2

Semi-Analytical Method for Analyzing Models and Model Selection Measures Based on Moment Analysis

AMIT DHURANDHAR
University of Florida
and
ALIN DOBRA
University of Florida

In this article we propose a moment-based method for studying models and model selection measures. By focusing on the probabilistic space of classifiers induced by the classification algorithm rather than on that of datasets, we obtain efficient characterizations for computing the moments, which is followed by visualization of the resulting formulae that are too complicated for direct interpretation. By assuming the data to be drawn independently and identically distributed from the underlying probability distribution, and by going over the space of all possible datasets, we establish general relationships between the generalization error, hold-out-set error, cross-validation error, and leave-one-out error. We later exemplify the method and the results by studying the behavior of the errors for the naive Bayes classifier.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Model selection, generalization error, classification

ACM Reference Format:

Dhurandhar, A. and Dobra, A. 2009. Semi-analytical method for analyzing models and model selection measures based on moment analysis. ACM Trans. Knowl. Discov. Data. 3, 1, Article 2 (March 2009), 51 pages. DOI = 10.1145/1497577.1497579 http://doi.acm.org/10.1145/1497577.1497579

1. INTRODUCTION

Consider the problem of estimating how a given classification algorithm (as opposed to a particular classifier) performs on a given joint distribution over

This work is supported by the National Science Foundation Grant, NSF-CAREER-IIS-0448264.
Authors' addresses: A. Dhurandhar (contact author), A. Dobra, University of Florida, Gainesville, FL-32611; email: amitdhur@ufl.edu.

the input-output space ($X \times Y$). As opposed to the general setup in machine learning where the distribution is unknown and only independently and identically distributed (i.i.d.) samples are available, in this scenario, *in principle*, the behavior of classification algorithm can be accurately studied. Should this problem be solvable efficiently, it offers an alternative line of study for classification algorithms and potentially unique insights into the *nonasymptotic* behavior of learning algorithms.

While the problem of estimating classification algorithm performance on a given distribution might look simple, solving it efficiently poses significant technical hurdles. The most natural way of studying a classification algorithm would be to sample N datapoints from the given distribution, train the algorithm to produce a classifier, test the classifier on a few sampled test sets, and report the average error computed over these test sets. A shortcoming of the aforementioned approach is that based on just one single instance of the algorithm (since the algorithm was trained on a single dataset of size N) we conclude about its general behavior. A straightforward extension of the preceding approach to make results more relevant in studying the algorithm would be to sample multiple datasets of size N, train on each of them to produce different classifiers, compute the test error for each of the classifiers, and calculate the average and variance of the obtained test errors. This procedure would be a better indicator of the behavior of the algorithm than the previous case, since we study multiple instances of the algorithm rather than just an isolated instance. Ideally, we would want to study the behavior of the algorithm by training it on all possible datasets of size N, producing a variety of classifiers and then evaluating the expected value and variance of the Generalization Error (GE) of each of these classifiers. The GE of a classifier ζ is given by

$$\begin{aligned} GE(\zeta) &= E[\lambda(\zeta(x), y)] \\ &= P[\zeta(x) \neq y], \end{aligned} \tag{1}$$

where $\lambda(.,.)$ is a 0-1 loss function, x is an input, and y is an output (class label), and the expectation is over the input-output space $X \times Y$.

The expected value and variance of GE over all possible classifiers[1] are denoted by

$$E_{\mathcal{Z}(N)}[GE(\zeta)] \tag{2}$$

$$Var(GE(\zeta)) = E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[GE(\zeta)GE(\zeta')] - E_{\mathcal{Z}(N)}[GE(\zeta)]^2, \tag{3}$$

where $\mathcal{Z}(N)$ represents the space of all possible classifiers produced by training the classification algorithm on all datasets of size N (denoted by $D(N)$), drawn from the joint distribution. Thus, the moments provide a natural and informative avenue for studying classification algorithms. The question that now arises is: Can we compute them efficiently? In the work that we present here, we provide a general framework for computing these quantities for an arbitrary classification algorithm efficiently. By extensive use of the linearity

[1]Going over $\mathcal{Z}(N)$ is more general than going over $D(N)$, since the classification algorithm can be randomized.

of expectation and change of the order of sums (and integrals), the moments of GE can be expressed in terms of the behavior of the classification algorithm on specific inputs rather than on the whole space, thus reducing the complexity (i.e., number of terms) from an exponential in the size of the input space to linear for the computation of the first moment and quadratic for the second moment.

The specific contributions we make in this article are as follows.

(1) We propose a new methodology to analyze statistical behavior of models and model evaluation measures. The methodology is based on defining random variables for quantities of interest, computing their moments, and then understanding their behavior by visualization. By introducing a probability space over the classifiers and computing the moments of the GE, we have the following two advantages over the theoretical results given by *Statistical Learning Theory* (SLT) [Vapnik 1998] from the point of view of studying learning methods: (a) We obtain qualitative results about classifiers based on the actual classification algorithms, not the expressiveness of the class of functions to which the classifier belongs to; and (b) the results are not as pessimistic. A disadvantage of our method is that we lose the ability to make generalized statements to the extent that SLT makes.
(2) We reduce the complexity (i.e., number of terms) of computing the moments from an exponential in the input size to a polynomial along with the complexity of each individual term in Theorem 1.
(3) We establish connections between the moments of hold-out-set error, cross-validation error, and leave-one-out error and the moments of the GE, indicating efficient strategies for computing these moments.
(4) We exemplify the general theory by deriving formulations for the moments of the Naive Bayes Classifier (NBC) in the discrete setting. We discuss strategies to make the analysis scalable. The strategies are truly generic in the sense that they can be used to "quickly" approximate cdf's of random variables in settings where moments are known. In the latter portion of the article we discuss ways of extending the analysis to the continuous NBC and other classification algorithms in general.
(5) We perform empirical studies wherein we first show that our method is more accurate than Monte Carlo used to directly estimate the moments. We explain reasons for this observed behavior. We then portray the manner in which the formulations can be used as an exploratory tool in studying *nonasymptotic* behavior of the model selection measures. Lastly, we study the NBC with respect to distributions built on real UCI datasets, which depicts the behavior of the classification algorithm not only on the particular datasets but rather on a class of datasets that are similar to the UCI datasets, since the moments are over all datasets of a particular size.

1.1 Applications of the Methodology

The primary goal of the methodology is creating an avenue in which learning algorithms can be studied precisely, that is, studying the statistical behavior

of a particular algorithm with respect to a chosen/built distribution. Next, we discuss the two most important perspectives in which the methodology can be applied.

1.1.1 *Algorithmic Perspective.* If a researcher/practitioner designs a new classification algorithm, he/she needs to validate it. Standard practice is to validate the algorithm on a relatively small (5–20) number of datasets and to report the performance. By observing the behavior of only a few instances of the algorithm the designer infers its quality. Moreover, if the algorithm underperforms on some datasets, it can be sometimes difficult to pinpoint the precise reason for its failure. If instead he/she is able to derive parametric expressions for the moments of GE, the test results would be more relevant to the particular classification algorithm, since the moments are over all possible datasets of a particular size drawn i.i.d. from some chosen/built distribution. Testing individually on all these datasets is an impossible task. Thus, by computing the moments using the parametric expressions, the algorithm would be tested on a plethora of datasets with the results being highly accurate. Moreover, since the testing is done in a controlled environment that is, all the parameters are known to the designer while testing, he/she can precisely pinpoint the conditions under which the algorithm performs well and the conditions under which the algorithm underperforms.

1.1.2 *Dataset Perspective.* If an algorithm designer validates his/her algorithm by computing moments, as mentioned earlier, it can instill greater confidence in the practitioner searching for an appropriate algorithm for his/her dataset. The reason is that if the practitioner has a dataset which has a similar structure or is from a similar source as the test dataset on which an empirical distribution was built and favorable results reported by the designer, then this would mean that the results apply not only to that particular test dataset, but also to other, similar types of datasets. Moreover, since the practitioner's dataset belongs to this similar collection, the results would also apply to his. Note that a distribution is just a weighting of different datasets and this perspective is used in the aforesaid exposition.

If the dataset is categorical, it can be precisely modeled by a multinomial distribution in the following manner. A multinomial is completely characterized by the probabilities in each of its cells (which sum to 1) and the total count N (sum of individual cell counts). The designer can set the number of cells in the multinomial to be the number of cells in his contingency table, with empirical estimates for the individual cell probabilities being the corresponding cell counts divided by the size of the dataset, which is the value of N. With this we have a fully specified multnomial distribution, using which we can compute the formulations, consequently characterizing the moments of the GE. Since the estimates for the cell probabilities are based on the available dataset, the true underlying distribution, of which this dataset is a sample, may have different values. This scenario can be accounted for by varying the cell probabilities and observing the variation in the estimates of GE. This would assist in deciphering the sensitivity to noise of the model in question. In the continuous case,

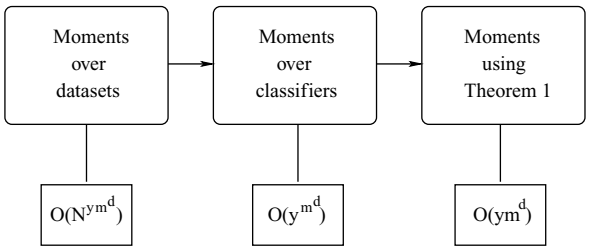

Fig. 1. Numbers of terms for three methods that analytically compute the first moment are shown. For the second moment the number of terms is just the square of the given complexity. N is the dataset size, y is the number of classes (output space), d is the number of attributes, and $O(m)$ is the number of distinct values per attribute. It can be seen that Theorem 1 significantly reduces the complexity without compromising on accuracy.

there is no such generic distribution (as the multinomial), but a popular choice could be a mixture of Gaussians (since universal approximators), though any other distribution may be chosen. It is important to notice here that the goal of computing the moments of GE is to also study the behavior of the classification algorithm on a set of datasets other than the original; hence, the distribution built on the original dataset need only be a reasonable approximation.

1.1.3 *Roadmap.* The rest of the article is organized as follows. In Section 2 we briefly survey related work. In Section 3 we derive Theorem 1. In the leftmost block of Figure 1 we see that the naive approach of analytically computing moments by summing over all possible datasets results in exponential number of terms to be computed in the input-output size with base being the sample size N. In the center block we see that by summing over classifiers the number of terms reduces to an exponential in the input space with base the output space Y. In the rightmost block we see that the application of Theorem 1 dramatically reduces the complexity to a low degree polynomial in the input-output space. In this section we also derive relationships between the moments of the Generalization error, hold-out-set error, multifold cross-validation error, and leave-one-out error, which are independent of any classification algorithm. This lays the foundation for the study of classification algorithms along with model selection measures. In Section 4 we illustrate the actual methodology by applying it to the NBC in one dimension. In Sections 5 and 6 we extend the method to the NBC in multiple dimensions and discuss ways of accurately and efficiently approximating the individual terms in the moments. These approximation techniques are generic in the sense that they can be used to approximate the cumulative distribution function of any random variable, given its moments. Hence, if in the future formulations for the moments of GE are developed for classification algorithms other than the NBC, these approximation methods will still be applicable. In Section 7 we report experiments with three major goals in mind: (i) to show that our method is more accurate than Monte Carlo used to directly estimate the moments; (ii) to portray the manner in which the derived formulations can be used as an exploratory tool to study model selection measures in conjunction with a classification algorithm; and (iii) to depict a way of building distributions on real data and studying the classification algorithm with

respect to these built distributions, which informs us about the behavior of the algorithm on the particular dataset as well as on a plethora of similar datasets (since the moments are over all possible datasets). We then discuss extensibility of the methodology in Section 8. Finally, we propose promising lines for future research and summarize the major developments in Section 9.

2. RELATED WORK

Statistical Learning Theory categorizes classification algorithms (actually the more general learning algorithms) into different classes, called concept classes. The concept class of a classification algorithm is determined by its Vapnik-Chervonenkis (VC) dimension, which is related to the shattering capability of the algorithm. Distribution-free bounds on the generalization error of a classifier built using a particular classification algorithm belonging to a concept class are derived in SLT. The bounds are functions of the VC dimension and the sample size. The strength of this technique is that by finding the VC dimension of an algorithm, we can derive error bounds for the classifiers built using this algorithm without ever referring to the underlying distribution. A fallout of this very general characterization is that the bounds are usually loose [Boucheron et al. 2005; Williamson 2001], which in turn results in making statements about any particular classifier weak.

The idea we pursue in this article is to define a class of classifiers induced by a given learning algorithm and i.i.d. data of a given size. Any member of this class can be viewed as a *sample classifier*, and the characterization of the class is strongly connected to the behavior of the classifier. This class of classifiers is much smaller than the classes considered in SLT. The downside of our method is the fact that we lose the strength to make generalized statements to the extent that SLT does. With this noted, in the next section we establish relationships between the statistical behavior of the generalization error and empirical errors for the members of this class of classifiers and relationships that have the character of general results (they hold irrespective of the classification algorithm that induces the class). We now review work done with respect to these empirical errors (model selection measures).

There is a large body of both experimental and theoretical work that addresses the problem of understanding various model selection measures. Shao [1993] showed that asymptotically Leave-One-Out(LOO) chooses the best but not the simplest model. Devroye et al. [1996] derived distribution-free bounds for cross-validation. The bounds they found were for the nearest-neighbor model. Breiman [1996] showed that cross-validation gives an unbiased estimate of the first moment of the generalization error. Though cross-validation has desired characteristics with estimating the first moment, Breiman stated that its variance can be significant. Theoretical bounds on LOO error under certain algorithmic stability assumptions were given by Kearns and Ron [1997]. They showed that the worst-case error of the LOO estimate is not much worse than the training error estimate. Elisseeff and Pontil [2003] introduced the notion of training stability. They showed that even with this weaker notion of stability good bounds could be obtained on the generalization error. Blum et al. [1999]

showed that v-fold cross-validation is at least as good as $\frac{N}{v}$ hold-out-set estimation on expectation. Kohavi [1995] conducted experiments on naive Bayes and C4.5 using cross-validation. Through his experiments he concluded that ten fold stratified cross-validation should be used for model selection. Moore and Lee [1994] proposed heuristics to speed-up cross-validation. Plutowski [1996] survey included papers with theoretical results, heuristics, and experiments on cross-validation. His survey was especially geared towards the behavior of cross-validation on neural networks. He inferred from the previously published results that cross-validation is robust. More recently, Bengio and Grandvalet [2003] proved that there is no universally unbiased estimator of the variance of cross-validation. Zhu and Rohwer [1996] proposed a simple setting in which cross-validation performs poorly. Goutte [1997] refuted this proposed setting and claimed that a realistic scenario in which cross-validation fails is still an open question.

The work we present here covers the middle ground between these theoretical and empirical results by allowing classifier specific results based on moment analysis. Such an endeavor is important since the gap between theoretical and empirical results is significant [Langford 2005].

3. MOMENT ANALYSIS

Probability distributions completely characterize the behavior of a random variable. Moments of a random variable give us information about its probability distribution. Thus, if we have knowledge of the moments of a random variable we can make statements about its behavior. In some cases, characterizing a finite subset of moments may prove to be a more desired alternative than characterizing the entire distribution, which can be computationally expensive to compute. This is precisely what we do when we study the behavior of the generalization error of a classifier and the error estimation methods called hold-out-error, leave-one-out error, and cross-validation error. Characterizing the distribution, though possible, can turn out to be a tedious task and studying the moments instead is a more viable option. As a result, we employ moment analysis and use linearity of expectation to explore the relationship between various estimates for the error of classifiers: Generalization Error (GE), Hold-out-set Error (HE), and Cross-validation Error (CE) (leave-one-out error is just a particular case of CE and we do not analyze it independently). The relationships are drawn by going over the space of all possible datasets. The actual computation of moments, though, is conducted by going over the space of classifiers induced by a particular classification algorithm and i.i.d. data, as can be seen in Section 4. This is done since it leads to computational efficiency. We interchangeably go over the space of datasets and space of classifiers as deemed appropriate, since the classification algorithm is assumed deterministic. In other words, we have

$$E_{x_1 \times \cdots \times x_N}[\mathcal{F}(\zeta(x_1, x_2, \ldots, x_N))] = E_{D(N)}[\mathcal{F}(\zeta[D(N)])] = E_{\mathcal{Z}(N)}[\mathcal{F}(\zeta)],$$

where $\mathcal{F}()$ is some function that operates on a classifier. We also consider learning algorithms to be symmetric. Throughout this section and in the rest of the article we use the notation in Table I unless stated otherwise.

Table I. Notation Used in the Article

Symbol	Meaning
X	Random vector modeling input
$\mathcal{X}$	Domain of random vector (input space) X
Y	Random variable modeling output
$Y(x)$	Random variable modeling output for input x
$\mathcal{Y}$	Set of class labels (output space)
D	Dataset
(x, y)	Data-point from dataset D
D_t	Training dataset
D_s	Testing dataset
D_i	ith part/fold of D (for cross validation)
$D_t^{i,j}$	Common training part in ith and jth run of cross-validation
N	Size of dataset
N_t	Size of training dataset
N_s	Size of testing dataset
v	Number of folds of cross validation
ζ	Classifier
$\zeta[D]$	Classifier build from dataset D
$GE(\zeta)$	Generalization error of classifier ζ
$HE(\zeta)$	Hold-out-set error of classifier ζ
$CE(\zeta)$	Cross validation error of classifier ζ
$\mathcal{Z}(S)$	The set of classifiers obtained by application of classification algorithm to an i.i.d. set of size S
$D(S)$	Dataset of size S
$E_{\mathcal{Z}(S)}[]$	Expectation with respect to the space of classifiers built on a sample of size S

3.1 Generalization Error

The notion of generalization error is defined with respect to an underlying probability distribution defined over the input-output space and a loss function (error metric). We model this probability space with the random vector X for input and random variable Y for output. When the input is fixed, $Y(x)$ is the random variable that models the output.[2] We assume in this article that the domain $\mathcal{X}$ of X is discrete; all the theory can be extended to continuous essentially by replacing the counting measure with Lebesgue measure and sums with integrals. Whenever the probability and expectation is with respect to this probabilistic space (i.e., (X, Y)) that models the problem, we will not use any index. For other probabilistic spaces, we will specify by an index what is the probability space we refer to. We denote the error metric by $\lambda(a, b)$; in this work we will use only the 0-1 metric that takes value 1 if $a \neq b$ and 0 otherwise. With this, the generalization error of a classifier ζ is

$$\begin{aligned} GE(\zeta) &= E[\lambda(\zeta(X), Y)] \\ &= P[\zeta(X) \neq Y] \\ &= \sum_{x \in \mathcal{X}} P[X = x] P[\zeta(x) \neq Y(x)], \end{aligned} \tag{4}$$

[2]By modeling the output for a given input as a random variable, we allow the output to be randomized, as it might be in most real circumstances.

where we used the fact that for 0-1 loss function, the expectation is the probability that the prediction is erroneous. Notice that the notation using $Y(x)$ is really a conditional on $X=x$. We use this notation since it is intuitive and more compact. The last equation for the generalization error is the most useful in this article, since it decomposes a global measure, namely generalization error, defined over the entire space into micromeasures, one for each input.

By carefully selecting the class of classifiers for which the moment analysis of the generalization error is performed, meaningful and relevant probabilistic statements can be made about the generalization error of a particular classifier from this class. The probability distribution over the classifiers will be based on the randomness of the data used to produce the classifier. To formalize this, let $\mathcal{Z}(N)$ be the class of classifiers built over a dataset of size N with a probability space defined over it. With this, the kth moment around 0 of the generalization error is

$$E_{\mathcal{Z}(N)}[GE(\zeta)^k] = \sum_{\zeta \in \mathcal{Z}(N)} P_{\mathcal{Z}}[\zeta] GE(\zeta)^k.$$

The problem with this definition is that it talks about global characterization of classifiers which can be hard to capture. We rewrite the formulae for the first and second moment in terms of fine-granularity structure of the classifiers.

While deriving these moments, we have to consider double expectations of the form $E_{\mathcal{Z}(N)}[E[\mathcal{F}(x,\zeta)]]$ with $\mathcal{F}(x,\zeta)$ a function that depends both on the input x and the classifier. With this we arrive at the result

$$\begin{aligned} E_{\mathcal{Z}(N)}[E[\mathcal{F}(x,\zeta)]] &= \sum_{\zeta \in \mathcal{Z}(N)} P_{\mathcal{Z}(N)}[\zeta] \sum_{x \in \mathcal{X}} P[X=x] \mathcal{F}(x,\zeta) \\ &= \sum_{x \in \mathcal{X}} P[X=x] \sum_{\zeta \in \mathcal{Z}(N)} P_{\mathcal{Z}(N)}[\zeta] \mathcal{F}(x,\zeta) \\ &= E[E_{\mathcal{Z}(N)}[\mathcal{F}(x,\zeta)]] \end{aligned} \tag{5}$$

that uses the fact that $P[X=x]$ does not depend on a particular ζ and $P_{\mathcal{Z}(N)}[\zeta]$ does not depend on a particular x, even though both quantities depend on the underlying probability distribution.

Using the definition of the moments, Eq. (4) and Eq. (5) we have the following theorem. The proof is in the Appendix. In fact all relatively lengthy proofs have been placed in the Appendix.

THEOREM 1. *The first and second moment of GE are given by*

$$E_{\mathcal{Z}(N)}[GE(\zeta)] = \sum_{x \in \mathcal{X}} P[X=x] \sum_{y \in \mathcal{Y}} P_{\mathcal{Z}(N)}[\zeta(x)=y] P[Y(x) \neq y]$$

and

$$\begin{aligned} E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[GE(\zeta)GE(\zeta')] = \sum_{x \in \mathcal{X}} \sum_{x' \in \mathcal{X}} & P[X=x]P[X=x'] \cdot \\ & \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y'] \cdot \\ & P[Y(x) \neq y] P[Y(x') \neq y']. \end{aligned}$$

In both series of equations we made the transition from a summation over the class of classifiers to a summation over the possible outputs, since the focus changed from the classifier to the prediction of the classifier for a specific input (x is fixed inside the first summation). What this effectively does is allow the computation of moments using only local information (behavior on particular inputs), not global information (behavior on all inputs). This results in speeding the process of computing the moments.

3.2 Application of Theorem 1 to Continuous Spaces

The key aspect in the derivation of Theorem 1 is the use of linearity of expectation. Linearity of expectation holds for both sums as well as integrals. Nowhere in the proof of Theorem 1 are specific properties of the space being discrete used. Thus the results derived in Theorem 1 are applicable to the continuous domain as well. To obtain equivalent formulae in the continuous domain we just need to switch from the counting measure to the Lebesgue measure. With this we have the formulations

$$E_{\mathcal{Z}(N)}[GE(\zeta)] = \int_{x\in\mathcal{X}} P[X=x] \sum_{y\in\mathcal{Y}} P_{\mathcal{Z}(N)}[\zeta(x)=y]P[Y(x)\neq y]dx$$

and

$$E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[GE(\zeta)GE(\zeta')] = \int_{x\in\mathcal{X}}\int_{x'\in\mathcal{X}} P[X=x]P[X=x'] \cdot \sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y'] \cdot P[Y(x)\neq y]P[Y(x')\neq y']dxdx'.$$

In fact, the formulations apply to any sigma finite measure [Doob 1994] defined on the input space. Measure, theoretic notation is unnecessary and can be tedious to follow, and hence for simplicity of notation we derived Theorem 1 assuming counting measure.

3.3 Impact of Theorem 1

The method we introduced before for computing the moments of the generalization error are based on decomposing the moment into contributions of individual input-output pairs. With such a decomposition, not only does the analysis become simpler, but the complexity of the algorithm required is reduced. In particular, the complexity of computing the first moment is proportional to the size of the input-output space and the complexity of estimating probabilities of the form $P_{\mathcal{Z}}[\zeta(x)=y]$. The complexity of the second moment is quadratic in the size of the input-output space and proportional to the complexity of estimating $P_{\mathcal{Z}}[\zeta(x)=y \wedge \zeta(x')=y']$. To see the impact of Theorem 1, we look at the two most natural alternatives for computing the moments using closed-form formulae.

3.3.1 *Moments over Datasets.* In this method we sum over all possible datasets and multiply the generalization error of the classifier built from the dataset with the probability to obtain the dataset as an i.i.d. sample from the

underlying probability distribution. Formally,

$$E_{\mathcal{D}(N)}[GE(\zeta)] = \sum_{D \in \mathcal{D}(N)} P[D] GE(\zeta[D]) \tag{6}$$

where $\mathcal{D}(N)$ is the set of all possible datasets of size N. This is the most natural procedure of computing the moments in closed form. The problem with this method is that the number of terms in the preceding sum is exponential in the size of the input-output space with base N. This is seen in Figure 1 where the number of terms is $O(N^{y m^d})$ (d is the number of attributes, y is the number of classes (output space), and m is the number of values of each attribute).

3.3.2 *Moments over Classifiers.* In this method we sum over all the possible classifiers rather than summing over all possible datasets. The use of this idea is that since the classification algorithms are assumed to be deterministic, multiple datasets map to a single classifier and hence the space of classifiers is comparatively smaller than the space of possible datasets. We have

$$E_{\mathcal{Z}(N)}[GE(\zeta)] = \sum_{\zeta \in \mathcal{Z}(N)} P[\zeta] GE(\zeta) \tag{7}$$

where $\mathcal{Z}(N)$ is the set of all possible classifiers built on a dataset of size N. The complexity in this case reduces from $O(N^{m^d})$ to $O(y^{m^d})$. This is a significant reduction since y will always be much smaller than N. However, the complexity is still exponential in the input space (m^d).

In summary, the advantages of Theorem 1 are: (a) The formulations are exact; (b) each of the terms depends only on the local behavior of the classification algorithm (applicable when input is either discrete or continuous); and (c) the number of terms is reduced to a low degree polynomial (applicable to the discrete case only, since in the continuous case the number of distinct inputs is infinite, in which case we simply integrate). We will use these facts to compute moments of the generalization error for the NBC in Section 4.

3.4 Error Estimation Methods

The exact computation of the generalization error depends on the actual underlying probability distribution, which is unknown, and hence other estimates for the generalization error have been introduced: Hold-Out-Set (HOS), Leave-One-Out (LOO), and v-fold Cross-Validation (CV). In the this subsection we establish relationships between moments of these error metrics and the moments of the generalization error with respect to some distribution over the classifiers. The general setup for the analysis for all these metrics is the following. A dataset D of size N is provided, containing i.i.d. samples coming from the underlying distribution over the input and outputs. The set is further divided and used both to build a classifier and to estimate the generalization error; the particular way this is achieved is slightly different for each error metric. The important question we will ask is how the values of the error metrics relate to the generalization error. In all the developments that follow we will assume that $\zeta[D]$ is the classifier built deterministically from the dataset D.

3.4.1 *Hold-Out-Set (HOS) Error.* The HOS error involves randomly partitioning the dataset D into two parts: D_t, the training dataset of fixed size N_t; and D_s, the test dataset of fixed size N_s. A classifier is built over the training dataset and the generalization error is estimated as the average error over the test dataset. Formally, denoting the random variable that gives the HOS error by HE we have

$$HE = \frac{1}{N_s} \sum_{(x,y) \in D_s} \lambda(\zeta[D_t](x), y), \tag{8}$$

where y is the actual label for the input x.

PROPOSITION 1. *The expected value of HE is given by*

$$E_{D_t(N_t) \times D_s(N_s)}[HE] = E_{D_t(N_t)}[GE(\zeta[D_t])].$$

The proof of the proposition is in the Appendix. We observe from the aforesaid result that the expected value of HE is dependent only on the size of the training set D_t. This result is intuitive since only N_t datapoints are used for building the classifier.

LEMMA 1. *The second moment of HE is given by*

$$E_{D_t(N_t) \times D_s(N_s)}[HE^2] = \frac{1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])] + \frac{N_s - 1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])^2].$$

The proof of this lemma is in the Appendix.

THEOREM 2. *The variance of HE is given by*
$Var_{D_t(N_t) \times D_s(N_s)}(HE) = \frac{1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])] + \frac{N_s - 1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])^2] - E_{D(N_t)}$
$[GE(\zeta[D_t])]^2$.

PROOF. The proof of the theorem immediately follows from the Proposition 1 and Lemma 1 and by using the formula for the variance of a random variable in this case HE

$$Var(HE) = E[HE^2] - E[HE]^2. \qquad \square$$

Unlike the first moment, the variance depends on the sizes of both the training set as well as the test set.

3.4.2 *Multifold Cross-Validation Error.* v-fold cross-validation consists in randomly partitioning the available data into v equi-size parts, then training v classifiers using all data but one chunk, and then testing the performance of the classifier on this chunk. The estimate of the generalization error of the classifier built from the entire data is the average error over the chunks. Using notation in this article and denoting by D_i the ith chunk of the dataset D, the Cross-validation Error (CE) is

$$CE = \frac{1}{v} \sum_{i=1}^{v} HE_i.$$

Notice that we expressed *CE* in terms of *HE*, the HOS error. By substituting the formula for *HE* in Eq. (8) into the previous equation, a direct definition for *CV* is obtained, if desired.

In this case we have a classifier for each chunk not a single classifier for the entire data. We model the selection of N i.i.d. samples that constitute the dataset D and the partitioning into v chunks. With this we have the following proposition.

PROPOSITION 2. *The expected value of CE is given by*

$$E_{D_t\left(\frac{v-1}{v}N\right)\times D_i\left(\frac{N}{v}\right)}[CE] = E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])].$$

PROOF. Using the previous equation, Proposition 1 we get the following result.

$$\begin{aligned} E_{D_t\left(\frac{v-1}{v}N\right)\times D_i\left(\frac{N}{v}\right)}[CE] &= \frac{1}{v}\sum_{i=1}^{v} E_{D_t\left(\frac{v-1}{v}N\right)\times D_i\left(\frac{N}{v}\right)}[HE_i] \\ &= \frac{1}{v}\sum_{i=1}^{v} E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])] \\ &= E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])] \end{aligned}$$

□

This result follows the intuition, since it states that the expected error is the generalization error of a classifier trained on $\frac{v-1}{v}N$ datapoints. Thus, at least on expectation, the cross-validation behaves exactly like the HOS estimate that is trained over $v-1$ chunks.

For *CE* we could compute the second moment around zero using the strategy previously shown and then compute the variance. Here, we compute the variance using the relationship between the variance of the sum and the variances and covariances of individual terms. In this way we can decompose the overall variance of *CE* into the sum of variances of individual estimators and the sum of covariances of pairs of such estimators; this decomposition significantly enhances the understanding of the behavior of *CE*, as we will see in the example in Section 4.

$$Var(CE) = \frac{1}{v^2}\left(\sum_{i=1}^{v} Var(HE_i) + \sum_{i\neq j} Cov(HE_i, HE_j)\right) \tag{9}$$

The quantity $Var(HE_i)$, the variance of the HE on training data of size $\frac{v-1}{v}N$ and test data of size $\frac{1}{v}N$, is computed using the formulae in the previous section. The only things that remain to be computed are the covariances. Since we already computed the expectation of *HE*, to compute the covariance it is enough to compute the quantity $\mathcal{Q} = E[HE_iHE_j]$ (since for any two random variables X, Y we have $Cov(X,Y) = E[XY] - E[X]E[Y]$). D_t^{ij} denotes $D \setminus \{D_i \cup D_j\}$ and let $N_s = \frac{N}{v}$. With this we have the following lemma, whose proof is in the Appendix.

LEMMA 2. *The* $E_{D_t^{ij}\left(\frac{v-2}{v}N\right)\times D_i\left(\frac{N}{v}\right)\times D_j\left(\frac{N}{v}\right)}[HE_iHE_j] = E_{D_t^{ij}\left(\frac{v-2}{v}N\right)}[GE(\zeta[D \setminus D_i])$ $GE(\zeta[D \setminus D_j])]$.

THEOREM 3. *The variance of CE is given by*

$$\begin{aligned} Var_{D_t^{ij}\left(\frac{v-2}{v}N\right)\times D_j\left(\frac{N}{v}\right)}CE = &\frac{1}{N}E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])] \\ &+\frac{N-v}{vN}E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])^2] - \frac{v-2}{v}E_{D_t\left(\frac{v-1}{v}N\right)}[GE(\zeta[D_t])]^2 \\ &+\frac{v-1}{v}E_{D_t^{ij}\left(\frac{v-2}{v}N\right)}[GE(\zeta[D\setminus D_j])GE(\zeta[D\setminus D_i])]. \end{aligned}$$

PROOF. The expression for the covariance is immediate from the previous result and so using Eq. (9) we derive the variance of *CE*. □

It is worth mentioning that Leave-One-Out (LOO) is just a special case of v-fold cross-validation ($v = N$ for leave-one-out). The preceding formulae apply also to LOO and thus no separate analysis is necessary.

With this we have related the first two moments of *HE* and *CE* to that of *GE*. Hence, if we can compute the moments of *GE* we can also compute the moments of *HE* and *CE*, allowing us to study the model as well as the selection measures. In the next couple of sections we thus focus our attention on computing the moments of *GE* efficiently for a particular classification model: NBC.

4. EXAMPLE: NAIVE BAYES CLASSIFIER

The results (i.e., expressions and relationships) we derived in the previous section were applicable to any deterministic classification algorithm. We can thus use these results to study the behavior of the errors for a classification algorithm of our choice.

The classification algorithm we consider in this article is naive Bayes. We first study the naive Bayes for a single input attribute (i.e., for one dimension) and later the generalized version maintaining scalability. As we will see, these moments are too complicated, as mathematical formulae, to interpret. We will plot these moments to gain an understanding of the behavior of the errors under different conditions, thus portraying the usefulness of the proposed method.

4.1 Naive Bayes Classifier Model

In order to compute the moments of the generalization error, moments that we linked in Section 3 with the moments of the hold-out-set error and cross-validation, we first have to select a classifier and specify the construction method. We selected the single input, the naive Bayes classifier, since the analysis is not too complicated but highlights both the method and the difficulties that have to be overcome. As we will see, even this simplified version exhibits interesting behavior. We fix the number of class labels to 2 as well. In the next section, we discuss how the analysis we present here extends to the general NBC.

Given values for any of the inputs, the NBC computes the probability to see any of the class labels as output, under the assumption that the input attributes influence the output attribute independently. The prediction is the class label that has the largest such estimated probability. For the version of

Table II. Contingency Table of Input X

X	y_1	y_2	
x_1	N_{11}	N_{12}	
x_2	N_{21}	N_{22}	
$\vdots$			
x_n	N_{n1}	N_{n2}	
	N_1	N_2	N

Table III. Notation

Symbol	Symantics
p_{c1}	prior of class C_1
p_{c2}	prior of class C_2
$p^h_{\bar{i}j}$	joint probability of being in $h_{\bar{i}},\ C_j$
N_1	r.v. denoting number of datapoints in class C_1
N_2	r.v. denoting number of datapoints in class C_2
$N^h_{\bar{i}j}$	r.v. denoting number of datapoints in $h_{\bar{i}},\ C_j$
$N_{\bar{i}j}$	r.v. denoting number of datapoints in cell $\bar{i},\ C_j$
N	Size of dataset

the naive Bayes classifier we consider here (i.e., a single input), the prediction given input x is

$$\zeta(x) = \text{argmax}_{k\in\{1,2\}} P[Y = y_k]P[X = x|Y = y_k].$$

The probabilities that appear in the formula are estimated using the counts in the contingency table in Table II. Using the fact that $P[Y = y_k]P[X = x|Y = y_k] = P[X = x \wedge Y = y_k]$ and the fact that $P[X = x_i \wedge Y = y_k]$ is $\frac{N_{ik}}{N}$, the prediction of the classifier is

$$\zeta(x_i) = \begin{cases} y_1 & \text{if } N_{i1} \geq N_{i2}) \\ y_2 & \text{if } N_{i1} < N_{i2}) \end{cases}.$$

4.2 Computation of the Moments of GE

Under the already stated data generation model, the moments of the generalization error for the NBC can be computed (see Table III for notation.). We now present three approaches for computing the moments and show that the approach using Theorem 1 is by far the most practical.

Going over datasets. If we calculate the moments by going over all possible datasets, the number of terms in the formulation for the moments is exponential in the number of attribute values with the base of the exponential being the size of the dataset (i.e., $O(N^n)$ terms). This is because each of the cells in Table II can take $O(N)$ values. The formulation of the first moment would be

$$E_{D(N)}[GE(\zeta(D(N)))] = \sum_{N_{11}=0}^{N} \sum_{N_{12}=0}^{N-N_{11}} \cdots \sum_{N_{n1}=0}^{N-(N_{11}+\cdots+N_{(n-1)2})} eP[N_{11}, \ldots, N_{n2}], \quad (10)$$

where e is the corresponding error of the classifier. We see that this formulation can be tedious to deal with. So can we do better? Yes we definitely can and this

spurs from the following observation. For the NBC built on Table II all we care about in the classification process is the relative counts in each of the rows. Thus, if we had to classify a datapoint with attribute value x_i we would classify it into class y_1 if $N_{i1} > N_{i2}$ and vice versa. What this means is that, irrespective of the actual counts of N_{i1} and N_{i2}, as long $N_{i1} > N_{i2}$ the classification algorithm would make the same prediction, namely we would have the same classifier. We can hence switch from going over the space of all possible datasets to going over the space of all possible classifiers, with the advantage of reducing the number of terms.

Going over classifiers. If we find the moments by going over the space of possible classifiers we reduce the number of terms from $O(N^n)$ to $O(2^n)$. This is because there are only two possible relations between the counts in any row ($\geq$ or $<$). The formulation for the first moment would then be

$$E_{Z(N)}[GE(\zeta)] = e_1 P[N_{11} \geq N_{12}, \ldots, N_{n1} \geq N_{n2}] + e_2 P[N_{11} < N_{12}, \ldots, N_{n1} \geq N_{n2}] + \cdots + e_{2^n} P[N_{11} < N_{12}, \ldots, N_{n1} < N_{n2}]$$

where e_1, e_2 to e_{2^n} are the corresponding errors. Though this formulation reduces the complexity significantly, since $N \gg 2$ for any practical scenario, nonetheless the number of terms is still exponential in n. Can we still do better? The answer is yes again. Here is where Theorem 1 gains prominence. To restate, Theorem 1 says that while calculating the first moment we just need to look at particular rows of the table and to calculate the second moment just pairs of rows. This reduces the complexity significantly without compromising on the accuracy, as we will see.

Going over classifiers using Theorem 1. If we use Theorem 1, the number of terms reduces from an exponential in n to small polynomial in n. Thus, the number of terms in finding the first moment is just $O(n)$ and that for the second moment is $O(n^2)$. The formulation for the first moment would then be

$$E_{Z(N)}[GE(\zeta)] = e_1 P[N_{11} \geq N_{12}] + e_2 P[N_{11} < N_{12}] + \cdots + e_{2n} P[N_{n1} < N_{n2}],$$

where e_1, e_2 to e_{2n} are the corresponding errors. They are basically the respective cell probabilities of the multinomial (i.e., e_1 is probability of a datapoint belonging to the cell x_1C_2, e_2 is probability to belong to x_1C_1, and so on). For the second moment we would have joint probabilities, with the expression being the following.

$$E_{Z(N)}[GE(\zeta)^2] = (e_1 + e_3) P[N_{11} \geq N_{12}, N_{21} \geq N_{22}] + (e_2 + e_3) P[N_{11} < N_{12}, N_{21} \geq N_{22}] + \cdots + (e_{2n-2} + e_{2n}) P[N_{(n-1)1} < N_{(n-1)2}, N_{n1} < N_{n2}]$$

We have thus reduced the number of terms from $O(N^n)$ to $O(n^k)$ where k is small and depends on the order of the moment we are interested in. This formulation has another advantage. The complexity of calculating the individual probabilities is also significantly reduced. The probabilities for the first moment can be computed in $O(N^2)$ time and that for the second in $O(N^4)$ time rather than $O(N^{n-1})$ and $O(N^{2n-2})$ time, respectively.

Further optimizations can be done by identifying independence between random variables and expressing them as binomial cdf's, using the incomplete

regularized beta function to calculate these cdf's in essentially constant time. In fact, in future sections we discuss the general NBC model for which the cdf's (probabilities) cannot be computed directly, as it turns out to be too expensive. Therein we propose strategies to efficiently compute these probabilities. The same strategies can be used here to make the computation more scalable.

The situation when ζ and ζ' are the classifiers constructed for two different folds in cross-validation requires special treatment. Without loss of generality, assume that the classifiers are built for the folds 1 and 2. If we let $D_1, \ldots, D_v$ be the partitioning of the datasets into v parts, ζ is constructed using $D_2 \cup D_3 \cup \cdots \cup D_v$ and ζ' is constructed using $D_1 \cup D_3 \cup \cdots \cup D_v$, thus $D_3 \cup \cdots \cup D_v$ training data is common for both. If we denote by N_{jk} the number of datapoints with $X = x_j$ and $Y = y_k$ in this common part and by $N_{jk}^{(1)}$ and $N_{jk}^{(2)}$ the number of such datapoints in D_1 and D_2, respectively, then we have to compute probabilities of the form

$$P\left[\left(N_{i1}^{(2)} + N_{i1} > N_{i2}^{(2)} + N_{i2}\right) \wedge \left(N_{j1}^{(1)} + N_{j1} > N_{j2}^{(1)} + N_{j2}\right)\right].$$

The estimation of this probability using the previous method requires fixing the values of 6 random variables, thus giving an $O(N^6)$ algorithm. Again, further optimizations can be carried out using the strategies given in Sections 7 and 8.

Using the moments of GE, the moments of HE and CE are found using relationships already derived.

5. MULTIDIMENSIONAL NBC

In the previous section we discussed the NBC built on data in a single dimension. As the dimensionality increases, the cost of exactly computing the moments from the formulations also increases. To maintain the scalability of the method, we propose a number of approximation schemes which can be used to estimate the probabilities efficiently and accurately. As we will see, approximating these probabilities leads to highly accurate estimates of the moments for low computational cost, as against directly using Monte Carlo. The approximation schemes we propose assist in efficient computation of the probabilities in arbitrary dimension. As a matter of fact, the approximation schemes are generic enough to be applied to any application where cumulative distribution functions (cdfs) need to be approximated efficiently.

5.1 Calculation of Basic Probabilities

Having come up with the probabilistic formulation for discerning the moments of the generalization error, we are now faced with the daunting task of efficiently calculating the probabilities involved for the NBC when the number of dimensions is more than one. In this section we will mainly discuss single probabilities and the extension to joint probabilities is in Section 6. Let us now briefly preview the kind of probabilities we need to decipher.

With reference to Figure 2, considering the cell $x_1 y_1$ without loss of generality (w.l.o.g.) and by the naive Bayes classifier independence assumption, we need to find the probability of the following condition being true for the two-dimensional

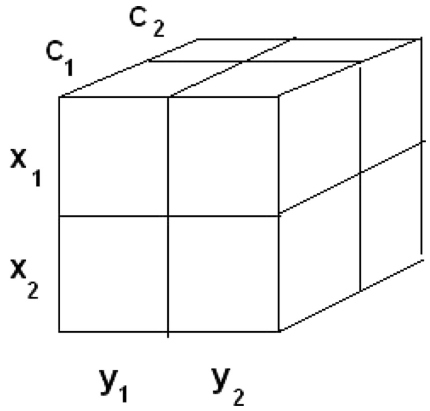

Fig. 2. The scenario when we have two attributes, each having two values with two class labels.

case.

$$\begin{aligned} p_{c1}\frac{p^x_{11}}{p_{c1}}\frac{p^y_{11}}{p_{c1}} &> p_{c2}\frac{p^x_{12}}{p_{c2}}\frac{p^y_{12}}{p_{c2}} \\ \text{namely } p_{c2}p^x_{11}p^y_{11} &> p_{c1}p^x_{12}p^y_{12} \\ \text{namely } N_2N^x_{11}N^y_{11} &> N_1N^x_{12}N^y_{12} \end{aligned}$$

In general for the d-dimensional ($d \geq 2$) case we have to find the following probability

$$P\big[N_2^{(d-1)}N^{x1}_{11}N^{x2}_{11}\ldots N^{xd}_{11} > N_1^{(d-1)}N^{x1}_{12}N^{x2}_{12}\ldots N^{xd}_{12}\big], \tag{11}$$

where the *xi*'s are random variables.

5.2 Direct Calculation

We can find the probability $P[N_2^{(d-1)}N^{x1}_{11}N^{x2}_{11}\ldots N^{xd}_{11} > N_1^{(d-1)}N^{x1}_{12}N^{x2}_{12}\ldots N^{xd}_{12}]$ by summing over all possible assignments of the multinomial random variables involved. For the two-dimensional case shown in Figure 2, we have

$$\begin{aligned} P\big[N_2N^x_{11}N^y_{11} > N_1N^x_{12}N^y_{12}\big] = \sum_{N_{111}}\sum_{N_{121}}\sum_{N_{211}}\sum_{N_{112}}\sum_{N_{122}}\sum_{N_{212}}\sum_{N_{222}} P[N_{111},\ N_{121},\ N_{211},\\ N_{112},\ N_{122},\ N_{212},\ N_{221},\ N_{222}]\cdot I\big[N_2N^x_{11}N^y_{11} > N_1N^x_{12}N^y_{12}\big] \end{aligned}$$

where $N_2 = N_{112}+N_{122}+N_{212}+N_{222}$, $N^x_{11} = N_{111}+N_{121}$, $N^y_{11} = N_{111}+N_{211}$, $N_1 = N - N_2$, $N^x_{12} = N_{112}+N_{122}$, and $I[condition] = 1$ if *condition* is true else $I[condition] = 0$. Each of the summations takes $O(N)$ values and so the worst-case time complexity is $O(N^7)$. We thus observe that for the simple scenario depicted, the time to compute the probabilities is unreasonable even for small-size datasets($N = 100$, say). The number of summations increases linearly with the dimensionality of the space. Hence, the time complexity is exponential in the dimensionality. We thus need to resort to approximations to speed-up the process.

5.3 Approximation Techniques

If all the moments of a random variable are known then we know the Moment Generating Function (MGF) of the random variable and, as a consequence, the probability generating function and hence the precise cdf for any value in the domain of the random variable. If only a subset of the moments are known, then we can at best approximate the MGF and so the cdf.

We need to compute probabilities(cdf's) of the following form $P[X > 0]$, where $X = N_2^{(d-1)} N_{11}^{x1} N_{11}^{x2} \dots N_{11}^{xd} - N_1^{(d-1)} N_{12}^{x1} N_{12}^{x2} \dots N_{12}^{xd}$. Most of the alternative approximation techniques we propose in the subsections that follow, to efficiently compute the aforesaid probabilities(cdf's), are based on the fact that we have knowledge of some finite subset of the moments of the random variable X. We now elucidate a method to obtain these moments.

Derivation of moments. As previously mentioned the most general data generation model for the discrete case is the multinomial distribution. We know the moment generating function for it. A moment generating function generates all the moments of a random variable, uniquely defining its distribution. The MGF of a multivariate distribution is defined as

$$M_R(t) = E(e^{R't}), \tag{12}$$

where R is a q-dimensional random vector, R' is transpose of R, and $t \in R^q$. In our case q is the number of cells in the multinomial.

Taking different order partial derivatives of the moment generating function with respect to the elements of t, and setting these elements to zero, gives us moments of the product of the random variables in the multinomial raised to those orders. Formally,

$$\frac{\partial^{v1+v2+\dots+vq} M_R(t)}{\partial t_1^{v1} \partial t_2^{v2} \dots \partial t_q^{vq}} \Big|_{(t_1=t_2=\dots=t_q=0)} = E\left(R_1^{v1} R_2^{v2} \dots R_q^{vq}\right) \tag{13}$$

where $R' = (R_1, R_2, \dots, R_q)$, $t = (t_1, t_2, \dots, t_q)$, and $v_1, v_2, \dots, v_q$ is the order of the partial derivatives with respect to $t_1, t_2, \dots, t_q$, respectively.

The expressions for these derivatives can be precomputed or computed at runtime using tools such as mathematica [Wolfram-Research 2009]. But how does all of what we have just discussed relate to our problem? Consider the two-dimensional case given in Figure 2. We need to find the probability $P[Z > 0]$ where $Z = N_2 N_{11}^x N_{11}^y - N_1 N_{12}^x N_{12}^y$. The individual terms in the product can be expressed as a sum of certain random variables in the multinomial. Thus, Z can be written as the sum of the product of some of the multinomial random variables. Consider the first term in Z. We have

$$\begin{aligned} N_2 N_{11}^x N_{11}^y &= (N_{112} + N_{122} + N_{212} + N_{222})(N_{111} + N_{121})(N_{111} + N_{211}) \\ &= N_{112} N_{111}^2 + \dots + N_{222} N_{121} N_{211} \end{aligned}$$

and the second term also can be expressed in this form. Thus, Z can be written

as the sum of the products of the multinomial random variables.

$$\begin{aligned} E[Z] &= E[N_2 N_{11}^x N_{11}^y - N_1 N_{12}^x N_{12}^y] \\ &= E[N_2 N_{11}^x N_{11}^y] - E[N_1 N_{12}^x N_{12}^y] \\ &= E[N_{112} N_{111}^2 + \cdots + N_{222} N_{121} N_{211}] \\ &\quad - E[N_{111} N_{112}^2 + \cdots + N_{221} N_{122} N_{212}] \\ &= E[N_{112} N_{111}^2] + \cdots + E[N_{222} N_{121} N_{211}] \\ &\quad - E[N_{111} N_{112}^2] - \cdots - E[N_{221} N_{122} N_{212}] \end{aligned}$$

In the general case $Z = N_2^{d-1} N_{11\ldots1}^{x1} \ldots N_{11\ldots1}^{xd} - N_1^{d-1} N_{11\ldots12}^{x1} \ldots N_{11\ldots12}^{xd}$ where the subscript of N with dots has $d+1$ numbers. The expected value of Z is then given by

$$\begin{aligned} E[Z] =& E[N_{11\ldots12} N_{11\ldots1}^d] + \cdots + E[N_{m_1 m_2 \ldots m_d 2} N_{11\ldots m_d 1} N_{11\ldots m_{d-1}11} \ldots N_{m_1 1\ldots1}] \\ & - E[N_{11\ldots11} N_{11\ldots2}^d] - \cdots - E[N_{m_1 m_2 \ldots m_d 1} N_{11\ldots m_d 2} N_{11\ldots m_{d-1}12} \ldots \ N_{m_1 1\ldots12}] \end{aligned}$$

where m_i denotes the number of attribute values of xi. These expectations can be computed using the technique in the discussion before. Higher moments can also be found in the same vein, since we would only need to find expectations of higher degree polynomials in the random variables, of the multinomial. Similarly, the expressions for the moments in higher dimensions will also include higher degree polynomials.

We now elaborate on various methods that can be used to approximate the cdf of a random variable, given its moments. Note that only the methods which we found to be promising are explained in detail. The other methods, though reasonable, are described briefly.

5.3.1 *Series Approximations (SA).* The Edgeworth or the Gram-Charlier A series [Hall 1992] is used to approximate distributions of random variables whose moments or, more specifically, cumulants are known. These expansions consist in writing the characteristic function of the unknown distribution whose probability density is to be approximated in terms of the characteristic function of another known distribution (usually normal). The density to be found is then recovered by taking the inverse Fourier transform. This method works reasonably well in practice, as can be seen in Levin [1981] and Butler and Sutton [1998]. The major challenge, though, lies in choosing a distribution that will approximate the unknown distribution "well," as the accuracy of the cdf estimate depends on this. The performance of the method may vary significantly on the choice of this distribution, since choosing the normal distribution may not always give satisfactory results. This task of choosing an appropriate distribution is nontrivial.

5.3.2 *Optimization.* We have just seen a method of approximating the cdf using series expansions. Interestingly, this problem can also be framed as an optimization problem, wherein we find upper and lower bounds on the possible values of the cdf by optimizing over the set of all possible distributions having these moments. Since our unknown distribution is an element of this set, its cdf will lie within the bounds computed. This problem is called the classical moment

problem and has been studied in literature [Isii 1963, 1960; Karlin and Shapely 1953]. In fact up to three moments known, there are closed-form solutions for the bounds [Prekopa 1989]. In the material that follows, we present the optimization problem in its primal and dual form. We then explore strategies for solving it, given the fact that the most obvious ones can prove computationally expensive.

Assume that we know m moments of the discrete random variable X, denoted by $\mu_1, \ldots, \mu_m$ where μ_j is the jth moment. The domain of X is given by $U = x_0, x_1, \ldots, x_n$. $P[X = x_r] = p_r$ where $r \in 0, 1, \ldots, n$ and $\sum_r p_r = 1$. We only discuss the maximization version of the problem (i.e., finding the upper bound) since the minimization version (i.e., finding the lower bound) has an analogous description. Thus, in the primal space we have the following formulation.

$$
\begin{aligned}
max \;\; & P[X <= x_r] = \sum_{i=0}^{r} p_i, r \leq n \\
subject\ to: \;\; & \sum_{i=0}^{n} p_i = 1 \\
& \sum_{i=0}^{n} x_i p_i = \mu_1 \\
& \vdots \\
& \sum_{i=0}^{n} x_i^m p_i = \mu_m \\
& p_i \geq 0, \qquad \forall\, i \leq n
\end{aligned}
$$

Solving the aforesaid optimization problem gives us an upper bound on $P[X <= x_r]$.

For our LP problem the dual is

$$
\begin{aligned}
min \;\; & \sum_{k=0}^{m} y_k \mu_k \\
subject\ to: \;\; & \sum_{k=0}^{m} y_k x^k - 1 \geq 0; \qquad \forall\, x \in W, \\
& \sum_{k=0}^{m} y_k x^k \geq 0; \qquad \forall\, x \in U
\end{aligned}
$$

where y_k represent the dual variables and W represents a subset of U over which the cdf is computed. For scalability purposes we will need the dual formulation. We observe that the number of variables is reduced to just $m+1$ in the dual formulation but the number of constraints has increased to the size of the domain of X. We now propose some strategies to solve this optimization problem, discuss their shortcomings, and eventually suggest our preferred strategy.

Using standard linear programming solvers (LP). We have a linear programming problem whose domain is discrete and finite. On careful inspection for our problem we observe that the number of variables in the primal formulation and the number of constraints in the dual increases exponentially in the dimensionality of the space (i.e., the domain of the random variable X). Though the current state-of-the-art LP solvers (using interior point methods) can solve linear optimization problems of the order of thousands of variables and constraints rapidly, our problem can exceed these counts by a significant margin, even for moderate dataset sizes and reasonable dimension, thus becoming computationally intractable. Since standard methods for solving this LP can prove inefficient we investigate other possibilities.

In the next three approaches we extend the domain of the random variable X to include all the integers between the extremeties of the original domain. The current domain is thus a superset of the original domain and so are the possible distributions. Thus, the upper bound calculated in this scenario will be greater than or equal to the upper bound of the original problem. This extension is done to enhance the performance of the next two approaches since it tread jumps the problem of explicitly enumerating the domain of X and is a requirement for the third, as we will soon see.

Gradient descent with binary search (GD). We use gradient descent on the dual to find new values of the vector $\bar{y} = [y_0, \ldots, y_m]$. We descend on the affine objective function starting from an arbitrary point until the violation of any constraint. We report the value as an upper bound on the cdf. The advantage of this method is that it is fast. The pitfall, though, is that the bound can be extremely loose and far from the optimum.

Gradient descent with local topology search (GDTS). Perform gradient descent as mentioned before. Choose a random set of points around the current best solution. Again perform gradient descent on the feasible subset of the chosen points. Choose the best solution and repeat until some reasonable stopping criteria. This works well sometimes in practice, though not always.

Prekopa's algorithm (PA). Prekopa [1989] gave an algorithm for the discrete moment problem. In his algorithm we maintain an $m+1 \times m+1$ matrix called the basis matrix B which needs to have a particular structure to be dual feasible. We iteratively update the columns of this matrix until it becomes primal feasible, resulting in the optimal solution to the optimization problem.[3] The issue with this algorithm is that there is no guarantee with respect to the time required for the algorithm to find this primal feasible basis structure.

In the remaining approaches we further extend the domain of the random variable X to be continuous within the given range. Again, for the same reason described before, the bound is unintrusive.

Sequential quadratic programming (SQP). Sequential quadratic programming is a method for nonlinear optimization. It is known to have local convergence for nonlinear nonconvex problems and will thus globally converge in the

[3]For explanation of the algorithm, read Prekopa [1989].

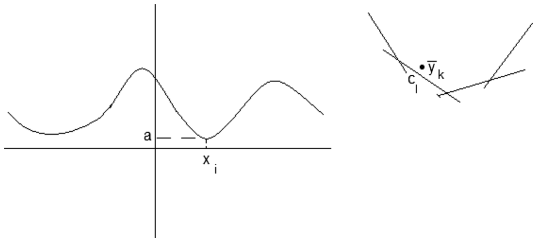

Fig. 3. The current iterate $\bar{y}_k$ just satisfies the constraint c_l and easily satisfies the other constraints. Suppose c_l is $\sum_{j=0}^{m} y_j x_i^j$, where x_i is a value of X, then in the diagram on the left we observe that for the kth iteration $\bar{y} = \bar{y}_k$ the polynomial $\sum_{j=0}^{m} y_j x^j = 0$ has a minimum at $X = x_i$ with the value of the polynomial being a. This is also the value of c_l evaluated at $\bar{y} = \bar{y}_k$.

case of convex optimization. The idea behind SQP is the following. We start with an initial feasible point, say $\bar{y}_{init}$. The original objective function is then approximated by a quadratic function around $\bar{y}_{init}$, which then is the objective for that particular iteration. The constraints are approximated by linear constraints around the same point. The solution of the quadratic program is a direction vector along which the next feasible point should be chosen. The step length can be found using standard line search procedures or more sophisticated merit functions. On deriving the new feasible point, the procedure is repeated until a suitable stopping criterion. Thus at every iteration a quadratic programming problem is solved.

For our specific problem the objective function is affine, thus a quadratic approximation of it yields the original objective function. We have no equality constraints. For the inequality constraints, we use the following idea. The two equations representing the infinite number of linear constraints given in the dual formulation can be perceived as being polynomials in x with coefficients $\bar{y}$. For a particular iteration with iterate($\bar{y}$) known, we find the lowest value that the polynomials take. This value is the value of the most violated (if some constraints are violated)/just satisfied (if no constraint is violated) linear constraint. This is shown in Figure 3. The constraint $c_l = \sum_{j=0}^{m} y_j x_i^j$ is just satisfied. With this in view we arrive at the following formulation of our optimization problem at the kth iteration.

$$min \quad \mu^T d_k$$
$$subject\ to: \sum_{j=0}^{m} y_j^{(k)} x_i^j + \sum_{j=0}^{m} x_i^j d_k \geq 0;\ \bar{y}_k = \left[y_0^{(k)}, \ldots, y_m^{(k)}\right]^4$$

This technique gives a sense of the nonlinear boundary traced out by the constraints. The aforementioned values can be deduced by finding roots of the derivative of the 2 polynomials with respect to x and then finding the minimum of these values evaluated at the real roots of its derivative. The number of roots is bounded by the number of moments; in fact, it is equal to $m-1$. Since this approach does not require the enumeration of each of the linear constraints and the operations described are fast with results being accurate, this turns out

[4] $y_i^{(k)}$ is the value of y_i at the kth iteration.

to be a good option for solving this optimization problem. We carried out the optimization using the Matlab [Wolfram-Research 2009] function *fmincon* and the procedure just illustrated.

Semidefinite programming (*SDP*). A semidefinite programming problem has a linear objective, linear equality constraints, and Linear Matrix Inequality (LMI) constraints. Here is an example formulation

$$\begin{aligned} &min\ \ c^T q \\ &subject\ to:\ \ q_1F_1 + \cdots + q_nF_n + H \preceq 0 \\ &\qquad\qquad Aq = b \end{aligned}$$

where $H,\ F_1, \ldots,\ F_n$ are positive semidefinite matrices, $q \in R^n$, $b \in R^p$, and $A \in R^{p\times n}$. SDP's can be efficiently solved by interior point methods. As it turns out, we can express our semi-infinite LP as an SDP.

Consider the constraint $c_1(x) = \sum_{i=0}^{m} y_i x^i$. The constraint $c_1(x)$ satisfies $c_1(x) \geq 0\ \forall x \in [a, b]$ iff $\exists$ a $m+1 \times m+1$ positive semidefinite matrix S such that

$$\sum_{i+j=2l-1} S(i, j) = 0; \quad l = 1, \ldots, m$$

$$\sum_{k=0}^{l} \sum_{r=k}^{k+m-l} y_r rC_k(m-r)C_{l-k}a^{r-k}b^k = \sum_{i+j=2l} S(i, j); \quad l = 0, \ldots, m.$$

$S \succeq 0$ means S is positive semidefinite.

The proof of this result is given in Bertsimas and Popescu [1998].

We derive the equivalent semidefinite formulation for the second constraint $c_2(x) = \sum_{i=0}^{m} y_i x^i - 1$ to be greater than or equal to zero. To accomplish this, we replace y_0 by $y_0 - 1$ in the previous set of equalities, since $c_2(x) = c_1(x) - 1$. Thus $\forall x \in [a, b]$ we have the following semidefinite formulation for the second constraint.

$$\sum_{i+j=2l-1} S(i, j) = 0; \quad l = 1, \ldots, m$$

$$\sum_{k=1}^{l} \sum_{r=k}^{k+m-l} y_r rC_k(m-r)C_{l-k}a^{r-k}b^k + \sum_{r=1}^{m-l} y_r(m-r)C_l a^r + y_0 - 1 = \sum_{i+j=2l} S(i, j); \quad l = 1, \ldots, m$$

$$\sum_{r=1}^{m} y_r a^r + y_0 - 1 = S(0, 0)$$

$$S \succeq 0$$

Combining the preceding two results we have the following semidefinite program with $O(m^2)$ constraints.

$$
\begin{aligned}
&min \ \sum_{k=0}^{m} y_k \mu_k \\
&subject\ to: \ \sum_{i+j=2l-1} G(i,j) = 0; \quad l = 1, \ldots, m \\
&\sum_{k=1}^{l} \sum_{r=k}^{k+m-l} y_r r C_k (m-r) C_{l-k} a^{r-k} b^k \\
&\quad + \sum_{r=1}^{m-l} y_r (m-r) C_l a^r + y_0 - 1 = \sum_{i+j=2l} G(i,j); \quad l = 1, \ldots, m \\
&\sum_{r=1}^{m} y_r a^r + y_0 - 1 = G(0,0) \\
&\sum_{i+j=2l-1} Z(i,j) = 0; \quad l = 1, \ldots, m \\
&\sum_{k=0}^{l} \sum_{r=k}^{k+m-l} y_r r C_k (m-r) C_{l-k} b^{r-k} c^k = \sum_{i+j=2l} Z(i,j); \quad l = 0, \ldots, m \\
&G \succeq 0, \quad Z \succeq 0
\end{aligned}
$$

Here G and Z are $m+1 \times m+1$ positive semidefinite matrices. The domain of the random variable is $[a, c]$. Solving this semidefinite program yields an upper bound on the cdf $P[X <= b]$, where $a \leq b \leq c$. We used a free online SDP solver [Wu and Boyd 1996] to solve the preceding semidefinite program. Through empirical studies that follow we found this approach to be the best in solving the optimization problem in terms of a balance between speed, reliability, and accuracy.

5.3.3 *Random Sampling Using Formulations (RS).* Random sampling is a sampling technique in which we select a sample from a larger population, wherein each individual is chosen entirely by chance and each member of the population has possibly an unequal chance of being included in the sample. Random sampling reduces the likelihood of bias. It is known that asymptotically the estimates found using random sampling converge to their true values.

For our problem the cdf's can be computed using this sampling procedure.

As we will see in the experiments, only sampling in conjunction with our formulations for the moments makes for an efficient method. If we directly use sampling without using the formulations, we would first need to sample for building a set of classifiers, and then for each classifier built we would need to sample test sets from the distribution. The reason is that the expectation in the moments is with respect to all possible datasets of size N. This process can prove computationally intensive for acquiring accurate estimates of the moments.

5.4 Empirical Comparison of the CDF Computing Methods

Consider the two-dimensional case in Figure 2. We instantiated all the cell probabilities to be equal. We found the probability $P[N_2 N_{11}^x N_{11}^y > N_1 N_{12}^x N_{12}^y]$

Table IV. Empirical Comparison of the CDF Computing Methods in Terms of Execution Time

Method	Dataset Size 10	Dataset Size 100	Dataset Size 1000
Direct	25 hrs	200 centuries	200 billion yrs
SA	0.1 msec	0.1 msec	0.1 msec
LP	3.5 sec	2 min	2:30 hrs
GD	0.13 sec	0.13 sec	0.13 sec
PA	1 sec	25 sec	5 min
GDTS	3.5 sec	3.5 sec	3.5 sec
SQP	3.5 sec	3.5 sec	3.5 sec
SDP	0.1 sec	0.1 sec	0.1 sec
RS_{100}	0.08 sec	0.08 sec	0.1 sec
RS_{1000}	0.65 sec	0.66 sec	0.98 sec
RS_{10000}	6.3 sec	6.5 sec	9.6 sec

RS_n denotes the random sampling procedure using n samples to estimate the probabilities.

by the methods suggested, varying the dataset size from 10–1000 in multiples of 10 and having knowledge of the first six moments of the random variable $X = N_2 N_{11}^x N_{11}^y - N_1 N_{12}^x N_{12}^y$. The actual probability in all the three cases is around 0.5 (actually just less than 0.5). The execution speeds for the various methods are given in Table IV. All the methods were run on the same computer. We used Matlab to implement all the methods, except SDP for which we used a SDP solver [Wu and Boyd 1996]. From the table we see that the SDP and gradient descent methods, are lightning fast. The SQP and gradient descent with topology search methods take a couple of seconds to execute. The thing to notice here is that SDP, SQP, the two gradient descent methods, and the series approximation method are oblivious to the size of dataset with regard to execution time. In terms of accuracy the gradient descent method is sensitive to initialization and the series approximation method to the choice of distribution, as previously stated. A normal distribution gives an estimate of 0.5, which is good in this case since the original distribution is symmetric about the origin. But for finding cdf's near the extremeties of the domain of X the error can be considerable. Since the domain of X is finite, variants of the beta distribution with a change of variable (i.e., shifting and scaling the distribution) can provide better approximation capabilities. The SQP and SDP methods are robust and insensitive to initialization (as long as the initial point is feasible). The bound found by SQP is 0.64–0.34 and that found by SDP is 0.62–0.33. The LP solver also finds a similar bound of 0.62–0.34 but the execution time scales quadratically with the size of the input. On increasing the number of moments to 9, the bounds become tighter and essentially require the same execution time. The SDP, SQP, and LP methods all give a bound of 0.51–0.48. Thus by increasing the number of moments we can get arbitrarily tight bounds. For RS we observe from Table IV and Table V that the method does not scale much in time with the size of dataset, but produces extremely good confidence bounds as the number of samples increases. With 1000 samples we already have pretty tight bounds with time required being just over half a second. Also as previously stated the cdf's can be calculated together rather than independently.

Table V. 95% Confidence Bounds for Random Sampling

Samples	Dataset Size 10	Dataset Size 100	Dataset Size 1000
100	0.7–0.23	0.72–0.26	0.69–0.31
1000	0.54–0.4	0.56–0.42	0.57–0.42
10000	0.5–0.44	0.51–0.47	0.52–0.48

Table VI. Comparison of Methods for Computing the CDF

Method	Accuracy	Speed
Direct	Exact solution	Low
Series Approximation	Variable	High
Standard LP solvers	High	Low
Gradient descent	Low	High
Prekopa's Algorithm	High	Low-Moderate
Gradient descent (topology search)	Moderate	Moderate
Sequential Quadratic Programming	High	Moderate
Semi-definite Programming	High	High
Random Sampling	High	Moderate

Recommendation. The SDP method is the best but SQP and RS are also acceptable.

6. CALCULATION OF CUMULATIVE JOINT PROBABILITIES

Cumulative joint probabilities need to be calculated for computation of higher moments. Using the random sampling method, these probabilities can be computed in similar fashion as the single probabilities shown before. But for the other methods, knowledge of the moments is required. Cumulative joint probabilities are defined over multiple random variables, wherein each random variable satisfies some inequality or equality. In our case, for the second moment we need to find the following kind of cumulative joint probabilities $P[X > 0, Y > 0]$, where X and Y are random variables (overriding their definition in Table I). Since the probability is of an event over two distinct random variables, the previous method of computing moments cannot be directly applied. An important question is: Can we somehow through certain transformations reuse the previous method? Fortunately, the answer is affirmative. The intuition behind the technique we propose is as follows. We find another random variable $Z = f(X, Y)$ (polynomial in X and Y) such that $Z > 0$ iff $X > 0$ and $Y > 0$. Since the two events are equivalent their probabilities are also equal. By taking derivatives of the MGF of the multinomial we get expressions for the moments of polynomials of the multinomial random variables. Thus, $f(X, Y)$ is required to be a polynomial in X and Y. We now discuss the challenges in finding such a function and eventually suggest a solution.

Geometrically, we can consider the random variables X, Y, and Z to denote the three coordinate axes. Then the function $f(X, Y)$ should have a positive value in the first quadrant and negative in the remaining three. If the domains of X and Y were infinite and continuous then this problem is potentially intractable, since the polynomial needs to have a discrete jump along the X and Y axes. Such behavior can be emulated at best approximately by polynomials. In

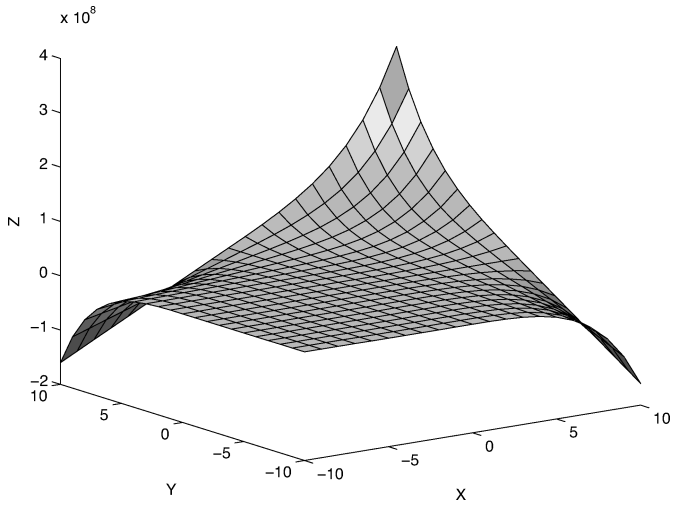

Fig. 4. The plot is of the polynomial $(x+10)^4x^2y+(y+10)^4y^2x-z=0$. We see that it is positive in the first quadrant and nonpositive in the remaining three.

our case, though, the domains of the random variables are finite, discrete, and symmetric about the origin. Therefore, what we care about is that the function behaves as desired only at these finite number of discrete points. One simple solution is to have a circle covering the relevant points in the first quadrant and with appropriate sign the function would be positive for all the points encompassed by it. This works for small domains of X and Y. As the domain size increases the circle intrudes into the other quadrants and no longer satisfies the conditions. Other simple functions such as XY or $X+Y$ or a product of the two also do not work. We now give a function that does work.[5] Consider the domain of X and Y to be integers in the interval [−a,a].[6] Then the polynomial is given by

$$Z=(X+a)^rX^2Y+(Y+a)^rY^2X, \tag{14}$$

where $r=max_b\lfloor\frac{ln[b]}{ln[\frac{a+1}{a-b}]}\rfloor+1$, $1<b<a$, and $b\in\mathbb{N}$. The value of r can be found numerically by finding the corresponding value of b which maximizes that function. Figure 4 depicts the polynomial for $a=10$ where $r=4$. The general shape remains the same for higher values of a.

Recommendation. If the degree of the polynomial is large, use the RS method for convenience, else use SDP or SQP for high accuracy.

With this we have all the ingredients necessary to perform experiments reasonably fast. This is exactly what we report in the next section.

[5]Proof in Appendix.

[6]In our problem X and Y have the same domain.

7. EXPERIMENTS

The experimental section is divided into three parts. In each part we accomplish a particular objective.

(1) In the first part we test if random sampling using our formulations is more accurate than random sampling used to directly estimate the moments. We also explain the reasons for the observed behavior.
(2) In the second part we depict the manner in which the formulations can be used as an exploratory tool in studying learning methods. In particular we report certain interesting observations regarding the model selection measures, namely HOS and CV.
(3) In the last part we build distributions on three UCI datasets and use the formulations for the moments to study the behavior of NBC on these built distributions. Since the moments are over all possible datasets, they provide information about the behavior of NBC, not only on the specific UCI datasets but rather on a class of datasets that are similar to and contain the UCI datasets.

7.1 Part 1: Monte Carlo (MC) vs. Random Sampling Using Formulations (RS)

In the previous section we proposed methods for efficiently and accurately computing the cdf's that are used in the computation of the moments. A natural question is: Why not use simple Monte Carlo to directly estimate the moments rather than derive the formulations and then perform random sampling? In this section, we show that MC fails to provide accurate estimates even in a simple scenario, while RS does an extremely good job for the same amount of computation (i.e., 10000 samples). *Notice that N the training set size and the sample size have different semantics. Since the expectations are over all datasets of size N, the sample size is the number of datasets of size N. More precisely, the sample size is the number of training sets of size N and not the value of N itself.* We first explain the plots and later discuss their implications.

General setup. We fix the total number of attributes to 2. Each attribute has two values with the number of classes also being 2. The five Figures 5, 6, 7, 8, and 9 depict the estimates of MC and RS for different amounts of correlation (measured using chi-square [Connor-Linton 2003]) between the attributes and the class labels, with increasing training set size.

Observations. From the Figure 5 we observe that when the attributes and class labels are uncorrelated, with increasing training set size the estimates of both MC and RS are accurate. Similar qualitative results are seen in Figure 9 when the attributes and class labels are totally correlated. Hence, for extremely low and high correlations both methods produce equally good estimates. The problem arises for the MC method when we move away from these extreme correlations. This is seen in Figures 6, 7, and 8. Both the MC and RS methods perform well initially, but at higher training set sizes (around 10000 and greater) the estimates of the MC method become grossly incorrect, while the RS method still performs exceptionally well. In fact, the

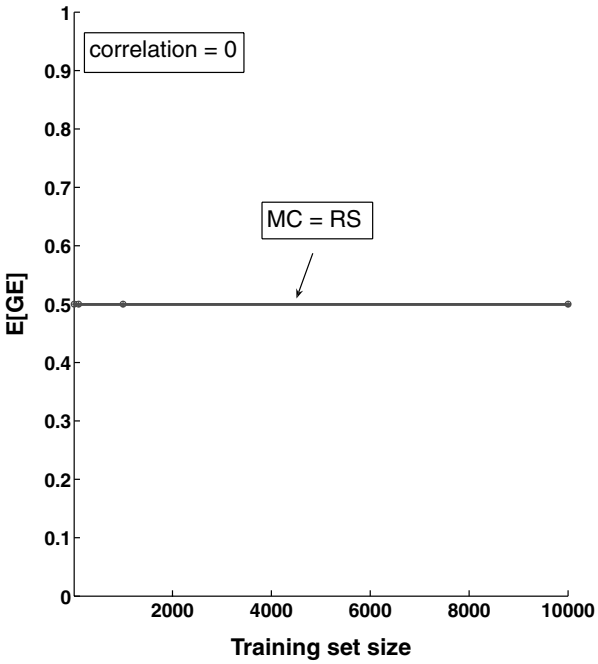

Fig. 5. Estimates of $E_{D(N)}[GE(\zeta)]$ by MC and RS with increasing training set size N. The attributes are uncorrelated with the class labels. True value of $E_{D(N)}[GE(\zeta)]$ is 0.5.

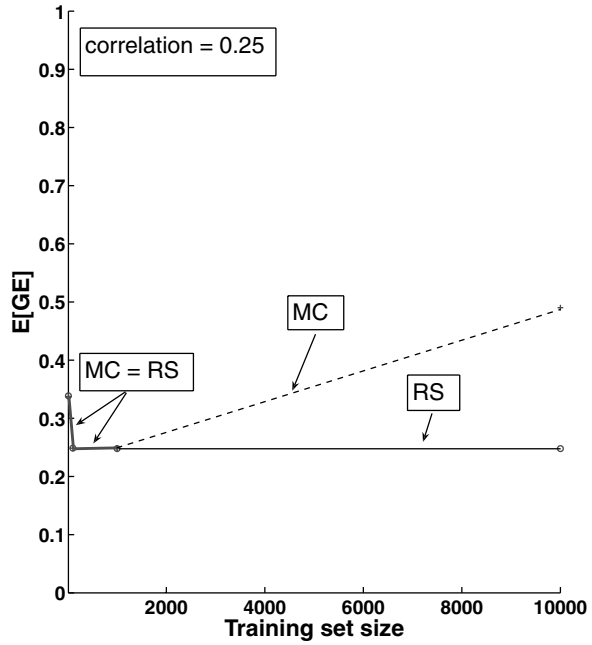

Fig. 6. Estimates of $E_{D(N)}[GE(\zeta)]$ by MC and RS with increasing training set size N. The correlation between the attributes and the class labels is 0.25. True value of $E_{D(N)}[GE(\zeta)]$ is 0.24.

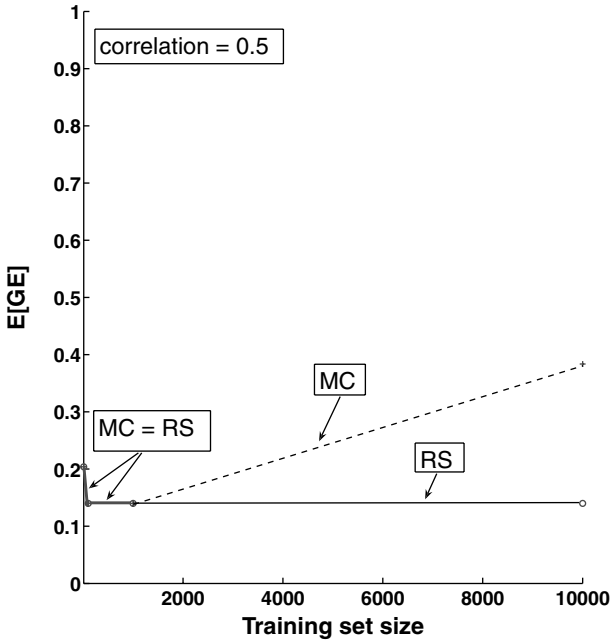

Fig. 7. Estimates of $E_{D(N)}[GE(\zeta)]$ by MC and RS with increasing training set size N. The correlation between the attributes and the class labels is 0.5. True value of $E_{D(N)}[GE(\zeta)]$ is 0.14.

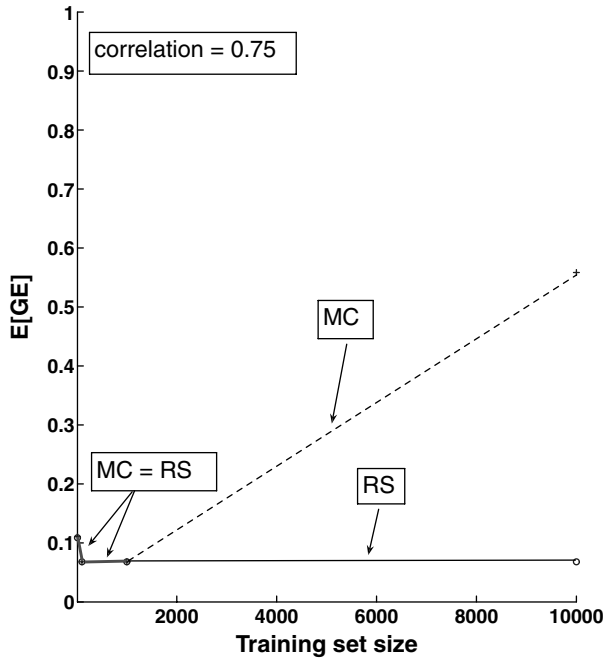

Fig. 8. Estimates of $E_{D(N)}[GE(\zeta)]$ by MC and RS with increasing training set size N. The correlation between the attributes and the class labels is 0.75. True value of $E_{D(N)}[GE(\zeta)]$ is 0.068.

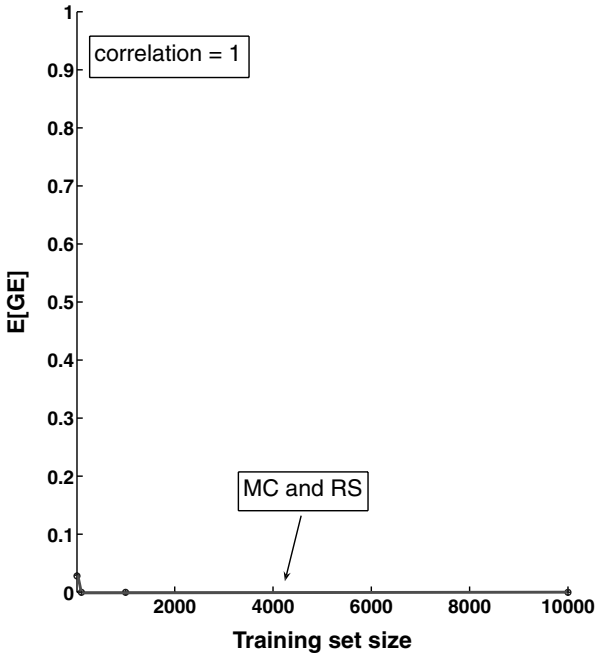

Fig. 9. Estimates of $E_{D(N)}[GE(\zeta)]$ by MC and RS with increasing training set size N. The attributes are totally correlated to the class labels. True value of $E_{D(N)}[GE(\zeta)]$ is 0.

estimates of RS become increasingly accurate with increasing training set size.

Justifications and implications. An explanation of the aforesaid phenomena is as follows: The term $E_{D(N)}[GE(\zeta)]$ denotes the expected GE of all classifiers that are induced by all possible training sets drawn from some distribution. In the continuous case the number of possible training sets of size N is infinite, while in the discrete case it is $O(N^{m-1})$, where m is the total number of cells in the contingency table. As N increases the number of possible training sets increases rapidly even for small values of m. Thus, with increasing N the complexity of $E_{D(N)}[GE(\zeta)]$ also increases. In the experiments we reported previously, the value of m is 8 ($2 \times 2 \times 2$) and with increasing N from 10–10000 the upsurge in the number of possible training sets is steep. Since we fix the amount of computation (i.e., the number of samples), the MC method is unable to get enough samples so as to accurately estimate (except at extreme correlations where almost each sample is representative of the underlying distribution) $E_{D(N)}[GE(\zeta)]$ at higher values of N (e.g., 10000). The MC method estimates are based on samples from a small subspace of the entire sample space. Hence, with increasing number of possible datasets we would have to proportionately increase the number of samples to get good estimates. The RS method is not as affected by increasing training set size. The reason for this is that the complexity (i.e., the parameter space) of the cdf's doesn't scale as much

with increasing N ($O(N^{O(d)})$ where d is the dimension as against $O(N^{m-1})$). Thus, in the case of the RS method, the high accuracy is sustained.

On increasing m, the number of possible training sets increases by a factor of N for each cell added and hence direct MC is intractable to get accurate estimates. The RS method does not scale likewise, since the number of terms (cdf's) is linear in m and the complexity of each term remains the practically unchanged for a fixed dimension. Since computing the first moment is a great challenge for the MC method, computing the second moment, which is over the $D(N)\times D(N)$ space, looks ominous. For the RS and the other suggested methods (e.g., optimization) this is equivalent to finding joint probabilities, which is not that hard a task.

7.2 Part 2: Formulations as an Exploratory Tool

In the previous sections we pointed out how the moments of the generalization error can be computed. In Section 3 we established connections between the moments of the generalization error (GE) and the moments of Hold-out-set Error (HE) and Cross-validation Error (CE). In this section we provide a graphical, simple to interpret representation of these moments for specific cases. While these visual representations do not replace general facts, they greatly improve our understanding, allowing rediscovery of empirically discovered properties of the error metrics and portraying the flexibility of the method to model different scenarios.

General setup. We study the behavior of the moments of *HE*, *CE*, and *GE* in one as well as multiple dimensions. The data distribution we use is a multinomial with a class prior of 0.4. The dataset size is set to 100, namely $N = 100$ for the first two studies and is varied for the third. We set $N = 100$ (and not higher) to clearly observe the effects of an increase in dimensionality on the behavior of these error metrics. The third study varies N, and studies the convergence behavior of these error metrics.

7.2.1 *HOS.* Our first study involves the dependency of the hold-out-set error on the splitting of the data into testing (the hold-out-set) and training. To get insight into the behavior of *HE*, we plotted the expectation in Figures 10 and 13, the variance in Figures 11 and 14, and the sum of the expectation and standard deviation in Figures 12 and 15 for single and multiple dimensions, respectively. As expected, the expectation of *HE* grows as the size of the training dataset reduces. On the other hand, the variance is reduced until the size of test data is 50%, then it increases slightly for the one-dimensional case. The general downwards trend is predictable using intuitive understanding of the naive Bayes classifier, but the fact that the variance has an upwards trend is not. We believe that the behavior on the second part of the graph is due to the fact that the behavior of the classifier becomes unstable as the size of the training dataset is reduced and this competes with the reduction due to the increase in the size of testing data. In higher dimensions the test data size is insufficient even for large test set fractions (as N is only 100) and so any increase in test size

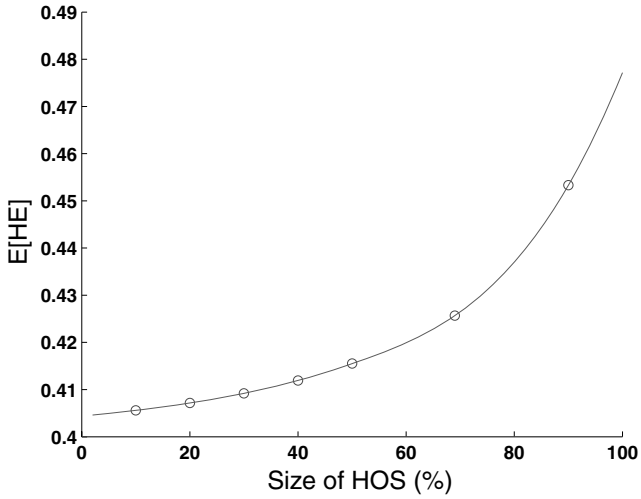

Fig. 10. HE expectation in single dimension.

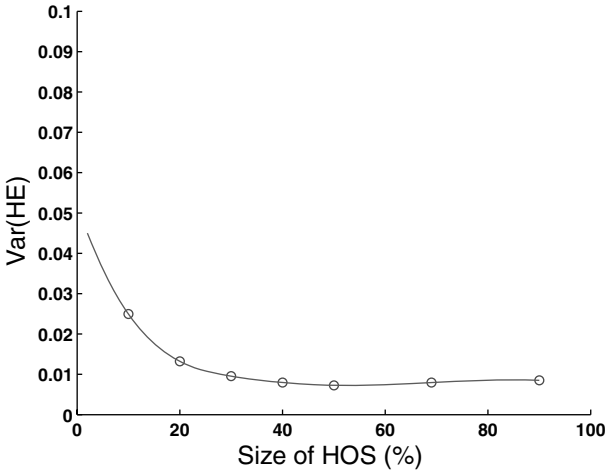

Fig. 11. HE variance in single dimension.

is desirable, leading to reduced variance. Our methodology established this fact exactly, without the doubts associated with intuitively determining the distinct behavior in different dimensions.

From the plots for the sum of the expectation and the standard deviation of *HE*, which indicate the pessimistic expected behavior, a good choice for the size of the test set is 40–50% for this particular instance. This best split depends on the size of the dataset and it is hard to select only based on intuition.

7.2.2 *Cross-Validation.* In our second study we observed the behavior of CV with varying number of folds. Here we observe the similar qualitative results in both lower and higher dimensions. As the number of folds increases, the following trends are observed: (a) The expectation of *CE* reduces (Figures 16,

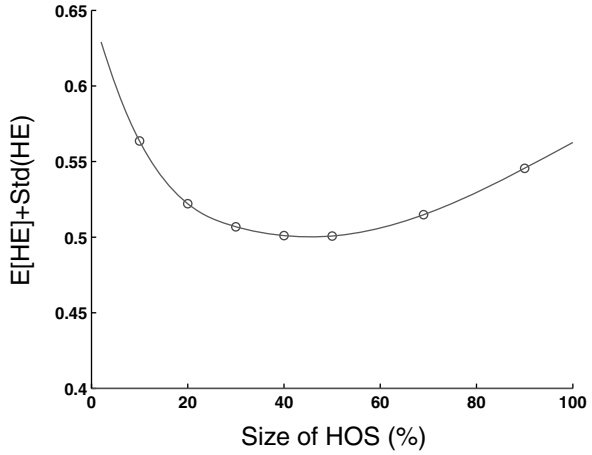

Fig. 12. $E[] + Std()$ of HE in single dimension.

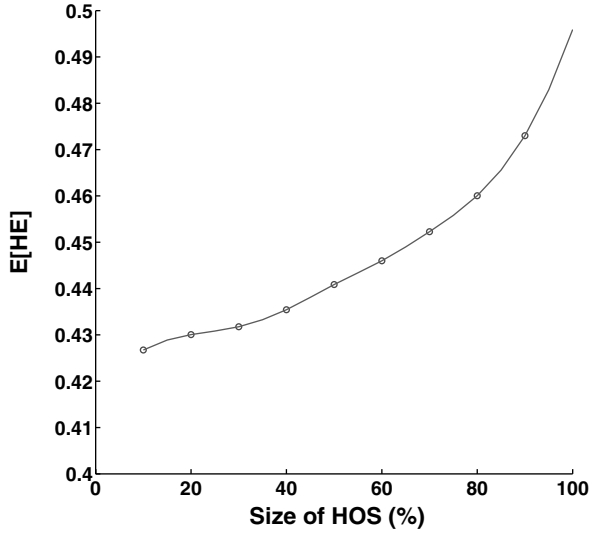

Fig. 13. HE expectation in multiple dimensions.

22) since the size of training data increases; (b) the variance of the classifier for each of the folds increases (Figures 17, 23) since the size of the test data decreases; (c) the covariance between the estimates of different folds decreases first, then increases again (Figures 18, 24). We explain this behavior next and the same trend is observed for the total variance of CE (Figures 19, 25) and the sum of the expectation and the standard deviation of CE (Figures 20, 26). Observe that the minimum of the sum of the expectation and the standard deviation (which indicates the pessimistic expected behavior) is around 10–20 folds, which coincides with the number of folds usually recommended.

A possible explanation for the behavior of the covariance between the estimates of different folds is based on the following two observations. First, when

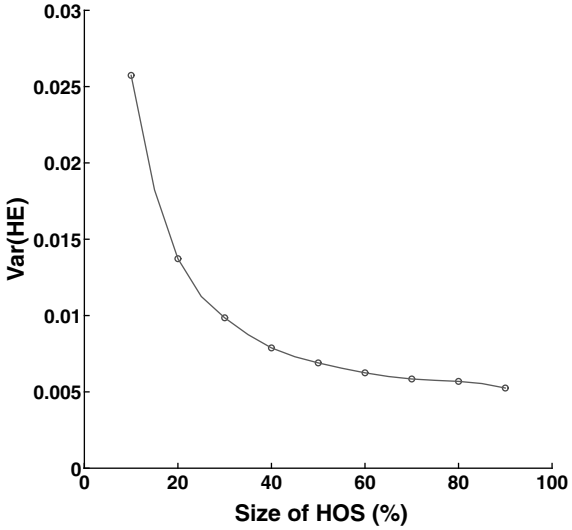

Fig. 14. HE variance in multiple dimensions.

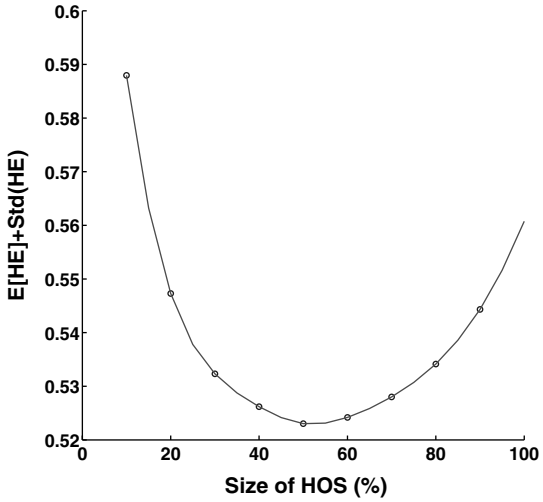

Fig. 15. $E[] + Std()$ of HE in multiple dimensions.

the number of folds is small, the errors of the estimates have large correlations despite the fact that the classifiers are negatively correlated. This happens because almost the entire training dataset of one classifier is the test set for the other, two-fold cross-validation in the extreme. Due to this, though the classifiers built may be similar or different, their errors are strongly positively correlated. Second, for large number of folds (leave-one-out situation in the extreme), there is a huge overlap between the training sets; thus, the classifiers built are almost the same and so the corresponding errors they make are highly correlated again. These two opposing trends produce the U-shaped curve of the

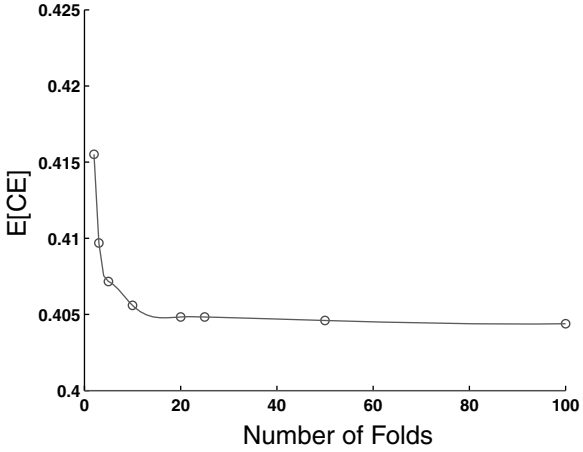

Fig. 16. Expectation of CE.

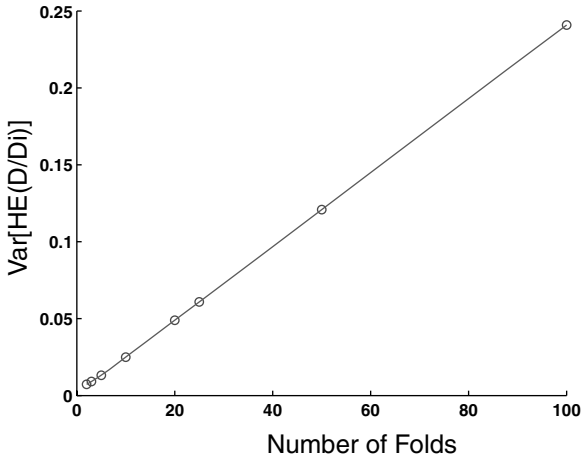

Fig. 17. Individual run variance of CE.

covariance. This has a significant effect on the overall variance and so the variance also has a similar form with the minimum around 10 folds. Predicting this behavior using only intuition, a reasonable number of experiments, or just theory is unlikely, since the interaction between the two trends is not clear.

Such insight is possible only because we are able to observe with high accuracy the factors that affect the behavior of these measures.

7.2.3 *Comparison of GE, HE, and CE.* The purpose of our last study we report was to determine the dependency of the three errors on the size of the dataset, which indicates the convergence behavior and relative merits of hold-out-set and cross-validation. In Figures 21 and 27 we plotted the moments of *GE*, *HE*, and *CE*, the size of the hold-out-set for *HE* was set to 40% and 20 folds for *CE*. As can be observed from the figure, the error of hold-out-set is

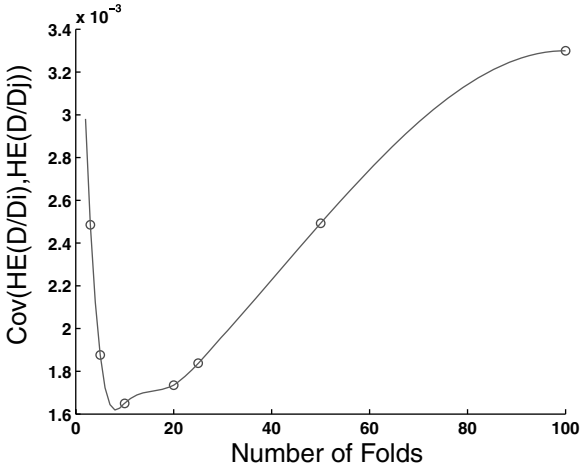

Fig. 18. Pairwise covariances of CE.

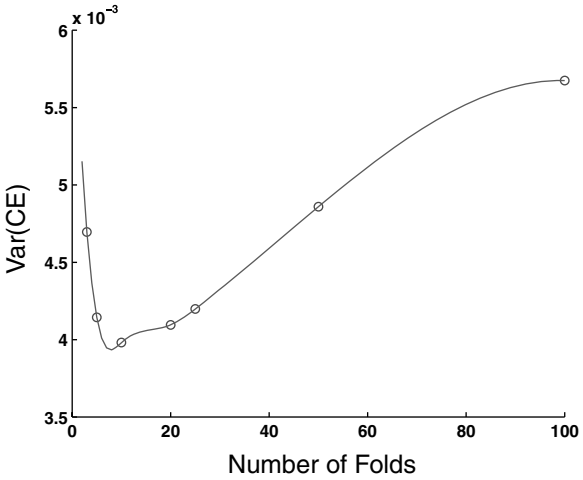

Fig. 19. Total variance of CE.

significantly larger for small datasets. The error of cross-validation is almost on par with the the generalization error. This property of cross-validation to reliably estimate the generalization error is known from empirical studies. But the method can be used to estimate how quickly (at what dataset size) *HE* and *CE* converge to *GE*.

This type of study can be used to observe the nonasymptotic convergence behavior of errors.

7.3 Part 3: Behavior of NBC on Real Data

In the case of real data, we observe the behavior of the moments of GE (i.e., expected value + standard deviation) on distributions built from three UCI datasets, shown in Figure 28. The results reported give an idea of the

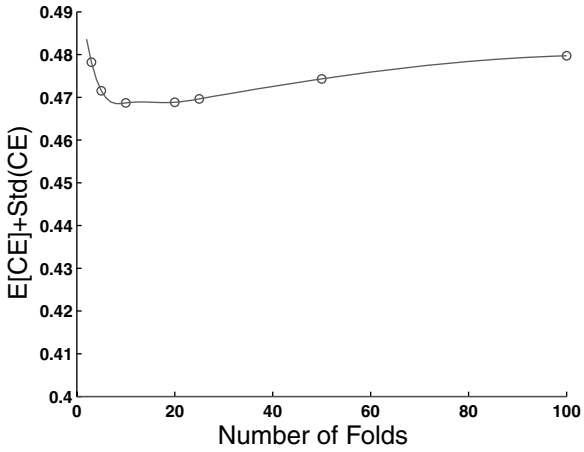

Fig. 20. $E[] + \sqrt{Var()}$ of CE.

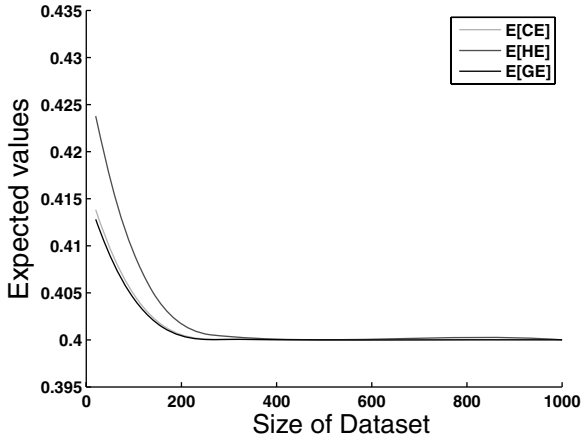

Fig. 21. Convergence behavior.

performance of NBC on datasets that are same as and similar to the original three datasets.

General setup. The datasets (Shuttle Landing Control, Pima Indians, and Balloon) we build distributions on have discrete as well as continuous attributes. We discretize the continuous attributes by splitting them at the mean of the given data. We then form a contingency table representing each of the datasets. The counts in the individual cells divided by the dataset size provide us with empirical estimates for the individual cell probabilities (p_i's). Thus, with the knowledge of N (dataset size) and the individual p_i's we have a multinomial distribution. Using this distribution we observe the behavior of NBC.

Observations and implications. We observe that the NBC has low error on the Shuttle Landing Control dataset, which implies that on datasets from a similar source the NBC has low error with high probability. On the other hand,

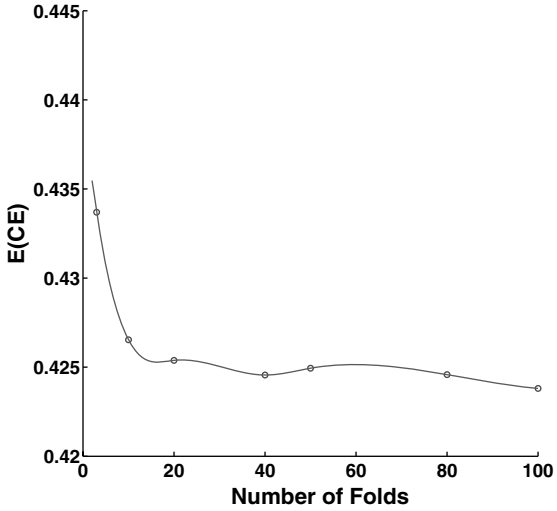

Fig. 22. CE expectation.

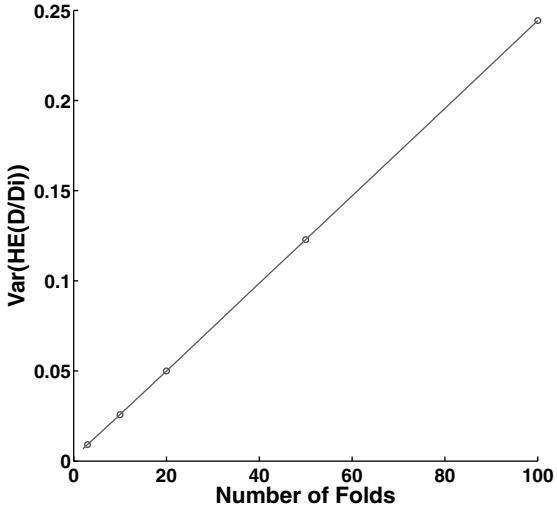

Fig. 23. Individual run variance of CE.

on the Balloon dataset the error is high, which implies that on the same and other similar datasets the performance of NBC is poor. On the Pima Indians dataset the performance of NBC lies somewhere in between the performance for the other two datasets.

Though the metric estimated here is moments of the *GE* taken over all classifers trained on datasets of a particular size while standard validation techniques such as cross-validation aim at estimating the *GE* of a particular classifier trained on a specific dataset, it is interesting to compare the estimates of these two estimators. The ten-fold cross-validation estimates for the three

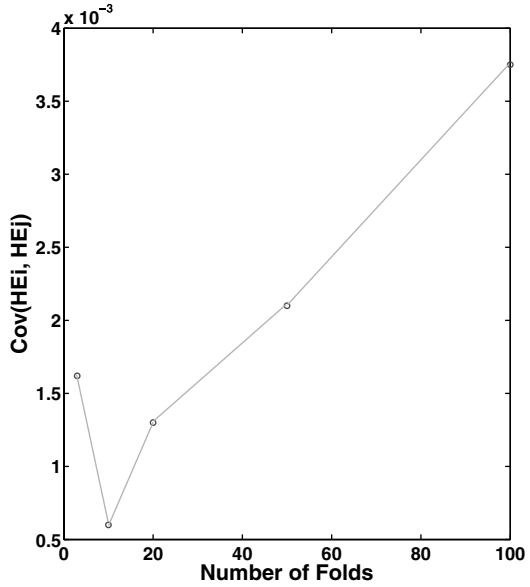

Fig. 24. Pairwise covariances of CE.

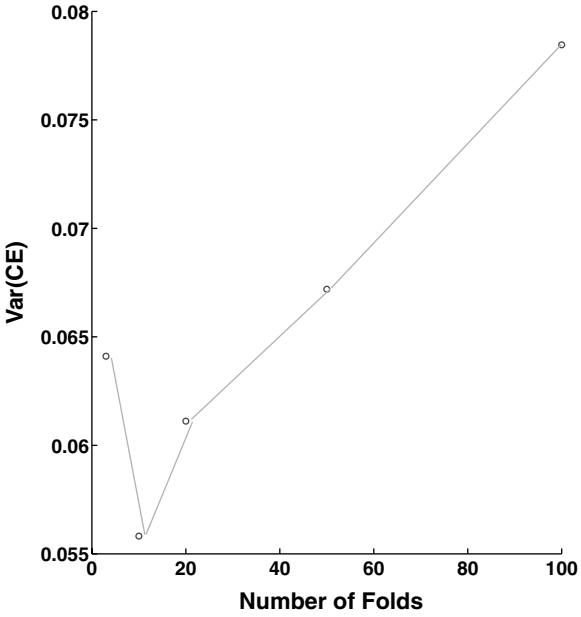

Fig. 25. Total variance of CE.

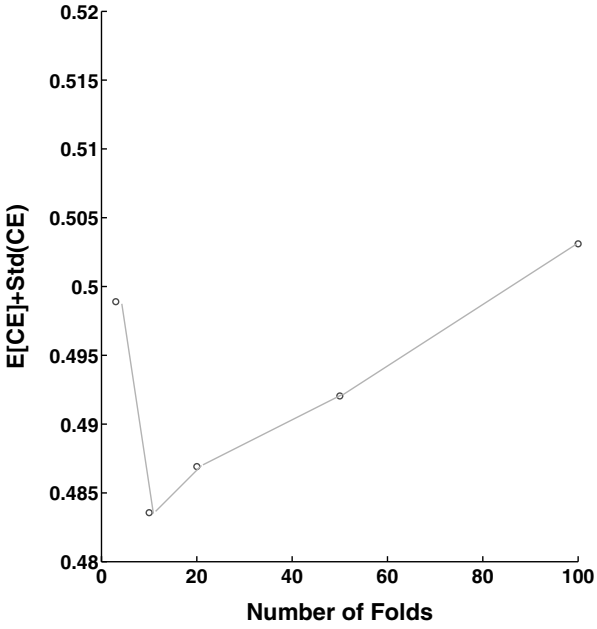

Fig. 26. $E[] + \sqrt{Var()}$ of CE.

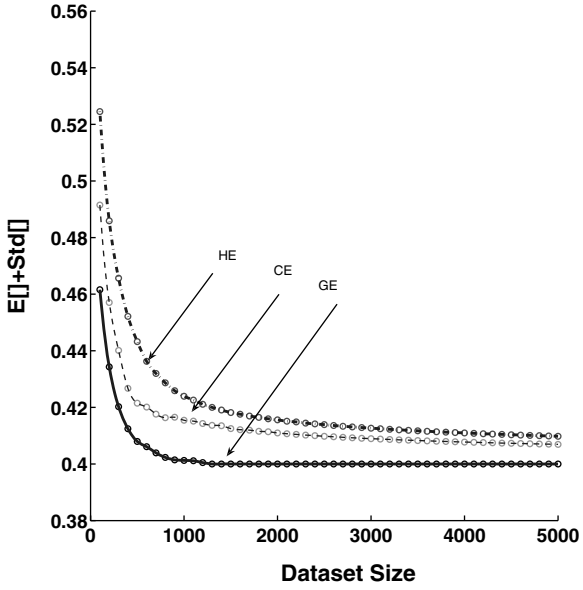

Fig. 27. Convergence behavior.

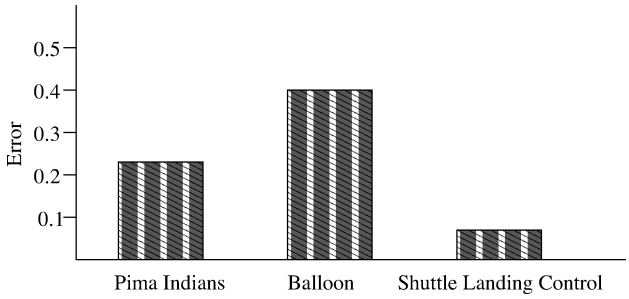

Fig. 28. Behavior of NBC on distributions built on three UCI datasets.

datasets are: 0.26 (Pima Indians), 0.38 (Balloon), and 0.12 (Shuttle Landing Control). In this case the cross-validation errors are qualitatively similar to the estimated moments, which implies that the different classifiers over which the moments are computed are reasonably alike. This, however, may not always occur since the performance on a particular dataset may vary considerably from the performance on other datasets from a similar source.

8. EXTENSION

We have laid down the basic groundwork necessary for characterizing classification models and model selection measures. In particular, we have characterized the NBC model applied to categorical data of arbitrary dimension and with binary class labels. In this section we discuss extensibility of the analysis and the methodology.

The extension of the analysis to NBC with multiple classes is straightforward. We adopt the "winner takes all" policy to classify a datapoint, that is, we classify the datapoint in the class that has the highest corresponding polynomial (of the form $N_2^{(d-1)}N_{11}^{x1}N_{11}^{x2}\ldots N_{11}^{xd}$) value. The approximation techniques employed for speedup are applicable to this scenario, too. In fact, the series approximation and the optimization techniques can be used to bound the cdf in any application where k (some integer) moments of the random variable are known. As mentioned before, the generalized expressions for the moments and the relationships of the moments of GE to the moments of the CE, LE, and HE hold even for the continuous case by switching from counting measure to Lebesgue measure. The challenge in this case, too, is to characterize the probabilities $P_{\mathcal{Z}(N)}[\zeta(x) = y]$ and $P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x) = y \wedge \zeta'(x') = y']$ for the model at hand. The essence in characterizing these probabilities for particular inputs (x) is expressing them as probabilities of the function of the training sample satisfying some condition. The function is determined by the model that is chosen, by prudently looking at the training algorithm in relation with the classifiers it outputs. For example, in the NBC case the function was $N_2^{(d-1)}N_{11}^{x1}N_{11}^{x2}\ldots N_{11}^{xd} - N_1^{(d-1)}N_{12}^{x1}N_{12}^{x2}\ldots N_{12}^{xd}$ for multiple dimensions and $N_1^{x1} - N_2^{x1}$ for a single dimension. The probability of this function being greater than zero is computed from any joint distribution that may specified over the data and the class labels. This observation directs us in characterizing other models of interest.

Analyzing NBC in the continuous case. In the continuous case, the NBC classifies an input based on the training sample class prior and the class conditionals, just as for the discrete case. The class label assigned to an input $z_1, z_2, \ldots, z_d$ is given by

$$\text{class label}(z_1, z_2, \ldots, z_d) = argmax_{C_i} P[C_i] \prod_{j=1}^{d} P[z_j|C_i],$$

where C_i denotes the class i. The NBC estimates the prior and the conditionals from the training sample. The prior is straightforward to estimate. For the conditionals a parametric model is chosen, usually a normal (could be some other choice). The parameters of the normal (mean and variance) can be estimated in *closed form* using parameter estimation methods such as Maximum Likelihood Estimate (MLE). Thus, each of the conditionals can be represented as a function of the sample and so can the prior. The term $P[C_i] \prod_{j=1}^{d} P[z_j|C_i]$ is hence a function of the sample. Since the classification occurs by taking the argmax of this term over all classes that is, by classifying the input in the class for which this term is the greatest (ignoring ties which can easily be accounted for), we arrive at a situation wherein the classification process is expressed as a function of the sample. Densities other than normal may be used; the key is to represent the preceding term as a function of the sample. With the initially chosen joint density over the data and the class labels, which may be different from the density for the conditionals, the probability of a classifier classifying an input into a particular class can be computed.

Extension to other classification algorithms. When we consider other classification models, the basic theme of characterizing them remains unchanged. We discuss possible ways of extending the analysis to some of these other models.

Decision trees. In the case of decision trees, for example, the classification occurs at the leaf nodes by choosing the most numerous class label. If no stopping criterion is enforced, all paths from root to leaf contain all attributes. If stopping criteria are enforced then the paths may contain smaller subsets of the available attributes. Thus, to classify an input into a particular class, we have to check the path in which it lies and that the corresponding class label is most numerous in this path. Formally, the expressions for the probabilities $P_{\mathcal{Z}(N)}[\zeta(x) = y]$ and $P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x) = y \wedge \zeta'(x') = y']$ are given by

$$P_{\mathcal{Z}(N)}[\zeta(x) = y] = \sum_{p} P_{\mathcal{Z}(N)}[ct(path_p\, y) > ct(path_p\, y'), path_p exists, \forall y' \in Y,\ y' \neq y],$$

where p indexes allowed paths by the tree algorithm in classifying input x, Y is the set of class labels, and the function $ct(.)$ counts the number of datapoints in a particular path with the corresponding class label. Thus, $ct(path_p\, y)$ is the count of the number of datapoints lying in $path_p$ having class label y. After the summation, the previously given righthand-side term is the probability that $path_p$ is present in the tree and the number of datapoints in $path_p$ with class label y is greater than the number of datapoints in $path_p$ with any other class label.

The probability that we need to find for the second moment is

$$P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y'] = \sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_p\, y) > ct(path_p\, y''), path_p exists, ct(path_q\, y') > ct(path_q\, y'''), path_q exists, \forall y'', y''' \in Y,\ y \neq y'',\ y' \neq y''']$$

where p and q index all allowed paths by the tree algorithm in classifying input x and x', respectively. The aforesaid two equations are generic in analyzing decision tree algorithms with any reasonable stopping criteria (e.g., for fixed-height trees of height h we would have the condition that a particular set of h attributes is chosen) and classification based on the most numerous class in the leaves. The approximation methods described before can be used to compute these probabilities. It is not difficult to generalize it further when the decision in leaves is some other measure than majority. In that case, we would just include that measure in the probability in place of the inequality.

k-Nearest Neighbor (kNN). In kNN, to compute $P[\zeta(x)=y]$, the probability that majority of the k-nearest neighbors of x belong to class y is found. To compute $P[\zeta(x) = y \wedge \zeta'(x') = y']$ the probability that the majority of the k-nearest neighbors of x belong to class y and majority of the k-nearest neighbors of x' belong to class y' is found.

The scenario wherein x is classified into class y depends on two factors: (1) the kNN's of x and (2) the class label of the majority of these kNN's. The first factor is determined by the distance metric used, which may be dependent or independent of the sample. The second factor is always determined by the sample. The $P_{\mathcal{Z}(N)}[\zeta(x)=y]$ is the probability of all possible ways that input x can be classified into class y, given the joint distribution over the input-output space. This probability for x is calculated by summing (or integrating for the continuous case) the joint probabilities of having a particular set of kNN's and the majority of this set of kNN's has a class label y, over all possible kNN's that the input can have. Formally,

$$P_{\mathcal{Z}(N)}[\zeta(x)=y] = \sum_{q\in Q} P_{\mathcal{Z}(N)}[q, c(q,y) > c(q,y'), \forall y' \in Y,\ y' \neq y],$$

where q is a set of kNN's of the given input and Q is the set containing all possible q. $c(q,t)$ is a function which counts the number of kNN's in q that lie in class t. The $P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y']$ used in the computation of the second moment is calculated by going over kNN's of two inputs rather than one. The expression for this probability is given by

$$P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y'] = \sum_{q\in Q}\sum_{r\in R} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[q, c(q,y) > c(q,y''), r, c(r,y') > c(r,y''') \forall y'', y''' \in Y,\ y \neq y'',\ y' \neq y''']$$

where q and r are sets of kNN's of x and x', respectively. Q and R are sets containing all possible q and r, respectively. Finally, $c(.,.)$ has the same connotation as before. Here again we can use the approximation methods described before to estimate the relevant probabilities.

In this manner, by understanding the training and the functionality of the model, characterizations can be developed. It may not always be possible to do this, but if possible the characterizations developed aid in providing an accurate (if not exact) representation of the behavior of the learning model and the model selection measures, all in a short amount of time (for that accuracy).

9. CONCLUSION

In this section we summarize the major developments in this work and mention a future line of research.

The major contributions in this article are: (a) We developed general relationships between the moments of GE, HE, CE; (b) we reduced the time complexity by shifting the focus from looking at the entire input at one time to looking at only particular inputs; (c) we developed efficient formulations for the moments of the errors for the NBC using the fact in (b); (d) we proposed a plethora of strategies to efficiently compute cdf's required for the moment computation in higher dimensions for the NBC, thus making our method scalable, with a final recommendation of using SDP (or SQP) for single cdf computation and RS for joint cdf computation; and (e) we showed that if RS is used, then RS using our formulations is more efficient and accurate than estimating the moments directly using Monte Carlo. The reason is that the parameter space of the individual terms is much smaller than the entire space over which the moments have to be computed. Finally (f) We plotted our formulations on synthetic distributions as well as those built on real data portraying the manner in which the formulations can be used as an analysis tool. The major challenge in the future is to extend the analysis to other classification models such as k-nearest neighbors, decision trees, neural nets, etc. We have, however, taken an initial step in this direction by laying out the basic framework and characterizing the NBC, which is important in its own right. It is a model which is extensively used in industry, due to its robustness outperforming its more sophisticated counterparts in some real-world applications (e.g., spam filtering in Mozilla Thunderbird and Microsoft Outlook, bio-informatics, etc.).

In conclusion, we proposed a methodology to observe and understand nonasymptotic behavior of errors for the classification model at hand by making as much progress as possible in theory, in view of reducing the computation required for experiments. We then performed experiments to study specific situations that we are interested in. The extent to which this methodology can be used for studying learning methods is a part of our current investigation; however, we feel the method has promise.

ACKNOWLEDGMENTS

We would like to thank A. Banerjee for his comments regarding the introduction. We would also like to thank the editors and the anonymous reviewers

for their constructive comments. We are grateful to P. Gader for his encouragement. This work is supported by the National Science Foundation Grant NSF-CAREER-IIS-0448264.

10. APPENDIX

PROOF OF THEOREM 1.

$$\begin{aligned}
E_{\mathcal{Z}(N)}[GE(\zeta)] &= E_{\mathcal{Z}(N)}[E[\lambda(\zeta(X), Y)]] \\
&= E[E_{\mathcal{Z}(N)}[\lambda(\zeta(X), Y)]] \\
&= \sum_{x\in\mathcal{X}} P[X=x] \sum_{\zeta\in\mathcal{Z}(N)} P_{\mathcal{Z}(N)}[\zeta] P[\zeta(x)\neq Y(x)|\zeta] \\
&= \sum_{x\in\mathcal{X}} P[X=x] \sum_{\zeta\in\mathcal{Z}(N)} P_{\mathcal{Z}(N)}[\zeta] P[\zeta(x)=y, Y(x)\neq y|\zeta] \\
&= \sum_{x\in\mathcal{X}} P[X=x] \sum_{y\in\mathcal{Y}} \sum_{\zeta\in\mathcal{Z}(N)|\zeta(x)=y} P_{\mathcal{Z}(N)}[\zeta] P[\zeta(x)=y, Y(x)\neq y|\zeta] \\
&= \sum_{x\in\mathcal{X}} P[X=x] \sum_{y\in\mathcal{Y}} P_{\mathcal{Z}(N)}[\zeta(x)=y] P[Y(x)\neq y]
\end{aligned}$$

$$\begin{aligned}
&E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[GE(\zeta)GE(\zeta')] \\
&= E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[E[\lambda(\zeta(X), Y)]E[\lambda(\zeta'(X), Y)]] \\
&= \sum_{(\zeta,\zeta')\in\mathcal{Z}(N)\times\mathcal{Z}(N)} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta, \zeta'] \left(\sum_{x\in\mathcal{X}} P[X=x]P[\zeta(x)\neq Y(x)]\right) \\
&\qquad \left(\sum_{x\in\mathcal{X}} P[X=x]P[\zeta'(x)\neq Y(x)]\right) \\
&= \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}} P[X=x]P[X=x'] \sum_{(\zeta,\zeta')\in\mathcal{Z}(N)\times\mathcal{Z}(N)} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta, \zeta'] \\
&\qquad P[\zeta(x)\neq Y(x)]P[\zeta'(x')\neq Y(x')] \\
&= \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}} P[X=x]P[X=x'] \\
&\qquad \sum_{y\in\mathcal{Y}} \sum_{y'\in\mathcal{Y}} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y \wedge \zeta'(x')=y'] \\
&\qquad\qquad P[Y(x)\neq y]P[Y(x')\neq y']
\end{aligned}$$

□

PROOF OF PROPOSITION 1. Using the notation in Table I and realizing that all the datapoints are i.i.d. we derive the following result we have.

$$\begin{aligned}
E_{D_t(N_t)\times D_s(N_s)}[HE] &= E_{D_t(N_t)}\left[E_{D_s(N_s)}\left[\frac{\sum_{(x,y)\in D_s} \lambda(\zeta[D_t](x), y)}{N_s}\right]\right] \\
&= E_{D_t(N_t)}[E_{D_s(N_s)}[P[\zeta[D_t](x)\neq y|D_s]]] \\
&= E_{D_t(N_t)}\left[\sum_{D_s} P[\zeta[D_t](x)\neq y|D_s]P[D_s]\right]
\end{aligned}$$

$$= E_{D_t(N_t)}\left[\sum_{D_s} P[\zeta[D_t](x) \neq y, D_s]\right]$$
$$= E_{D_t(N_t)}[P[\zeta[D_t](x) \neq y]]$$
$$= E_{D_t(N_t)}[GE(\zeta[D_t])]$$

where we used the fact that by going over all values of one random variable we get the probability for the other. □

PROOF OF LEMMA 1. To compute the second moment of HE, from the definition in Eq. (8) from the article we have

$$E_{D_t(N_t)\times D_s(N_s)}[HE^2]$$
$$= \frac{1}{N_s^2} E_{D_t(N_t)\times D_s(N_s)}\left[\sum_{(x,y)\in D_s}\sum_{(x',y')\in D_s} \lambda(\zeta[D_t](x), y)\lambda(\zeta[D_t](x'), y')\right]$$

The expression under the double sum depends on whether (x, y) and (x', y') are the same. When they are the same, we are precisely in the case we derived for $E_{D_t(N_t)\times D_s(N_s)}[HE]$ earlier, except that we have N_s^2 in the denominator. This gives us the following term: $\frac{1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])]$. When they are different, that is, when $(x, y) \neq (x', y')$, then we get

$$\frac{1}{N_s^2} E_{D_t(N_t)\times D_s(N_s)}\left[\sum_{(x,y)\in D_s}\sum_{(x',y')\in D_s\setminus(x,y)} \lambda(\zeta[D_t](x), y)\lambda(\zeta[D_t](x'), y')\right]$$
$$= \frac{N_s - 1}{N_s} E_{D_t(N_t)\times D_s(N_s)}\left[\frac{\sum_{(x,y)\in D_s}\lambda(\zeta[D_t](x), y)}{N_s}\frac{\sum_{(x',y')\in D_s\setminus(x,y)}\lambda(\zeta[D_t](x'), y')}{N_s - 1}\right]$$
$$= \frac{N_s - 1}{N_s} E_{D_t(N_t)\times D_s(N_s)}[P[\zeta[D_t](x) \neq y | (x, y) \in D_s]P[\zeta[D_t](x') \neq y' | (x', y')$$
$$\in D_s \setminus (x, y)]] = \frac{N_s - 1}{N_s} E_{D_t(N_t)}[E_{D_s}[P[\zeta[D_t](x) \neq y | (x, y)$$
$$\in D_s]]E_{D_s}[P[\zeta[D_t](x') \neq y' | (x', y') \in D_s \setminus (x, y)]]]$$
$$= \frac{N_s - 1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])^2]$$

where we used the primary fact that since the samples are i.i.d. any function applied on two distinct inputs is also independent. This the reason why the $E_{D_s}[]$ factorizes.

Putting everything together, and observing that terms inside summations are constants, we have

$$E_{D_t(N_t)\times D_s(N_s)}[HE^2] = \frac{1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])] + \frac{N_s - 1}{N_s} E_{D_t(N_t)}[GE(\zeta[D_t])^2]. \quad \square$$

PROOF OF LEMMA 2.

$$
\begin{aligned}
&E_{D_t^{ij}\left(\frac{v-2}{v}N\right)\times D_i(\frac{N}{v})\times D_j(\frac{N}{v})}[HE_iHE_j] \\
&= E_{D_t^{ij}\left(\frac{v-2}{v}N\right)\times D_i(\frac{N}{v})}\left[HE_iE_{D_j(\frac{N}{v})}\left[\frac{\sum_{(x_j,y_j)\in D_j}\lambda(\zeta[D_t^j](x_j),\,y_j)}{N_s}\right]\right] \\
&= E_{D_t^{ij}\left(\frac{v-2}{v}N\right)}\left[GE(\zeta[D\setminus D_j])E_{D_i(\frac{N}{v})}[HE_i]\right] \\
&= E_{D_t^{ij}\left(\frac{v-2}{v}N\right)}[GE(\zeta[D\setminus D_j])GE(\zeta[D\setminus D_i])]
\end{aligned}
$$

where we used the fact that the datasets D_i and D_j are disjoint and drawn i.i.d. It is important to observe that due to the fact that $D \setminus D_i$ and $D \setminus D_j$ intersect (the intersection is $D \setminus (D_i \cup D_j)$), the two classifiers will neither be independent nor identical. As was the case for the first and second moment of GE, this moment will depend only on the size of the intersection and the sizes of the two sets, since all points are i.i.d. This means that the expression has the same value for any pair $i, j, i \neq j$. □

PROPOSITION 3. *The polynomial* $(x+a)^r x^2 y + (y+a)^r y^2 x > 0$ *iff* $x > 0$ *and* $y > 0$*, where* $x, y \in [-a, -a+1, \ldots, a]$*,* $r = max_b \lfloor \frac{ln[b]}{ln[\frac{a+1}{a-b}]} \rfloor + 1$*,* $a \in \mathbb{N}$*,* $1 < b < a$ *and* $b \in \mathbb{N}$.

PROOF. One direction is trivial. If $x > 0$ and $y > 0$ then definitely the polynomial is greater than zero for any value of r. Now lets prove the other direction, namely if $(x+a)^r x^2 y + (y+a)^r y^2 x > 0$ then $x > 0$ and $y > 0$ where $x, y \in [-a, \ldots, a]$, $r = max_b \lfloor \frac{ln[b]}{ln[\frac{a+1}{a-b}]} \rfloor + 1$, $1 < b < a$ and $b \in \mathbb{N}$. In other words, if $x \leq 0$ or $y \leq 0$ then $(x+a)^r x^2 y + (y+a)^r y^2 x \leq 0$. We prove this result by forming cases.

Case 1. Both x and y are zero.
The value of the polynomial is zero.

Case 2. One of x or y is zero.
The value of the polynomial again is zero, since each of the two terms separated by a sum have xy as a factor.

Case 3. Both x and y are less than zero.
Consider the first term $(x+a)^r x^2 y$. This term is nonpositive since $x+a$ is always non-negative (since $x \in [-a, \ldots, a]$) and x^2 is always positive but y is nonpositive. Analogous argument for the second term and so it, too, is nonpositive. Thus their sum is nonpositive.

Case 4. One of x or y is negative and the other is positive. Assume without loss of generality that x is positive and y is negative.

$$
\begin{aligned}
(x+a)^r x^2 y + (y+a)^r y^2 x &\leq 0 \\
\textit{only if}\quad (x+a)^r x + (y+a)^r y &\geq 0 \\
\textit{only if}\quad r &\geq \frac{ln[\frac{-y}{x}]}{ln[\frac{x+a}{y+a}]}
\end{aligned}
$$

On fixing the value of y the value of x at which the righthand side of the aforesaid achieves maximum is 1 (since we lower the value of x higher the righthand side but x is positive by our assumption and $x \in [-a, \ldots, a]$). Thus we have the previous inequality true only if

$$r \geq \frac{ln[-y]}{ln[\frac{a+1}{y+a}]}.$$

Let $b = -y$ then $1 \leq b \leq a$ since y is negative. Hence, if r satisfies the inequality for all possible allowed values of b, then only will it imply that the polynomial is less than or equal to zero in the specified range. For r to satisfy the inequality for all allowed values of b, it must satisfy the inequality for the value of b that the function is maximum. Also, for $b = 1$ and $b = a$ the righthand side is zero. So the range of b over which we want to find the maximum is $1 < b < a$. With this the minimum value of r that satifies the previous inequality in order for the polynomial to be less than or equal to zero is

$$r = max_b \lfloor \frac{ln[b]}{ln[\frac{a+1}{a-b}]} \rfloor + 1$$

The four cases cover all the possibilities and thus we have shown that if $x \leq 0$ or $y \leq 0$ then $(x+a)^r x^2 y + (y+a)^r y^2 x \leq 0$. Having also shown the other direction we have proved the proposition. □

REFERENCES

BENGIO, Y. AND GRANDVALET, Y. 2003. No unbiased estimator of the variance of k-fold cross validation. *J. Mach. Learn. Res*.

BERTSIMAS, D. AND POPESCU, I. 1998. Optimal inequalities in probability theory: A convex optimization approach. Tech. rep., Department of Mathematics, O.R., Cambridge, Massachusetts, 02139.

BLUM, A., KALAI, A., AND LANGFORD, J. 1999. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Computational Learing Theory*, 203–208.

BOUCHERON, S., BOUSQUET, O., AND LUGOSI, G. 2005. Introduction to statistical learning theory. http://www.kyb.mpg.de/publications/pdfs/pdf2819.pdf.

BOYD, S. AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge.

BREIMAN, L. 1996. Heuristics of instability and stabilization in model selection. *Ann. Statis. 24*, 6, 2350–2383.

BUTLER, R. AND SUTTON, R. 1998. Saddlepoint approximation for multivariate cumulative distribution functions and probability computations in sampling theory and outlier testing. *J. Amer. Statis. Assoc. 93*, 442, 596–604.

CONNOR-LINTON, J. 2003. Chi square tutorial. http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html.

DEVROYE, L., GYÖRFI, L., AND LUGOSI, G. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer.

DOOB, J. 1994. *Measure Theory*. Springer.

ELISSEEFF, A. AND PONTIL, M. 2003. Leave-one-out error and stability of learning algorithms with applications. In *Learning Theory and Practice*. IOS Press.

GOUTTE, C. 1997. Note on free lunches and cross-validation. *Neural Comput. 9*, 6, 1245–1249.

HALL, P. 1992. *The Bootstrap and Edgeworth Expansion*. Springer.

ISII, K. 1960. The extrema of probability determined by generalized moments(i) bounded random variables. *Ann. Inst. Stat. Math 12*, 119–133.

ISII, K. 1963. On the sharpeness of chebyshev-type inequalities. *Ann. Inst. Stat. Math 14*, 185–197.

KARLIN, S. AND SHAPELY, L. 1953. Geometry of moment spaces. *Memoirs Amer. Math. Soc. 12*.

KEARNS, M. AND RON, D. 1997. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In *Computational Learing Theory*, 152–162.

KOHAVI, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1137–1143.

KUTIN, S. AND NIYOGI, P. 2002. Almost-Everywhere algorithmic stability and generalization error. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*. Morgan Kaufmann Publishers, 275–282.

LANGFORD, J. 2005. Filed under: Prediction theory, problems. http://hunch.net/index.php?p=29.

LANGLEY, P. AND SAGE, S. 1999. Tractable average-case analysis of naive Bayesian classifiers. In *Proceedings of the 16th International Conference on Machine Learning*. Morgan Kaufmann, 220–228.

LEVIN, B. 1981. A representation for multinomial cumulative distribution functions. *Ann. Statis. 9*, 5, 1123–1126.

MOORE, A. AND LEE, M. 1994. Efficient algorithms for minimizing cross validation error. In *Proceedings of the International Conference on Machine Learning*, 190–198.

PLUTOWSKI, M. 1996. Survey: Cross-validation in theory and in practice. www.emotivate.com/CvSurvey.doc.

PREKOPA, A. 1989. The discrete moment problem and linear programming. RUTCOR Res. rep.

SHAO, J. 1993. Linear model selection by cross validation. *J. Amer. Statis. Assoc. 88*.

VAPNIK, V. 1998. *Statistical Learning Theory*. Wiley & Sons.

WILLIAMSON, R. 2001. Srm and vc theory (statistical learning theory). http://axiom.anu.edu.au/williams/papers/P151.pdf.

WOLFRAM-RESEARCH. 2009. Mathematica. http://www.wolfram.com/.

WU, S.-P. AND BOYD, S. 1996. Sdpsol: A parser/solver for sdp and maxdet problems with matrix structure. http://www.stanford.edu/ boyd/SDPSOL.html.

ZHU, H. AND ROHWER, R. 1996. No free lunch for cross validation. *Neural Comput. 8*, 7, 1421–1426.

Received May 2008; revised August 2008; accepted September 2008