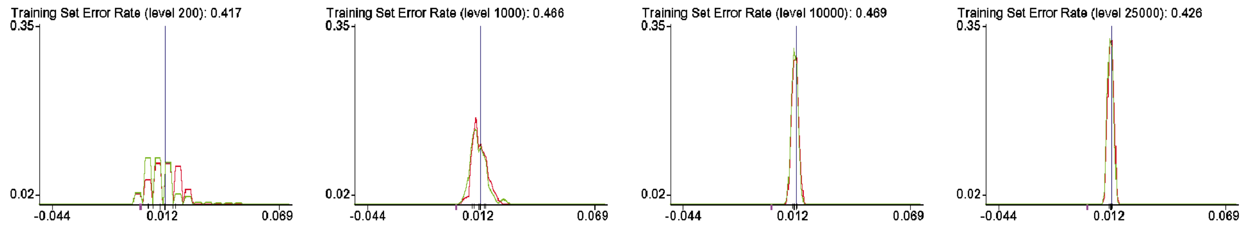

Fig. 9. Two-class problem, no enrichment, w.m.stream: 2 by 2 squares.

```
    While $S.not.enriched$
        GENERATE.RANDOM.SQUARE()
        CALCULATE.COVERAGES($F$)
        Procedure:CHECK.STRONG.ENRICHMENT()
            If $Frac.greens = 0$
                Set $S.not.enriched := 0$
            Else
                Set $S.not.enriched := 1$
        end Procedure:CHECK.STRONG.
            ENRICHMENT()
    end while
    UPDATE.VARIABLES()
end while
```

Now we have separation of means, but that problem of non-bell-shaped mass functions has, not surprisingly, once again appeared. A moment's reflection, however, tells us why. For given that we are randomly generating 2x2 square regions in the feature space, but only keeping those which only lie entirely within the central region occupied by the red points, the probability that a red point lying at a corner of the red region gets hit by such a set is $1/4$ the probability that a red point lying in the interior or the red region gets hit, and it is $1/2$ the probability that a red point lying on the edge of the red region (but not at a corner) gets hit. This is simply because each interior point has four different possible 2x2 squares which may hit it, each edge point has two different possible 2x2 squares, but each corner point has only one possible 2x2 square which may hit it. Keep in mind, the squares in the stream must lie entirely within the red region.

So the shapes of the mass functions resulting from our run, shown in Fig. 10 are exactly what we would expect. For reds, the bulk of the function concentrates on the interior red points, and becomes a bell-shaped "subfunction" with a given expectation, say m. The edge (noncorner) points generate a "subfunction" which also approaches bell-shape and has expectation $m/2$. Finally, the corner points generate a bell-shaped "subfunction" having expectation $m/4$. As the number of weak models approaches infinity, the variances of each of these "subfunctions" approaches zero.

So far as the green points are concerned, since no weak model in our stream can hit any of them, the distribution function of green points is a simple step function with expectation 0.

The astute reader may well be concerned with our result in Fig. 9, where we did not carry out any enrichment, for given that our feature space is an 18 by 18 square grid, points on the extreme edge of this grid are less likely to be hit by randomly generated 2x2 squares just as corner and edge red points were less likely to be hit by enriched 2x2 squares above. This is indeed the case. However, in order to take this wrinkle out of the picture, we effectively removed the edges and corners of our feature space for that example by identifying opposite edges, effectively turning the feature space into a torus. We simply felt that the additional complication introduced by points located at the extreme edges of the space, was a complication we didn't need at this point.

At any rate, this issue of subfunctions resulting from unequal treatment of differently positioned red points, causes clear problems for our classifier since as can be seen from the example above, the corner reds, no matter how long we seem to run the algorithm, remain on the wrong side of the threshold.

This is a manifestation of the problem of uniformity. A stream of weak models is said to constitute a uniform cover of the space if any two points of a given class (e.g., any two reds or any two greens) are equally likely to be captured by a weak model in the stream. One should refer to [9] (or Section 3 following) for a precise definition, but based on our informal discussion here, the following facts emerge: 1) If we simply generate a stream of random subsets of the feature space, that stream provides a uniform cover (but enriching it with respect to the training set fails to enrich it with respect to the test set, and so, the stochastic model built from such a stream does not generalize, to any degree whatsoever, to new data); 2) if we generate a nonenriched stream of "thick" subsets of the feature space, that stream

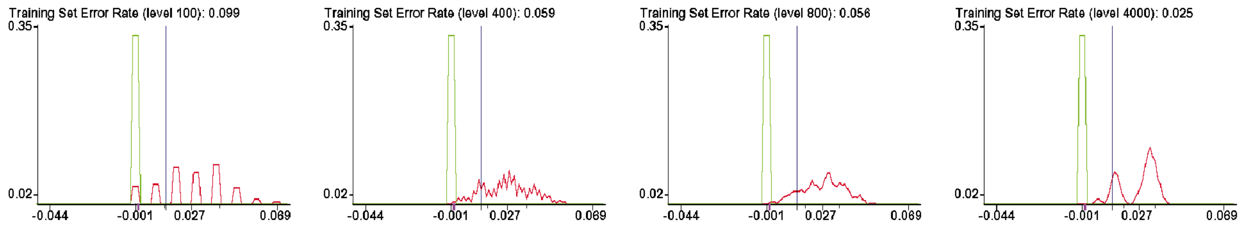

Fig. 10. Two-class problem, enrichment, w.m.stream: 2 by 2 squares.

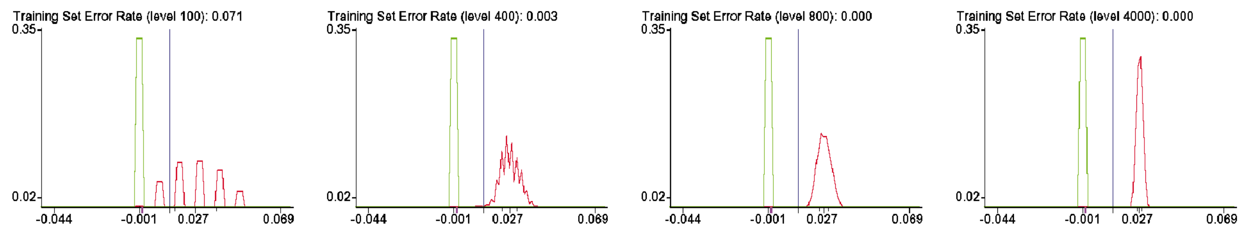

Fig. 11. Two-class problem, enrichment, uniform w.m.stream: 2 by 2 squares.

provides a uniform cover of the feature space (if the space lies on a torus) (but a nonenriched stream fails to distinguish among the different classes in the feature space and as such is useless for doing pattern recognition); 3) if we generate an enriched stream of "thick" subsets of the feature space that stream does distinguish among the difference classes, and it does generalize to new data, but it fails to provide a uniform cover of the feature space, and as such, the accuracy of the resulting classifier may suffer.

2.6 Uniformity Forcing

Thus, with respect to the three central factors—enrichment, uniformity, and generalization-power—any two can be achieved naturally, with an appropriate stream of weak models, if we don't worry about achieving the third. Since we are doing pattern recognition here, let us resolve, initially, that achieving generalization-power is essential.

The resulting conflict between enrichment and uniformity can now be ameliorated algorithmically by adding another requirement for retaining weak models beyond that dictated by enrichment. Specifically, we will require that the algorithm keep track of the coverage of points in the feature space by weak models previously retained, and if a new weak model is being considered for retention we simply look at the points it captures, and if their coverage was below the average coverage for points of their class, then the new weak model will be retained. In effect, we select from the full stream of possible weak models those which capture, and hence, effectively, elevate, points in the feature space whose current coverage is light compared to others in their class. From a theoretical perspective, this algorithmic device allows us to sample from a *uniform* subspace of the space of enriched weak models, and as such we are still, effectively, taking sums of independent, identically distributed random variables. See Section 3.

Here is the algorithm:

```
Algorithm: Ntrte.unif
Procedure:INITIALIZE.ALL()
   INITIALIZE()
   Set $av_{red} := 0$
end Procedure:INITIALIZE.ALL()
While 1
    Set $S.not.yet.acceptable := 1$
    While $S.not.yet.acceptable$
       GENERATE.RANDOM.SQUARE()
       CALCULATE.COVERAGES($F$)
       CHECK.STRONG.ENRICHMENT()
       If $S.is.enriched = 1$
          Procedure:CHECK.UNIFORMITY.
             FORCING($F$)
             If $counter = 0$
                Set $S.not.yet.acceptable := 0$
             Else
                Set $S.av_{red}$ equal to the average of
                      $number_q$
                   for red points $q$ in $F$ which lie in $S$.
                If $S.av_{red} < av_{red}$
                      Set $S.not.yet.acceptable := 0$
          end Procedure:CHECK.UNIFORMITY
             .FORCING($F$)
    end while
    Procedure:UPDATE.ALL.VARIABLES($F$)
       UPDATE.VARIABLES()
       Update $av_{red}$ to be the average of $number_q$ for
          red $q$ in $F$
    end Procedure:UPDATE.ALL.VARIABLES($F$)
end while
```

A run of this algorithm produces the mass functions shown in Fig. 11. Notice that there are no longer separate "subfunctions" present and that the reds and the greens each have mass functions with distinct expectations, each of which appears to approach bell-shape.

2.7 The Full Algorithm

We're now ready to go back to the complete problem with training and test sets:

```
Algorithm: trte.unif
INITIALIZE.ALL()
While 1
    Set $S.not.yet.acceptable := 1$
    While $S.not.yet.acceptable$
       GENERATE.RANDOM.SQUARE()
       CALCULATE.COVERAGES($TR$)
       CHECK.STRONG.ENRICHMENT()
       If $S.is.enriched = 1$
          CHECK.UNIFORMITY.FORCING($TR$)
    end while
    UPDATE.ALL.VARIABLES($TR$)
end while
```

At last, we have exactly what we need. As we see in Fig. 12, both training and test sets show approximately bell-shaped mass functions for reds and greens, and as *counter* approaches infinity, the convergence of the variances to zero leads to perfect separation of the classes. (In light of the simplifications included in our previous four examples, we used a regular partition scheme for the run illustrated in Fig. 12, namely, a given point (i,j) of the feature space was placed into the training set iff $(\mathrm{i} + \mathrm{j}) \bmod (2) = 0$.)

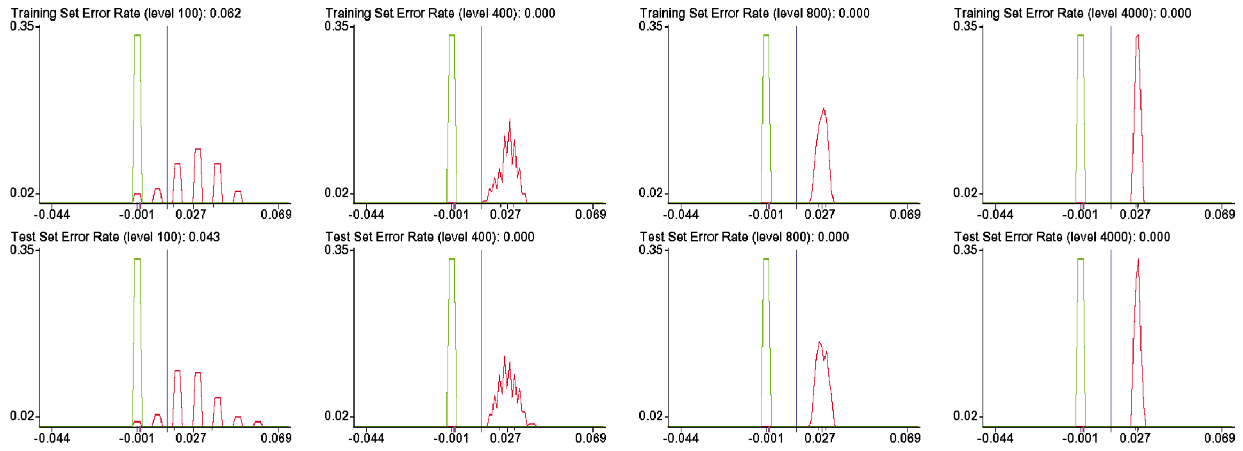

Fig. 12. Two-class problem, training and test sets, uniform w.m.stream: 2 by 2 squares.

2.8 Resistance to Overtraining

We are now in a position to see why SD is particularly resistant to overtraining and why it is possible for test set performance, in fact, classifier generality, to continue to improve after training set performance appears to reach a maximum. For the error rate on either the training or test set of a model produced by SD is based on the overlap of the mass functions associated with the training or test set for the different classes. The shapes of the mass functions, in turn, are based on the enrichment and uniformity characteristics of the weak models which go into the stochastic model being built. The fact that we use thick weak models means that these enrichment and uniformity characteristics, which are based on performance on training examples, carry over to some extent to test points. In effect, then, while the mass functions associated with the training points for the different classes are shrinking (variances approach 0) away from one another as the number of weak models increases, the mass functions associated with the test points are also shrinking away from one another. But due to the degradation in enrichment and uniformity characteristics of the stream in going from training to test sets, both the separation of (i.e., the expectations of) the mass functions, and their rates of shrinkage (i.e., the rates at which variances decrease), will usually be less for the test points. So, it may well happen that a point will be reached where the training functions lie entirely on opposite sides of the threshold, (error rate 0.0 on the training set), yet the variances of the test set functions are not yet small enough so that the functions associated with the test set lie on opposite sides of the threshold. In this case, as the number of weak models increases, the training set performance error rate will clearly not improve further; however, the variances of the training set mass functions will continue to decrease, as will the variances on the test set mass functions, and so the classifier will continue to improve showing, among other things, decreased error rate on the test set.

2.9 Some Additional Wrinkles

Needless to say, the examples we've considered so far are very simple and are just meant to illustrate the underlying concepts behind the method of stochastic discrimination. Without much effort, we can consider just slightly more complex synthetic examples for which our solution to the uniformity problem would not work quite so neatly. For example, by making the region occupied by the reds somewhat star-shaped, some red points couldn't even be captured by 2x2 squares which we required to lie entirely within the red region. Thus, we would have to go back to the more general form of enrichment used in our initial algorithm utilizing thick weak models. In this case, we would still have examples where, in an attempt to promote uniformity among red points, red points on the boundary of the red region could not have their coverage increased without also increasing the coverage of neighboring green points, points whose coverage we probably would want to *decrease* in order to promote uniformity among greens. In effect, this conflict would make it literally impossible to induce perfect uniformity by subsampling any stream of sufficiently thick weak models. In other words, no matter what we do in subsampling our stream, perfect uniformity could not be achieved in this case. In actual practice, the hope is that such problematic points on boundaries of class regions do not account for a large enough percentage of the population to cause significant problems. In our experience, this hope is usually realized.

2.10 Multiclass Problems

The transition from 2-class problems, such as that considered above, to arbitrary m-class problems, is completely straightforward and adds little to the underlying theory. This is discussed in [9]. We might add one small point here, however.

The idea behind handling m-class problems is to break them into a set of 2-class problems. In [9], two ways of doing this are mentioned, namely, considering m-many problems of the form "class i" vs. "classes other than i" for every i less than m, or considering $m(m-1)$-many problems of the form "class i" vs. "class j" for every $i \neq j$ less than m. In the original algorithmic implementations of SD, the first method was used. However, due to advantages gained from what we referred to as the secondary stochastic effect, [9] went into more detail about, and, in fact, used an implementation based on the second method.

Recent advances in algorithm design, such as the inclusion of automatic scaling routines for regulating the size of weak models, as well as the obvious advantage of only having to deal with m-many binary problems instead of $m(m-1)$-many, have recently led us back to favoring

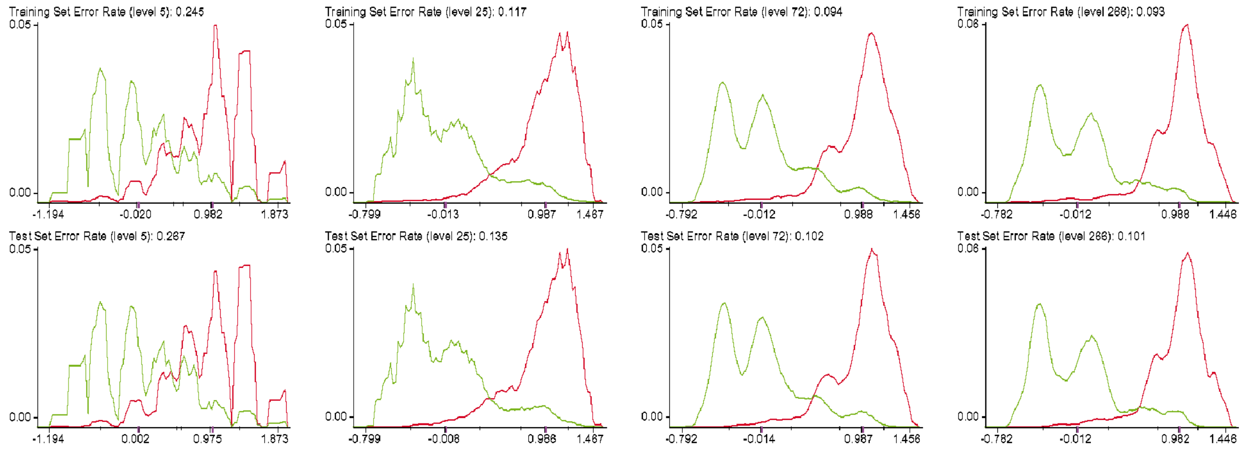

Fig. 13. Gene splicing problem, training/test sets, w.m.stream: unions of rectangles.

method 1. The idea is just as in [9], but we can make things somewhat simpler here. For each i, we run our uniform, training/test set, algorithm, **trte.unif**, on the problem where the "reds" are all points of class i and the "greens" are all points of classes other than i, with the one change of normalizing the random variable $probty_q$ by replacing

$$\text{Update } number_q := number_q + C_s(q)$$

with

$$\text{Update } i.number_q := i.number_q + \left(\frac{C_S(q) - Frac.greens}{Frac.reads - Frac.greens}\right)$$

in each of the m-many subalgorithms.

The primary effect of this change is to cause, for each $i \leq m$, the expectation of $i.probty_q$ to be 1.0 for red points q, and to be 0.0 for green points q. So, quite simply, to classify any given point q, one need only evaluate each random variable $i.probty_q$ and classify q as being of class i_0, where i_0 is the index such that $i_0.probty_q$ is largest.

Other than this, enrichment and uniformity are handled individually for each of the m-many problems, basically as before.

Needless to say, we are now left with m-many pairs of mass functions for the m-many component problems we are now dealing with. In order to simplify visualization, in all future graphical depictions of mass functions, the "upper" mass function will actually be the average of the "red" mass functions over all m-many problems and the "lower" mass function will actually be the average of the "green" mass functions over all m-many problems.

2.11 The Real World

In actual practice, we consider as possible weak models, finite unions of rectangularly shaped subsets of the given feature space and just require the simpler enrichment step first introduced in our second algorithm. While the uniformity algorithm used is quite similar to that used above in our uniform, training/test set, algorithm, **trte.unif**, it must be modified slightly to take into account these more complex weak models and the more general enrichment scheme. The resulting implementation of SD, which we will use in actual experiments reported on henceforth, in this paper, is known as SDK.

Of course, as alluded to above, it is usually impossible in practice on "real-world" problems to force perfect uniformity. However, one can do remarkably well. For example, let us consider an example from the University of California, Irvine database concerning gene splicing. This is a 3-class problem containing 3,190 examples sitting in a discrete feature space of size 8^{60}. We randomly broke the set of examples into training and test sets of approximately

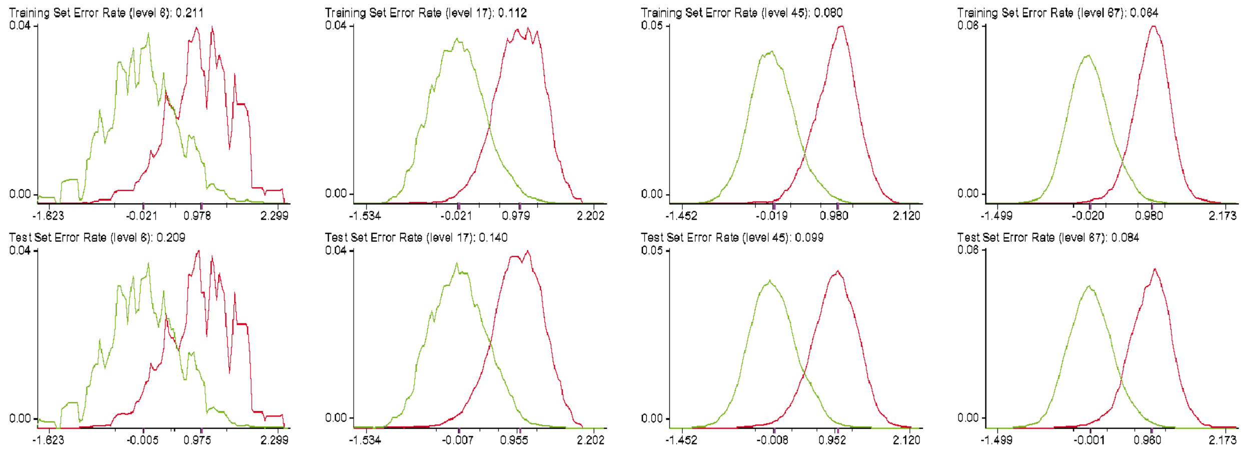

Fig. 14. Gene Splicing problem, training/test sets, uniform w.m.stream: unions of rectangles.

equal size and ran our algorithmic implementation, SDK, on the result. For our first run, we removed all uniformity forcing code and the output is illustrated in Fig. 13. We then repeated the procedure, this time with the uniformity forcing code in place. The output is illustrated in Fig. 14. Keeping in mind that the mass functions shown are now averages (we are dealing here with a 3-class problem), the output is exactly what our discussion above would lead one to expect. In particular, note that the expectations of the mass functions associated with the training set are further apart than those associated with the test set (since enrichment of the weak model stream is carried out by looking only at the training set and so the degree of enrichment as measured on the test set is less), and, of course, note the subpeaks on the mass functions associated with the run which didn't force uniformity.

3 The Underlying Mathematics

3.1 Introduction

The discussion above involving separating probability mass functions is intended for motivational purposes. In order to prove rigorously that the method operates as indicated, we start on a very different track. For the weak models from which our classifiers are built are (randomly) chosen from the collection of all subsets of an underlying feature space, and as such, the random variables we consider first in a mathematical treatment of this topic are random variables defined on sample spaces whose members are *subsets* of the feature space rather than *members* of the feature space.

3.2 The Sample Space

Specifically, let F denote the underlying feature space associated with a given 2-class pattern recognition problem (let R and G denote the two classes), and let $\mathbf{F}$ denote the power set of F, that is, the set of all subsets of F. We assume that F is finite, and as such, will use counting measures for calculating probabilities as needed. If probabilities are taken with respect to the underlying feature space, we will use the usual notation "Pr"; however, if they are taken with respect to the space $\mathbf{F}$, we will use the notation "$Pr_{\mathbf{F}}$."

Our first objective is to narrow down the space $\mathbf{F}$ along the lines discussed in our motivational sequence above, namely, to restrict our attention to a subspace of $\mathbf{F}$ whose elements 1) do NOT have the same chance of capturing R points as of capturing G points (see Fig. 5); and 2) provide a uniform cover (see Fig. 11).

Point 1 is easy to define:

Definition 3.1. *A subset $\mathbf{M}$ of $\mathbf{F}$ is said to be (R,G)-enriched if*

$$inf\{|Pr(M|R) - Pr(M|G)| \mid M \in \mathbf{M}\} > 0.$$

Point 2 is more difficult. We start by introducing the following notation:

Definition 3.2. *Given a subset $\mathbf{M}$ of $\mathbf{F}$ and a pair of real numbers (x, y), $\mathbf{M}_{(x,y)}$ denotes the set of M in $\mathbf{M}$ such that $Pr(M|R) = x$ and $Pr(M|G) = y$.*

It turns out to be easy to prove (see [9]) that the requirement that two points of a given class are equally likely to be captured by a weak model (of a given size), is equivalent to the following:

Definition 3.3. *A subset $\mathbf{M}$ of $\mathbf{F}$ is said to be (R,G)-uniform if for every q in either R or G, and every nonempty subset of $\mathbf{M}$ of the form $\mathbf{M}_{(x,y)}$, $Pr_{\mathbf{F}}(q \in M|M \in \mathbf{M}_{(x,y)})$ is equal to x if q is a member of R, and is equal to y if q is a member of G.*

3.3 The Base Random Variable

Let, then, $\mathbf{M}$ be a given (R,G)-enriched, (R,G)-uniform, subspace of $\mathbf{F}$. We define a random variable $X_{(R,G)}$ on the sample space $F \times \mathbf{M}$ by

$$X_{(R,G)}(q, S) = \left(\frac{C_S(q) - Pr(S|G)}{Pr(S|R) - Pr(S|G)}\right),$$

(where C_S is the characteristic function of the set S).

The following proposition is easy to prove.

Proposition 3.4 *For a given point q in F, consider the random variable $X^q_{(R,G)}$ which results from restricting $X_{(R,G)}$ to the space $\{q\} \times \mathbf{M}$. Then the variables $X^q_{(R,G)}$, for q restricted to R are all identically distributed, and have common expectation 1. The variables $X^q_{(R,G)}$, for q restricted to G are also all identically distributed. They have common expectation 0.*

Note once again that we are talking here about random variables defined on spaces of *subsets of the underlying feature space*. We now randomly sample from such spaces of weak models.

Let t be a given positive integer, and let us denote by $X^k_{(R,G)}$ the random variable corresponding to $X_{(R,G)}$ associated with the kth of t-many trials, that is, the random variable defined on the sample space $F \times \mathbf{M}^t$ whose value at any point $(q, (S_1, S_2, \ldots, S_t))$ is $X_{(R,G)}(q, S_k)$. Let $Y^t_{(R,G)}$ denote the random variable

$$\frac{\left(\sum_{k=1}^{t} X^k_{(R,G)}\right)}{t}.$$

By the Central Limit Theorem, as t increases, the probability distribution function of $Y^t_{(R,G)}$ approaches the normal probability distribution function having expectation that of $X_{(R,G)}$, and having variance $\frac{1}{t}^{th}$ that of $X_{(R,G)}$.

3.4 The Classifier

Thus, consider the following method of classification: given a point q in the feature space F, simply evaluate $Y^t_{(R,G)}(q, \mathbf{s}^t)$ at some (randomly chosen) member $\mathbf{s}^t = (S_1, S_2, \ldots, S_t)$ of $\mathbf{M}^t$, and proceed to classify q as being of type R iff that value is greater than 0.5. In light of our remarks above, by choosing t sufficiently large, one can make the probability that this method of classification makes an error as small as desired.

One can see this point clearly by looking at pictures of the probability mass functions associated with the random variables $Y^t_{(R,G)}(q, \mathbf{s}^t)$ for given fixed q. In particular, for sufficiently large t (so that we can effectively replace thinking in terms of probability mass functions, with thinking in terms of probability density functions) the pdf associated with points of type-G has a mean of 0.0 and a variance of, say, σ^2_G, and that associated with points of type-R has a mean of 1.0 and a variance of σ^2_R. An error occurs at

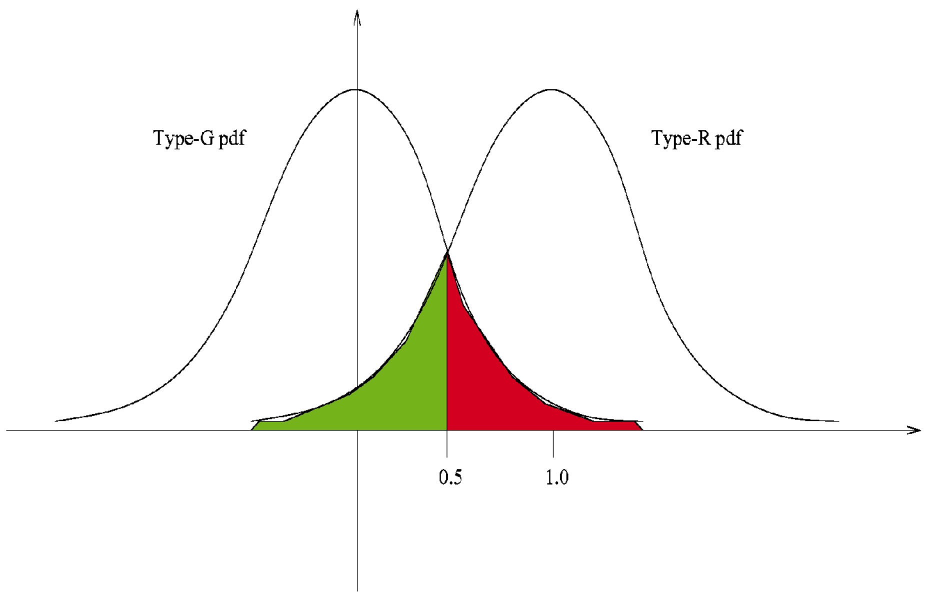

Fig. 15. Probability of error.

a given point q, if q is of type-G and the value of $Y^t_{(R,G)}(q, \mathbf{s}^t)$ for the (randomly chosen) member $\mathbf{s}^t$ of $\mathbf{M}^t$ is greater than 0.5, or q is of type-R and the value of $Y^t_{(R,G)}(q, \mathbf{s}^t)$ for the (randomly chosen) member $\mathbf{s}^t$ of $\mathbf{M}^t$ is less than or equal to 0.5. Thus, pictorially, the probability that a type-G point is incorrectly classified is the area of the red-shaded region in Fig. 15, and the probability that a type-R point is incorrectly classified is the area of the green-shaded region in Fig.15.

One must be careful here for these probabilities are taken with respect to the sample space $\mathbf{M}^t$. Let us identify a classifier for our two-class problem with a subset of feature space F (that subset being the points classified as being in R), and let us define, for a given t, and member $\mathbf{s}^t = (S_1, S_2, \ldots, S_t)$ of $\mathbf{M}^t$, the classifier M_{s^t} to be

$$\left\{q | Y^t_{(R,G)}(q, \mathbf{s}^t) > 0.5\right\}.$$

Then, what we are saying is that given any point q in the feature space, the probability that a member $\mathbf{s}^t$ of $\mathbf{M}^t$ produces a classifier M_{s^t} which makes a mistake at classifying q is the area of the shaded region in Fig. 15, that is, the area of the intersection of the regions bounded below by the x-axis and above by the graphs of the type-G and type-R pdfs. Since, as t approaches infinity, the variances of these pdfs approach 0, we can say that as t approaches infinity, the area of the (shaded) intersection, and hence, the probability of incorrect classification, approaches 0.

Unfortunately, the "probability of incorrect classification" we are talking about here is quite different from what is usually meant by "probability of error" for a classifier. We are taking probabilities *with respect to sample spaces of the form* $\mathbf{M}^t$, and not with respect to the underlying feature space. In order to fully understand the theory of stochastic discrimination, it is important to understand this point. At the expense of repeating ourselves, let us make this contrast clear. In the usual parlance, when a classifier is said to have an error rate of e, it is understood that the probability, with respect to the feature space, that a randomly chosen point is misclassified, is e; when we say that level-t stochastic classifiers have probability d of misclassification, we mean that for any fixed, generic point q in the feature space, the probability, with respect to the sample space $\mathbf{M}^t$, that a randomly chosen point $\mathbf{s}^t$ produces a classifier $M_{\mathbf{s}^t}$ which makes a mistake at classifying q as d.

3.5 The Duality Lemma

We can get a better handle on evaluating the accuracy of classifiers built by this method of stochastic discrimination, and in particular, develop the machinery necessary to prove that such classifiers generalize to test data, by examining the relationship between these two, seemingly disparate, notions of "error." We say this because for the case of stochastic discrimination, error is based solely on the characteristics of the probability density functions of the $Y^t_{(R,G)}(q, \mathbf{s}^t)$ associated with a certain stochastic process, and these probability density functions, in turn, are based on the characteristics of the underlying sample space of weak model subsets of the feature space. But if these weak model subsets generalize well from training to test data, then these characteristics of the sample space of weak model subsets of the feature space which determine the probability density functions of the $Y^t_{(R,G)}(q, \mathbf{s}^t)$ will be the same whether q is a training point or a test point. As such, the process of evaluating error will be the same whether done with respect to the training set or the test set. It is this basic idea which is behind our proof of the fact that classifiers built by this method of stochastic discrimination generalize to nontraining data.

We now proceed to relate the two notions of "error." For a given subset C of the feature space F, let $g^C_{\mathbf{s}^t}$ denote the probability mass function of the random variable $Y^t_{(R,G)}(q, \mathbf{s}^t)$ restricted to $q \in C$. Then it is clear that $Pr(M_{\mathbf{s}^t} | G)$ is equal to

$$\int_{[0.5,\infty)} g^G_{\mathbf{s}^t}$$

and $Pr(M_{\mathbf{s}^t} | R)$ is equal to

$$\int_{[0.5,\infty)} g^R_{\mathbf{s}^t}.$$

Thus, the "classical" probability of error, $e_{M_{\mathbf{s}^t}}$, for the classifier $M_{\mathbf{s}^t}$ is given by

$$e_{M_{\mathbf{s}^t}} = \int_{(-\infty,0.5]} g^R_{\mathbf{s}^t} + \int_{[0.5,\infty)} g^G_{\mathbf{s}^t}.$$

Of course, any such classifier, $M_{\mathbf{s}^t}$ is a function of the "randomly chosen" sample $\mathbf{s}^t$ in $\mathbf{M}^t$, and so our interest is really in the "expected error" of stochastic models built from samples from $\mathbf{M}^t$, that is, in the value

$$e(t) = \int_{(-\infty,0.5]} g^R_t + \int_{[0.5,\infty)} g^G_t$$

where, for any subset C of F, and any real number r, $g^C_t(r)$ is equal to the expectation of the random variable $g^C_{\mathbf{s}^t}(r)$ defined on the sample space $\mathbf{M}^t$.

So, how do we tie the probability mass functions g^R_t and g^G_t needed for classical calculation of error, to the mass functions of the random variables $Y^t_{(R,G)}(q, \mathbf{s}^t)$ used above in our discussion of "probability of misclassification?" With the so-called duality lemma (see [9] for a proof):

Theorem 3.5. *Given any positive integer t, and any point q in R (q in G), g^R_t (g^G_t) is equal to the mass function of the random variable $Y^t_{(R,G)}(q, \mathbf{s}^t)$ defined on $\mathbf{M}^t$.*

One is now in a position to derive classical estimates of accuracy for stochastic classifiers. For example, on a very simple level, since for a given point q, $Y^t_{(R,G)}(q, \mathbf{s}^t)$ is a sum of independent identically distributed random variables, we can use Chebyshev's inequality to see that for any given q,

$$Pr_{\mathbf{M}^t}\Big(|Y^t_{(R,G)}(q, \mathbf{s}^t) - E(X_{(R,G)}(q, S))| \geq 1/h\Big) \leq \frac{\sigma^2 h^2}{t},$$

where σ^2 is the variance of $X_{(R,G)}(q, S)$.

By Proposition 3.4, 0.5 is the mean of the expectations $E(X_{(R,G)}(q, S))$ (for type-R q and type-G q), and so, by taking h equal to 2, we immediately have that

$$\int_{(-\infty,0.5]} g^R_t \leq \frac{2}{t}\sigma^2_R$$

and

$$\int_{[0.5,\infty)} g^G_t \leq \frac{2}{t}\sigma^2_G,$$

where σ^2_R and σ^2_G are the variances of $X_{(R,G)}(q, S)$ for q in R and G, respectively. Thus,

$$e_{M_{\mathbf{s}^t}}(t) \leq \frac{2}{t}(\sigma^2_R + \sigma^2_G).$$

Again, this is a very simple estimate, presented here for illustrative purposes only. For a more elaborate statistical analysis of such issues, we refer the reader to [3].

3.6 Classifier Generalization

What about the notion of generalization to test data? Before one can even touch this issue, there has to be some understanding concerning the relationship between training and test sets. There are many ways to effect this, the most common involving some sort of topological proximity of like points between training and test sets (as illustrated above in our Fig. 8). We feel, however, that a very elegant, general way to define the notion of a training set being "representative" is simply to require that there exist *some* uniformly distributed collection of subsets of the feature space, such that each subset is better than random at distinguishing among the classes and such that each subset generalizes from training to test data. Thus, in effect, the notion is really "there exists an appropriate subspace $\mathbf{N}$ of $\mathbf{F}$ such that the training set is $\mathbf{N}$-representative of the test set." This doesn't require a topological relationship between training and test sets, and as such allows for a number of interesting alternative possibilities. It certainly seems to constitute a reasonable, minimal set of conditions. For a full development of these notions, we refer the reader to [9] and [10].

Given this, let us now assume that instead of carrying out the development of classifiers above using all examples for training, we just use training subsets R^{tr} and G^{tr}. We are now interested in evaluating the accuracy of the resulting stochastic classifier on test sets R^{te} and G^{te}. In this more general context, let us define, for any two subsets A and B of F, the expected error (at separating classes A and B) of level-t stochastic classifiers by

$$e(t, A, B) = \int_{(-\infty,0.5]} g^A_t + \int_{[0.5,\infty)} g^B_t.$$

Then the theorem is this:

Theorem 3.6. *Given any pair of subsets (A, B) of F, if (R^{tr}, G^{tr}) is $\mathbf{M}$-representative of (A, B), then $e(t, A, B) = e(t, R^{tr}, G^{tr})$.*

The proof of this theorem revolves around the duality lemma, and can be found in [9]. In essence, both $e(t, A, B)$ and $e(t, R^{tr}, G^{tr})$ are based on certain probability mass functions. By the duality lemma, these probability mass functions are equal to probability mass functions of random variables defined on the sample space of weak models. However, given that (R^{tr}, G^{tr}) is representative of (A, B) with respect to this space of weak models, these random variables, specifically, the $Y^t_{(R^{tr},G^{tr})}(q, \mathbf{s}^t)$, and $Y^t_{(A,B)}(q, \mathbf{s}^t)$, defined on $\mathbf{M}^t$, are identically distributed. Thus, by the duality lemma,

$$\begin{aligned} e(t, A, B) &= \int_{(-\infty,0.5]} g^A_t + \int_{[0.5,\infty)} g^B_t \\ &= \int_{(-\infty,0.5]} g^{R^{tr}}_t + \int_{[0.5,\infty)} g^{G^{tr}}_t = e(t, R^{tr}, G^{tr}) \end{aligned}.$$

The details concerning all of this can be found in [9] and [10].

3.7 Multiclass Problems

We deal with problems involving m-many classes, R_1, R_2, ... R_m, by breaking things into m-many 2-class problems. For any i, let G_i denote $\cup_{j\neq i} R_j$. Then, for each of the m-many problems "R_i vs. G_i," we carry out a development as above, and are left, for any positive integer t, with m-many functions $Y^t_{(R^{tr}_i, G^{tr}_i)}(q, \mathbf{s}^t_i)$. Then as above, assuming that t is large enough, if we choose samples $\mathbf{s}^t_i$ from (R^{tr}_i,G^{tr}_i)-enriched,

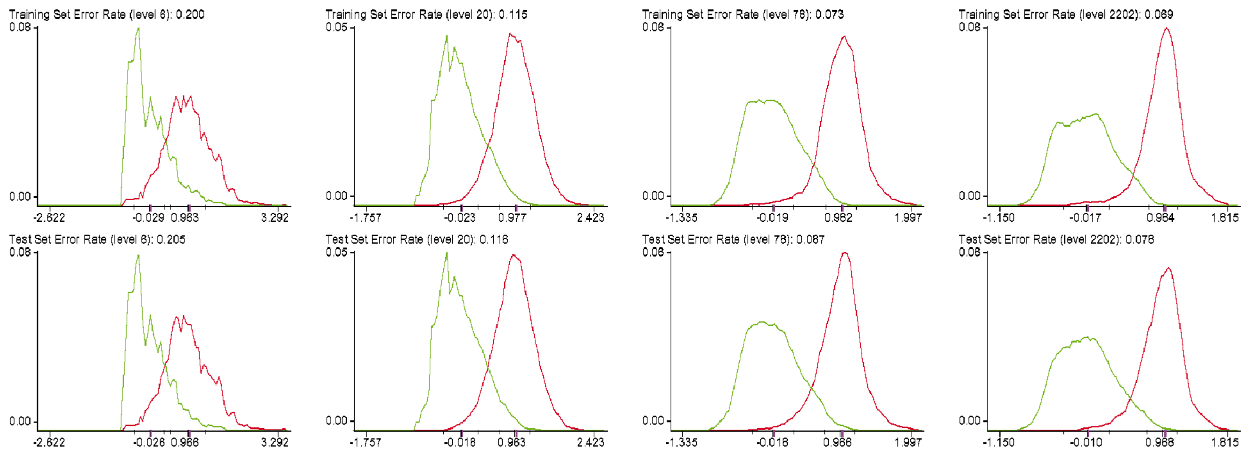

Fig. 16. Gene splicing problem, training/test sets, boosted w.m.stream: unions of rectangles.

(R_i^{tr},G_i^{tr})-uniform, spaces, then for any point q in the feature space, if q is of class i, then the value of $Y^t_{(R_i^{tr},G_i^{tr})}(q,\mathrm{s}_i^t)$ should be close to 1, and if q is not of class i then the value of $Y^t_{(R_i^{tr},G_i^{tr})}(q,\mathrm{s}_i^t)$ should be close to 0. Thus, if we simply classify a given point q to be of class i_0, where i_0 is the index for which $Y^t_{(R_{i_0}^{tr},G_{i_0}^{tr})}(q,\mathrm{s}_{i_0}^t)$ is largest, then with high probability (assuming that t is large enough), this classification of q will be correct.

3.8 Theory vs. Practice

One additional remark: In actual practice, many of our theoretical definitions concerning, for example, uniformity and representativeness, are not met exactly, but rather to within some error ϵ. It turns out that stochastic discrimination is quite robust and small values of ϵ propagate to small divergences from theoretical prediction. For detail on this, as well as for the derivation of a number of statistical estimates of classifier accuracy for this general method, we refer the reader to [3].

4 A Remark Concerning Boosting

Schapire's boosting algorithm has attracted a great deal of attention over the past few years. It is a method for combining classifiers trained on iteratively-modified distributions of examples, and while it has produced extremely good results for many problems in statistical pattern recognition, a completely satisfactory explanation for its success at generalizing performance to test data has not yet been forthcoming. We propose an explanation here.

Boosting algorithms iteratively modify training set distributions so as to deemphasize "easy" points, namely, points correctly classified by earlier "weak classifiers." If one views such algorithms from the appropriate perspective, this notion of redistribution provides a particular approach toward promoting uniformity. This is an algorithmic issue and clearly the uniformity-forcing algorithms we have discussed thus far are of a very different nature; however, in the context of SD, the goal of uniformity can be achieved in many ways.

In [11], we develop a theoretical base for boosting algorithms and discuss these issues in some detail. However, just to get some sense for the use of boosting as a means of achieving uniformity, we took the AdaBoost algorithm of Freund and Schapire (see [4]) and modified it so as to be usable within the program SDK in place of the default uniformity-forcing algorithm. We then took the same gene splicing dataset discussed above (see Figs. 13 and 14) and ran the modified version of SDK on it. Specifically, we randomly generated weak models as before, but after filtering them for enrichment, we filtered them a second time so as to minimize the AdaBoost-error based on the weight vector at that point in time. Once such a weak model was selected, we readjusted the weight vector as called for by AdaBoost. The results for this run are illustrated in Fig. 16. While the pdfs don't look quite as nice as those in Fig. 14 produced by SDK's standard uniformity forcing algorithm, by comparing them to those in Fig. 13, it is clear that an effective uniformity forcing algorithm is a play.

Of course AdaBoost is a selfcontained learning algorithm and we just used its weak learning iteration segment in producing this uniformity forcing algorithm. Using AdaBoost's final hypothesis output was not relevant here under SD's paradigm. However, by viewing boosting as a uniformity-forcing algorithm, the underlying theory of stochastic discrimination comes into play, and in particular, the theoretical basis behind the generalization to nontraining data of models built by the method of stochastic discrimination applies, with trivial modification, to show the generalization to nontraining data of models built by the method of boosting.

5 Experimental Results

We carried out a series of experiments using an algorithmic implementation of SD known as SDK. As discussed previously, the implementation itself is along the lines of the pseudocode introduced earlier in this paper, with relatively minor modifications as needed to deal with the fact that feature spaces, in general, are more complex than 18 by 18 grids.

More specifically, given a general, n-dimensional feature space F, SDK first embeds F into Euclidean n-space (by mapping any nonnumeric field values to nonnegative integers). For each coordinate k, $1 \leq k \leq n$, it then

data	size	FIA	ABO	ABA	FID	DBO	DBA	C45	5BO	5BA	SDK
crx	690	14.5	14.4	14.5	14.5	13.5	14.5	15.8	13.8	13.6	*12.4*
dia	768	26.1	*24.4*	26.1	27.8	25.3	26.4	28.4	25.7	*24.4*	25.5
gls	214	51.5	51.1	50.9	49.7	48.5	47.2	31.7	22.7	25.7	*20.3*
hrt	303	27.8	18.8	22.4	27.4	19.7	20.3	26.6	21.7	20.9	*17.4*
hep	155	19.7	18.6	16.8	21.6	18.0	20.1	21.2	16.3	17.5	*16.2*
ion	351	17.8	8.5	17.3	10.3	6.6	9.3	8.9	*5.8*	6.2	6.2
iri	150	35.2	4.7	28.4	38.3	4.3	18.8	5.9	5.0	5.0	*4.2*
lab	57	25.1	8.8	19.1	24.0	7.3	14.6	15.8	13.1	11.3	*6.1*
let	20000	92.9	92.9	91.9	92.3	91.8	91.8	13.8	*3.3*	6.8	*3.3*
sat	6435	58.3	58.3	58.3	57.6	56.5	56.7	14.8	8.9	10.6	*8.7*
seg	2310	75.8	75.8	54.5	73.7	53.3	54.3	3.6	*1.4*	2.7	1.9
son	208	25.9	16.5	25.9	31.4	15.2	26.1	28.9	19.0	24.3	*10.6*
soy	683	64.8	64.5	59.0	73.6	73.6	73.6	13.3	6.8	12.2	*5.9*
spl	3190	37.0	9.2	35.6	29.5	8.0	29.5	5.8	*4.9*	5.2	*4.9*
veh	846	64.3	64.4	57.6	61.3	61.2	61.0	29.9	22.6	26.1	*22.1*
vot	435	4.4	3.7	4.4	4.0	4.4	4.4	*3.5*	5.1	3.6	*3.5*
wsc	699	8.4	4.4	6.7	8.1	4.1	5.3	5.0	3.3	3.2	*2.6*

Fig. 17. Experimental results—Freund-Schapire.

determines maximum and minimum values, max_k and min_k, respectively, among training points for that coordinate. Weak models are now formed by simply taking finite unions of rectangular parallelopipeds $R \subseteq F$, each of which is generated as follows: 1) randomly choose a point $(q_1, q_2, \ldots, q_n)$ in the training set; 2) for each k between 1 and n randomly generate upper and lower bounds upr_k and lwr_k such that $min_k \leq lwr_k \leq q_k \leq upr_k \leq max_k$; 3) let R denote the set of $(x_1, x_2, \ldots, x_n)$ in F such that for each k, $1 \leq k \leq n$, $lwr_k \leq x_k \leq upr_k$. The only problem-specific modification used, if dictated by extremely "sparse" datasets, might have SDK further increase the "thickness" of weak models by pushing a number of lwr_k and upr_k out to their limits min_k and max_k, respectively, or have SDK increase the number of parallelopipeds in the finite union so as to increase the percentage of F covered by weak models. All problems considered here shared a key characteristic with our synthetic 18 by 18 example, namely that there existed a spatial relationship between points of like class. Thus as before, the evident "thickness" of the weak models just described, translated into an ability of these weak models to generalize. (Note that weak models of this sort can alternately be viewed simply as finite unions of finite intersections of regions determined by splitting F with axis-orthogonal hyperplanes.)

We retrieved by anonymous FTP, datasets from two major repositories for sets of standardized problems in pattern recognition, that at the University of California at Irvine, and that at the University of Porto in Portugal.

From Irvine, we chose 17 different datasets, namely, Australian credit (henceforth denoted "crx"), Pima diabetes (dia), glass (gls), Cleveland heart (hrt), hepatitis (hep), ionosphere (ion), iris (iri), labor (lab), letter (let), satimage (sat), segment (seg), sonar (son), soybean-large (soy), splice (spl), vehicle (veh), vote (vot), and Wisconsin breast cancer (wsc). Our decision about which sets to use was based solely on popularity—of the many recent papers containing comparative studies of pattern recognition methods, these 17 sets tended to be studied more than any others from the Irvine collection. We compared our results to those reported by Freund and Schapire in [4]. We decided to use this paper since it focuses on boosting (see [4]) and bagging (see [2]), two of the most popular and powerful methods currently being studied in pattern recognition. Three underlying "weak learning algorithms," FindAttrTest (henceforth denoted, "FIA"), FindDecRule (FID), and Quinlan's C4.5 (C45) (see [13]), along with their boosted and bagged versions (denoted ABO, DBO, 5BO, ABA, DBA, and 5BA, respectively), for a total of nine methods, are reported on in [4] and in our runs on these datasets, we used the same study paradigms (either 10-fold cross validation, or training/test set, depending on the dataset) as used by Freund and Schapire. Our only change (made because of time constraints) was that for two of the datasets which used training/test sets, namely, letter and satimage, we did not rerun the study 20 times with different seeds, but just reported the results for our single run (seed 1) of each problem. For the 10-fold cross-validation studies, however, we reran each cross-validation 10 times using different initial seeds for a total of 100 runs per study and for the remaining training/test set problem, soybean-large, we reran the study 20 times with different seeds and averaged the results. The reader should refer to [4] for the details here.

The results for our runs, as well as for those reported in [4] are presented in Fig. 17. The values listed under each method are error rates for the different problems. In Fig. 18, we summarize these results graphically. Our focus is toward the relative ranks of the various methods over the 17 problems considered. Specifically, for each of the 10 methods (as listed on the x-axis), we provide a bar ranging from the best rank that method achieved on any of the problems, to the worst rank. That bar also has a left tic located at the mean of the ranks of that method across all 17 problems and a right tic located at the mode of the ranks (assuming a mode exists for that method). The methods are listed in order of their mean rank and our graph includes a line graph connecting the means of the ranks.

From Porto, 10 sets were publicly available. However, two of these sets (heart data and German credit data) involved nontrivial cost matrices and since the implementation SDK does not take issues of cost into account,

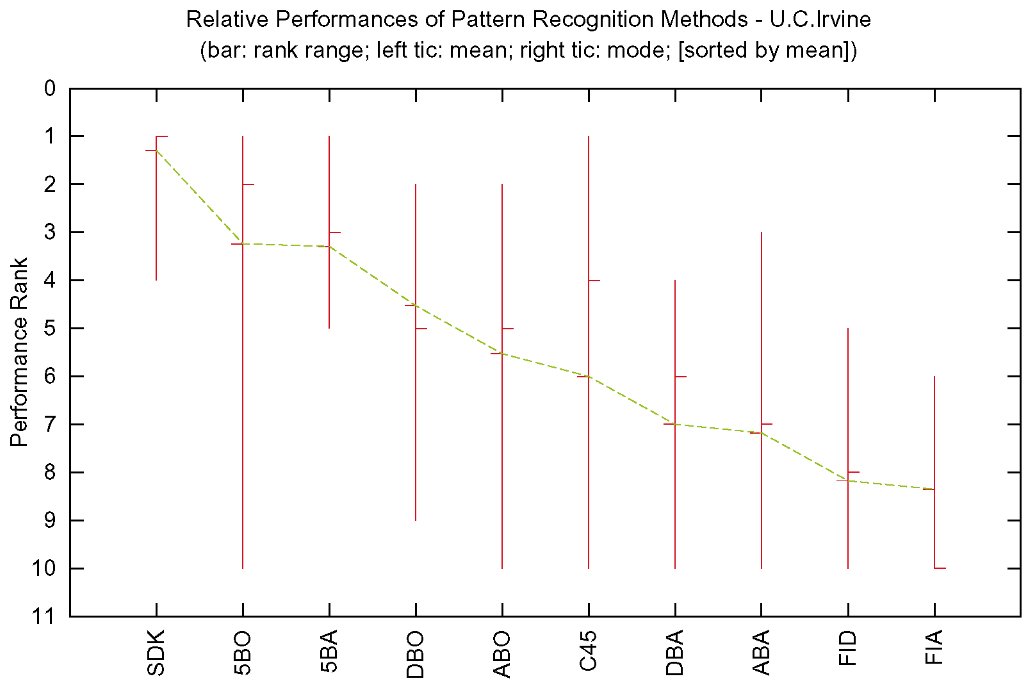

Fig. 18. Irvine comparisons.

we didn't use these sets. One additional set, namely the shuttle dataset, was extremely underrepresented in some classes (two test points for class seven out of a total of 58,000 training and test examples) so we also decided to eliminate that set from consideration. Of the remaining seven sets, we carried out studies using the same conditions as [12]. In particular, the Australian Credit Approval problem (henceforth denoted "crx") used

method	crx	dia	dna	let	sat	seg	veh
Ac2	0.181	0.276	0.245	0.245	0.157	0.031	0.296
Alloc80	0.201	0.301	0.064	0.064	0.132	0.030	0.173
BackProp	0.154	0.248	0.327	0.327	0.139	0.054	0.207
BayTree	0.171	0.271	0.124	0.124	0.147	0.033	0.271
Bayes	0.151	0.262	0.529	0.529	0.287	0.265	0.558
C4.5	0.155	0.270	0.132	0.132	0.150	0.040	0.266
Cal5	0.131	0.250	0.253	0.253	0.151	0.062	0.279
Cart	0.145	0.255	NA	NA	0.138	0.040	0.235
Castle	0.148	0.258	0.245	0.245	0.194	0.112	0.505
Cn2	0.204	0.289	0.115	0.115	0.150	0.043	0.314
Default	0.440	0.350	0.960	0.960	0.769	0.760	0.750
Dipol92	0.141	0.224	0.176	0.176	0.111	0.039	0.151
Discrim	0.141	0.225	0.302	0.302	0.171	0.116	0.216
IndCart	0.152	0.271	0.130	0.130	0.138	0.045	0.298
Itrule	0.137	0.245	0.594	0.594	NA	0.455	0.324
KNN	0.181	0.324	0.068	0.068	0.094	0.077	0.275
Kohonen	NA	0.273	0.252	0.252	0.179	0.067	0.340
LVQ	0.197	0.272	0.079	0.079	0.105	0.046	0.287
LogDisc	0.141	*0.223*	0.234	0.234	0.163	0.109	0.192
NewId	0.181	0.289	0.128	0.128	0.150	0.034	0.298
QuaDisc	0.207	0.262	0.113	0.113	0.155	0.157	*0.150*
Radial	0.145	0.243	0.233	0.233	0.121	0.069	0.307
SDK	*0.126*	0.233	*0.033*	*0.038*	*0.0865*	*0.021*	0.201
Smart	0.158	0.232	0.295	0.295	0.159	0.052	0.217

Fig. 19. Experimental results—Statlog.

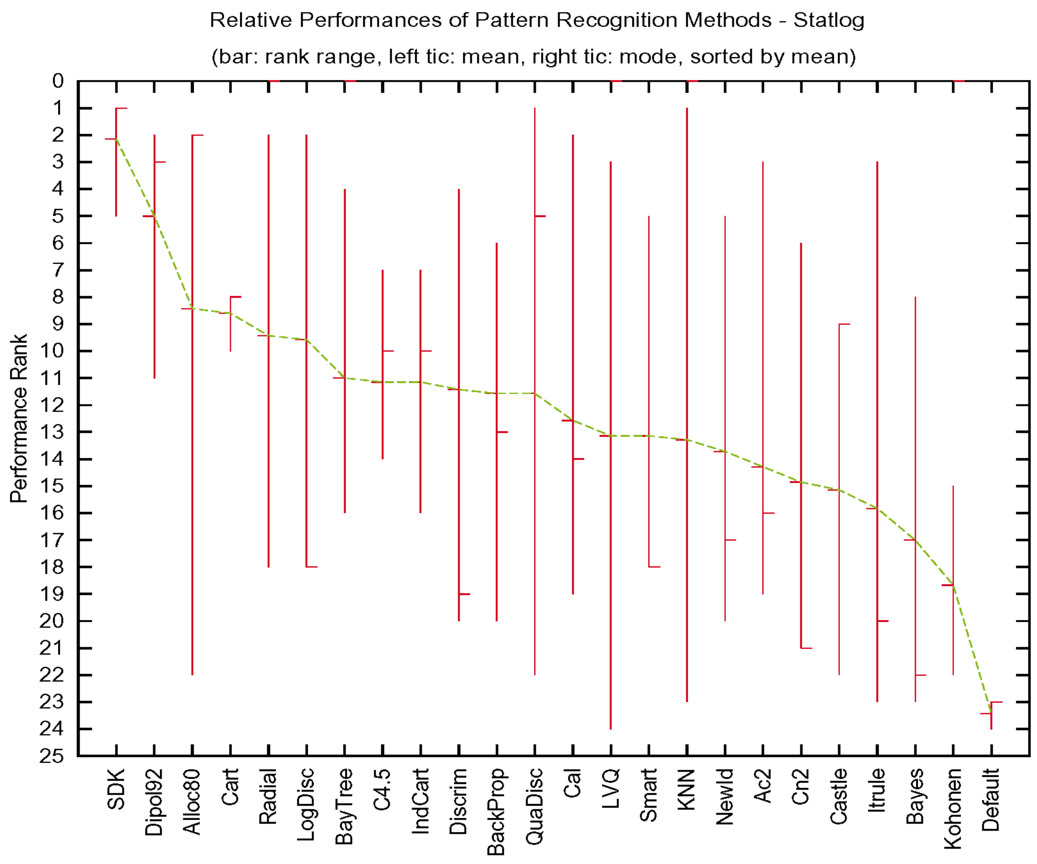

Fig. 20. Statlog comparisons.

10-fold cross-validation, the Pima Indians Diabetes problem (dia) used 12-fold cross-validation, the DNA problem (dna) used training (2,000 points) and test (1,186 points) sets, the Letter Image Recognition problem (let) used training (15,000 points) and test (5,000 points) sets, the Satellite Image problem (sat) used training (4,435 points) and test (2,000 points) sets, the Image Segmentation problem (seg) used 10-fold cross-validation, and the Vehicle Silhouette problem (veh) used 9-fold cross-validation.

The results for our runs, as well as for those reported in [12] are presented in Fig. 19. In Fig. 20, we graphically represent relative ranks as we did before in Fig. 18.

6 Conclusions

As a method, stochastic discrimination combines very weak components in such a way that the ability of the eventual classifier to generalize to nontraining data is comparable to the ability of the component pieces to generalize. Since by making the components very weak we can maintain their high generalization ability, SD can build *arbitrarily complex*, and highly accurate, classifiers which also generalize well. The theoretical basis for this resides in the structural characteristics of higher-order sample spaces of subsets of the underlying feature space and not on limiting VC dimension by restricting the complexity of the eventual classifier. In fact, the space of classifiers built by SD for most problems has infinite VC dimension. Limits on attainable error rate are dictated solely by the degree to which underlying weak models fail to generalize and the degree to which the space of such weak models fails to be enriched and uniform. However, these issues lie at the heart of what constitutes a solvable pattern recognition problem and in some sense provides a basis for establishing the optimality of stochastic discrimination as a general method for classifier construction. For a thorough discussion of these issues we refer the reader to [10].

In this paper, our goal was to bridge the gap between the theoretical promise shown for the method of stochastic discrimination in [8], [9] and its practical implementation as an effective, general method in pattern recognition. On 14 of 17 benchmark problems from the University of California, Irvine collection, the implementation of SD discussed here, SDK, outperformed both boosting and bagging applied to three underlying weak learning algorithms; and on five of the seven benchmark problems from the Statlog collection, SDK outperformed a comprehensive suite of 23 different methods. On only one problem, the Pima Indians diabetes study, did SDK finish out of the top 25 percent of methods considered. Here, it placed fourth out of 10.

Thus, there appears to be good evidence that the general method of stochastic discrimination is a very promising approach in the field of pattern recognition. As new algorithmic implementations are created, results should improve further. In particular, so far as we know, there has been little effort at adjusting the nature of the weak-model-stream, or of the enrichment and uniformity-forcing algorithms, based on specifics of any particular problem being considered. In fact, as discussed above, the results reported on here used a single program with essentially no adjustment of parameters from problem to problem. This may well have been the difficulty with respect to the Pima diabetes study. It will be a focus of further study.

ACKNOWLEDGMENTS

The author would like to thank T.K. Ho, E.G. Kleinberg, S.I. Kogitz, and A.S. Shekhawat for their useful comments based on preliminary versions of this paper. Many of their suggestions have been included here. This work uses software copyrighted by K Square Inc.

REFERENCES

[1] R. Berlind, "An Alternative Method of Stochastic Discrimination with Applications to Pattern Recognition," PhD thesis, SUNY/Buffalo, New York, 1994.
[2] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.
[3] D. Chen, "Statistical Estimates for Kleinberg's Method of Stochastic Discrimination," PhD thesis, SUNY/Buffalo, New York, 1998.
[4] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," *Proc. 13th Int'l Conf. Machine Learning,* pp. 148-156, July 1996.
[5] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting," *J. Computer and System Sciences,* pp. 119-139, 1997.
[6] T.K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 8, pp. 832-844, Aug. 1998.
[7] T.K. Ho, "Random Decision Forests," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 278-282, 1995.
[8] E.M. Kleinberg, "Stochastic Discrimination," *Annals of Math. and Artificial Intelligence,* pp. 207-239, 1990.
[9] E.M. Kleinberg, "An Overtraining-Resistant Stochastic Modeling Method for Pattern Recognition," *Annals of Statistics,* pp. 2,319-2,349, 1996.
[10] E.M. Kleinberg, "A Mathematically Rigorous Foundation for Supervised Learning," *Proc. First Int'l Workshop on Multiple Classifier Systems,* to appear.
[11] E.M. Kleinberg, "A Note on the Mathematics Underlying Boosting," preprint, to appear.
[12] D. Michie, D. Spiegelhalter, and C.C. Taylor, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.
[13] R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, Oct. 1993.
[14] V.N. Vapnik, *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, 1982.

Eugene M. Kleinberg received the BS degree in mathematics from the Massachusetts Institute of Technology (MIT) in 1967, and the PhD degree in mathematics from Rockefeller University in 1969. Following his graduate work, he was a C.L.E. Moore instructor of mathematics at MIT. He remained a member of the Mathematics Department at MIT for nine years, being promoted to assistant professor in 1971 and to associate professor in 1976. He has been a professor of mathematics at the State University of New York at Buffalo since 1980. While Dr. Kleinberg's research originally dealt with a range of topics in mathematical logic, primarily combinatorial set theory, for the past 16 years he has focused on issues in artificial intelligence, mainly machine learning and statistical pattern recognition.