To appear in Machine Learning: Proceedings of the Thirteenth International Conference, 1996

Bias Plus Variance Decomposition for Zero-One Loss Functions

Ron Kohavi
Data Mining and Visualization
Silicon Graphics, Inc.
2011 N. Shoreline Blvd
Mountain View, CA 94043-1389
ronnyk@sgi.com

David H. Wolpert
The Santa Fe Institute
1399 Hyde Park Rd.
Santa Fe, NM 87501
dhw@santafe.edu

Abstract

We present a bias-variance decomposition of expected misclassification rate, the most commonly used loss function in supervised classification learning. The bias-variance decomposition for quadratic loss functions is well known and serves as an important tool for analyzing learning algorithms, yet no decomposition was offered for the more commonly used zero-one (misclassification) loss functions until the recent work of Kong & Dietterich (1995) and Breiman (1996). Their decomposition suffers from some major shortcomings though (*e.g.*, potentially negative variance), which our decomposition avoids. We show that, in practice, the naive frequency-based estimation of the decomposition terms is by itself biased and show how to correct for this bias. We illustrate the decomposition on various algorithms and datasets from the UCI repository.

1 Introduction

The bias plus variance decomposition (Geman, Bienenstock & Doursat 1992) is a powerful tool from sampling theory statistics for analyzing supervised learning scenarios that have quadratic loss functions. As conventionally formulated, it breaks the expected cost given a fixed target and training set size into the sum of three non-negative quantities:

Intrinsic "target noise" This quantity is a lower bound on the expected cost of any learning algorithm. It is the expected cost of the Bayes-optimal classifier.

Squared "bias" This quantity measures how closely the learning algorithm's average guess (over all possible training sets of the given training set size) matches the target.

"Variance" This quantity measures how much the learning algorithm's guess "bounces around" for the different training sets of the given size.

In addition to the intuitive insight the bias and variance decomposition provides, it has several other useful attributes. Chief among these is the fact that there is often a "bias-variance tradeoff." Often as one modifies some aspect of the learning algorithm, it will have opposite effects on the bias and the variance. For example, usually as one increases the number of degrees of freedom in the algorithm, the bias shrinks but the variance increases. The optimal number of degrees of freedom (as far as expected loss is concerned) is the number of degrees of freedom that optimizes this trade-off between bias and variance.

For classification, the quadratic loss function is often inappropriate because the class labels are not numeric. In practice, an overwhelming majority of researchers in the Machine Learning community instead use expected misclassification rate, which is equivalent to the zero-one loss. Kong & Dietterich (1995) and Dietterich & Kong (1995) recently proposed a bias-variance decomposition for zero-one loss functions, but their proposal suffers from some major problems, such as the possibility of negative variance, and only allowing the values zero or one as the bias for a given test point.

In this paper, we provide an alternative zero-one loss decomposition that does not suffer from these problems and that obeys the desiderata that bias and variance should obey, as discussed in Wolpert (submitted). This paper is expository, being primarily concerned with bringing the zero-one loss bias-variance decomposition to the attention of the Machine Learning community.

After presenting our decomposition, we describe a set of experiments that illustrate the effects of bias and variance for some common induction algorithms. We also discuss a practical problem with estimating the quantities in the decomposition using the naive approach of frequency counts; the frequency-count estimators are biased in a way that depends on the training set size. We show how to correct the estimators so that they are unbiased.

2 Definitions

We use the following notation.

2.1 The underlying spaces

Let $\mathbf{X}$ and $\mathbf{Y}$ be the input and output spaces respectively, with cardinalities $|X|$ and $|Y|$ and elements $\mathbf{x}$ and $\mathbf{y}$, respectively. To maintain consistency with planned extensions of this paper, we assume that both X and Y are countable. However this assumption is not needed for this paper, provided all sums are replaced by integrals. In classification problems, Y is usually a small finite set.

The "target" $\mathbf{f}$ is a conditional probability distribution $P(Y_F = y_F \mid x)$, where $\mathbf{Y_F}$ is a Y-valued random variable. Unless explicitly stated otherwise, we assume that the target is fixed. As an example, if the target is a noise-free function from X to Y, for any fixed x we have $P(Y_F = y_F \mid x) = 1$ for one value of y_F, and 0 for all others.

The "hypothesis" $\mathbf{h}$ generated by a learning algorithm is a similar distribution $P(Y_H = y_H \mid x)$, where $\mathbf{Y_H}$ is a Y-valued random variable. As an example, if the hypothesis is a single-valued function from X to Y, as it is for many classifiers (*e.g.*, decision trees, nearest-neighbors), then $P(Y_H = y_H \mid x) = 1$ for one value of y_H, and 0 for all others.

We will drop the explicitly delineated random variables from the probabilities when the context is clear. For example, $P(y_H)$ will be used instead of $P(Y_H = y_H)$.

Proposition 1 *Y_F and Y_H are conditionally independent given f and a test point x.*

Proof: $P(y_F, y_H \mid f, x) = P(y_F \mid y_H, f, x)P(y_H \mid f, x) = P(y_F \mid f, x)P(y_H \mid f, x)$.
The last equality is true because (by definition) y_F depends only on the target f and the test point x. ▮

The training set $\mathbf{d}$ is a set of $\mathbf{m}$ pairs of x–y values. We do not make any assumptions about the distribution of pairs. In particular, our mathematical results do not require them to be generated in an i.i.d. (independently and identically distributed) manner, as commonly assumed.

To assign a penalty to a pair of values y_F and y_H, we use the **loss** function $\ell : Y \times Y \to \mathbb{R}$. In this paper we consider the zero-one loss function defined as

$$\ell(y_F, y_H) = 1 - \delta(y_F, y_H),$$

where $\delta(y_F, y_H) = 1$ if $y_F = y_H$ and 0 otherwise.

The **cost**, C, is a real-valued random variable defined as the loss over the random variables Y_F and Y_H. So the expected cost is

$$E(C) = \sum_{y_H, y_F} \ell(y_H, y_F) P(y_H, y_F) \ .$$

For zero-one loss, the cost is usually referred to as **misclassification rate** and is derived as follows:

$$\begin{aligned} E(C) &= \sum_{y_H, y_F} [1 - \delta(y_H, y_F)] \, P(y_H, y_F) \\ &= 1 - \sum_{y \in Y} P(Y_H = Y_F = y) \ . \end{aligned} \quad (1)$$

The notation used here is a simplified version of the extended Bayesian formalism (EBF) described in Wolpert (1994). In particular, the results of this paper do not depend on how the X-values in the test set are determined, so there is no need to define a random variable for those X-values as is done in the full EBF.

3 Bias Plus Variance for Zero-One Loss

We now show how to decompose the expected cost into its components and then provide geometric views of this decomposition, in particular relating it to quadratic loss in Euclidean spaces.

3.1 The Decomposition

We present the general result involving the expected zero-one loss, $E(C)$, where the (implicit) conditioning event is arbitrary. Then we specialize to the standard conditioning used in conventional sampling theory statistics: a single test point, target, and training set size.

$$E(C) = 1 - \sum_{y \in Y} P(Y_H = Y_F = y) \quad \text{(From equation 1)}$$

$$= \sum_{y \in Y} -P(Y_H = Y_F = y) + \sum_{y \in Y} P(Y_H = y) P(Y_F = y) +$$

$$\sum_{y \in Y} \left[-P(Y_H = y) P(Y_F = y) + \frac{1}{2} P(Y_F = y)^2 + \frac{1}{2} P(Y_H = y)^2 \right] +$$

$$\left[\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(Y_H = y)^2 \right] + \left[\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(Y_F = y)^2 \right].$$

Rearranging the terms, we have $E(C) =$

$$\begin{aligned}
&\sum_{y \in Y} [P(Y_H = y) P(Y_F = y) \\
&\qquad - P(Y_F = Y_H = y)] + && (\text{“covariance”}) \\
&\frac{1}{2} \sum_{y \in Y} (P(Y_F = y) - P(Y_H = y))^2 + && (\text{“bias}^2\text{”}) \\
&\frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_H = y)^2 \right) + && (\text{“variance”}) \\
&\frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_F = y)^2 \right) . && (\text{“}\sigma^2\text{”}) \quad (2)
\end{aligned}$$

In this paper we are interested in $E(C \mid f, m)$, the expected cost where the target is fixed and one averages over training sets of size m. One way to evaluate this quantity is to write it as $\sum_x P(x) E(C \mid f, m, x)$ and then use Equation 2 to get $E(C \mid f, m, x)$. By Proposition 1, y_H and y_F are independent when one conditions on f and x, hence the “covariance” term vanishes. So

$$E(C) = \sum_x P(x) \left(\sigma_x^2 + \text{bias}_x^2 + \text{variance}_x \right) \qquad (3)$$

where

$$\begin{aligned}
\text{bias}_x^2 &\equiv \frac{1}{2} \sum_{y \in Y} [P(Y_F = y \mid x) - P(Y_H = y \mid x)]^2 \\
\text{variance}_x &\equiv \frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_H = y \mid x)^2 \right) \\
\sigma_x^2 &\equiv \frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_F = y \mid x)^2 \right) .
\end{aligned}$$

(To simplify the exposition, the f and m in the conditioning events are still implicit even though x needs to be explicit.) To better understand these formulas, note that $P(Y_F = y \mid x)$ is the probability (after any noise is taken into account) that the fixed target takes on the value y at point x. To understand the quantity $P(Y_H = y \mid x)$, one must write it in full as

$$\begin{aligned}
&P(Y_H = y \mid f, m, x) = \\
&\quad = \sum_d P(d \mid f, m, x) P(Y_H = y \mid d, f, m, x) \\
&\quad = \sum_d P(d \mid f, m) P(Y_H = y \mid d, x) . \qquad (4)
\end{aligned}$$

In this expression, $P(d \mid f, m)$ is the probability of generating training set d from the target f, and $P(Y_H = y \mid d, x)$ is the probability that the learning algorithm makes guess y for point x in response to the training set d. So $P(Y_H = y \mid x)$ is the average (over training sets generated from f) Y value guessed by the learning algorithm for point x.

Note that while the quadratic loss decomposition involves quadratic terms, and the log loss decomposition involves logarithmic terms (Wolpert submitted), our zero-one loss decomposition does not involve Kronecker delta terms, but rather involves quadratic terms.

Our definitions of “bias^2,” “variance,” and “noise” obey some appropriate desiderata, including:

1. The “bias^2” term measures the squared difference between the target's average output and the algorithm's average output. It is a real-valued nonnegative quantity and equals zero only if $P(Y_F = y \mid x) = P(Y_H = y \mid x)$ for all x and y. These properties are shared by bias^2 for quadratic loss.
2. The variance term measures the “variability” (over Y_H) of $P(Y_H \mid x)$. It is a real-valued nonnegative quantity and equals zero for an algorithm that always makes the same guess regardless of the training set (*e.g.*, the Bayes optimal classifier). As the algorithm becomes more sensitive to changes in the training set, the variance increases. Moreover, given a distribution over training sets, the variance only measures the sensitivity of the learning algorithm to changes in the training set and is independent of the underlying target. This property is shared by variance for quadratic loss.
3. The noise measures the “variance” of the target in that the definitions of variance and noise are identical except for the interchange of Y_F and Y_H. In addition, the noise is independent of the learning algorithm. This property is shared by noise for quadratic loss.

In contrast to our definition of bias^2, the definitions of bias (for a fixed target and a given instance) suggested in Kong & Dietterich (1995), Dietterich & Kong (1995), and Breiman (1996) are only two-valued for binary classification; they cannot quantify subtler levels of mismatch between a learning algorithm and a target. However, their decompositions have the advantage that their bias is zero for the Bayes optimal classifier, while ours may not be.

The major distinction between the decompositions arises in the variance term. All of the desiderata for variance listed above are violated by the decompositions proposed by Kong & Dietterich (1995), Dietterich & Kong (1995), and Breiman (1996). Specifically, in their definitions the variance can be negative and is not minimized by a classifier that ignores the training set. Kong & Dietterich (1995) note this shortcoming explicitly. The following examples illustrates the phenomenon for the decomposition suggested by Breiman (1996). Assume a noise free target with 51% heads and 49% tails. Consider an x for which the target has the value tails; the average error of a majority classifier will be slightly above 50%, yet the probability of error for the "aggregate" majority classifier will be one. This causes the variance to be negative.

Another advantage of our decomposition is that its terms are a continuous function of the target. An infinitesimal change in the target, which changes the class most commonly predicted by the learning algorithm for a given x, will not cause a large change in our bias, variance, or noise terms. In contrast, the other definitions of bias and variance do not share this property.

3.2 The Bias-Variance Decomposition in Vector Form

We can rewrite Equation 2 in vector notation that may give a better geometrical interpretation to the decomposition. Define $\vec{\mathbf{F}} \equiv P(y_F)$, $\vec{\mathbf{H}} \equiv P(y_H)$, and $\vec{\mathbf{FH}} \equiv P(y_F, y_H)$. $\vec{F}$ and $\vec{H}$ are vectors in $\mathbb{R}^{|Y|}$ with their components indexed by Y values; $\vec{FH}$ is a matrix in $\mathbb{R}^{|Y|} \times \mathbb{R}^{|Y|}$. If we denote dot-products by "$\cdot$" and the dot-product of a vector with itself by squaring it, then

$$\text{covariance} = \text{Tr}(\vec{FH}) - \vec{F} \cdot \vec{H} \quad \text{(Tr is the trace)},$$

$$\text{bias}^2 = \tfrac{1}{2}\left(\vec{F} - \vec{H}\right)^2, \text{variance} = \tfrac{1}{2}\left(1 - \vec{H}^2\right), \text{ and}$$

$$\sigma^2 = \tfrac{1}{2}\left(1 - \vec{F}^2\right) .$$

3.3 Relation to the Quadratic Decomposition

To relate the zero-one loss decomposition to the more familiar quadratic loss decomposition let $\vec{Y_F}$ be an $\mathbb{R}^{|Y|}$-valued random variable restricted so that exactly one of its components equals 1 and all others equal 0; that single 1-valued component is the one with index y_F. So $\vec{Y_F}$ is Y_F re-expressed as a vector on the unit hypercube. Define $\vec{Y_H}$ similarly in terms of Y_H.

Since $\vec{Y_F}$ and $\vec{Y_H}$ are real-valued, we can define quadratic loss over them. In particular, recalling that squaring a vector is taking its dot product with itself, we have

$$(\vec{y_F} - \vec{y_H})^2 = \begin{cases} 2 & \text{if } y_F \neq y_H \\ 0 & \text{if } y_F = y_H \end{cases}$$

Accordingly,

$$\begin{aligned} E\left[\left(\vec{Y_F} - \vec{Y_H}\right)^2 \mid f, m, x\right] &= 2\,P(y_H \neq y_F \mid f, m, x) \\ &= 2\,E(C_{0\text{-}1 \text{ loss}} \mid f, m, x) \ . \end{aligned}$$

So by transforming the Y to a vector of indicator variables, we see that the expected cost for zero-one loss is one half the associated quadratic loss.

4 Experimental Methodology

We begin with a description of our experimental methodology, and then discuss a problem with the naive estimation of the terms in our decomposition by using frequency counts.

4.1 Our frequency counts experiments

To investigate the behavior of the terms in our decomposition, we ran a set of experiments on UCI repository (Murphy & Aha 1996). In each of those experiments, for a given dataset and a given learning algorithm, we estimated (the x-average of) bias^2, variance, intrinsic noise, and overall error as follows.

1. We randomly divided each dataset into two parts, **D** and **E**. D was used as if it were the "world" from which we sample training sets, while E was used to evaluate the terms in the decomposition. This idea is similar to the "bootstrap world" idea in Efron & Tibshirani (1993, Chapter 8).
2. We generate **N** training sets from D, each generated using uniform random sampling without replacement. (Note that our decomposition does

not require the training sets to be sampled in any specific manner (Section 2).) To get training sets of size $\mathbf{m}$, we chose D to be of size $2m$. This allows for $\binom{2m}{m}$ different possible training sets, enough to guarantee that we will not get many duplicate training sets in our set of N training sets, even for small values of m.

3. We ran the learning algorithm on each of the training sets and estimated the terms in Equation 3 and 4 using the generated classifier for each point x in the evaluation set E. (Equation 4 was used to estimate $p(y_H|f,m,x)$.) At first, all these terms were estimated using frequency counts.

Figure 1 (left) shows the estimate for bias^2 for different values of N when ID3 (Quinlan 1986) was executed on three datasets from the UCI repository. It is clear that our estimate of bias^2 using frequency counts shrinks as we increase N. Since infinite N gives the correct value of bias^2, this means that for any finite N the bias^2 estimate is always itself biased upwards. The variance term exhibits the opposite behavior: it is always biased low. The right hand side of Figure 1 shows the behavior with the corrected estimators we propose below in subsection 4.2.

To see why this biasing behavior arises, fix $y \in Y$ and $x \in X$. We will demonstrate that the bias of our frequency-based estimator of bias^2 holds for any such y and x, which immediately implies that it holds when one averages over y and x. Assume N is even and define $n \equiv N/2$. Given a set of N training sets, one option is to estimate bias^2 using two distinct sets of n training sets, and to then average their estimates to get an average n-training-set-based estimate. Another option is to average over all N training sets directly to get an N-training-set-based estimate. What we will show is that averaged over sets of $N = 2n$ training sets, the former strategy results in a higher estimate of bias^2 than the latter; the expected value of the estimate of bias^2 using n training sets is larger than the expected value using $2n$ training sets. Since the estimate using infinite sets is exactly correct, this means that our estimate of bias^2 is biased upwards more as fewer training sets are used.

Let $i \in \{1,2\}$ specify one of the two sets of n training sets we're examining. Define $\mathbf{w_i}$ to be the average over the training sets in set i of $P(Y_H = y \mid x)$, *i.e.*, $w_i \equiv \sum_j P(Y_H = y \mid x, d_{ij})/n$, where d_{ij} is the j'th training set in set i. Let $\mathbf{T}$ be $P(Y_F = y \mid x)$ (see Equation 3). We can now compute the expected difference between the two ways of estimating the bias (estimating for each set of n training sets and then averaging, minus estimating based on the full set of $N = 2n$ training sets):

$$E\left[\tfrac{1}{2}\textstyle\sum_{i=1,2}(T-w_i)^2 - \left(T - \tfrac{1}{2}\sum_{i=1,2} w_i\right)^2\right] =$$

$$\tfrac{1}{2}\textstyle\sum_{i=1,2} E\left[(T-w_i)^2\right] - E\left[\left(T-\tfrac{1}{2}\sum_{i=1,2} w_i\right)^2\right].$$

By Jensen's inequality (Cover & Thomas 1991), the first term on the right hand side is larger or equal to the second term. This shows that when we average once (over $2n$ instances) rather than twice (over n instances) we get a smaller estimate for the bias^2, which establishes the proposition. A similar argument holds for variance, which gets larger as N grows.

4.2 Unbiased estimators of bias^2 and variance

One way to correct these biases in our frequency counts estimators is to use a very large number of training sets. Although such an approach would have been feasible for our experiments, it is not in general; for many combinations of learning problem and learning algorithm, the associated computational requirements are prohibitive.

Fortunately, there is a straight-forward correction we can apply to our estimators to make them unbiased.[1] Define w_N as the frequency count estimate for $P(Y_H = y_h|f,m,x)$ based on the N training sets. Define $U_N \equiv w_N - T$, where T was defined above as $P(Y_F = y \mid x)$. The frequency counts estimate of bias^2 we used before was U_N^2; our proposed estimator is $V_N \equiv U_N^2 - w_N(1-w_N)/(N-1)$.

Proposition 2 *Under the assumption that we know T's value, V_N is an unbiased estimator for bias^2.*

Proof: Since bias$^2 = (E[U_N \mid n,T])^2$, we have to show that

$$E\left[U_N^2 - w_N(1-w_N)/(N-1) \mid N,T\right] = (E[U_N \mid n,T])^2$$

.

[1]It is not obvious that one would want an unbiased estimator for the estimated quantities, since that does not necessarily minimize expected error. (That is what the bias-variance tradeoff is all about.) However, we felt that getting an unbiased estimator would help understand the problem better and experiments we did indicated that the expected squared-error loss for our proposed unbiased estimator is smaller than that of the estimator based on the naive use of frequency counts.

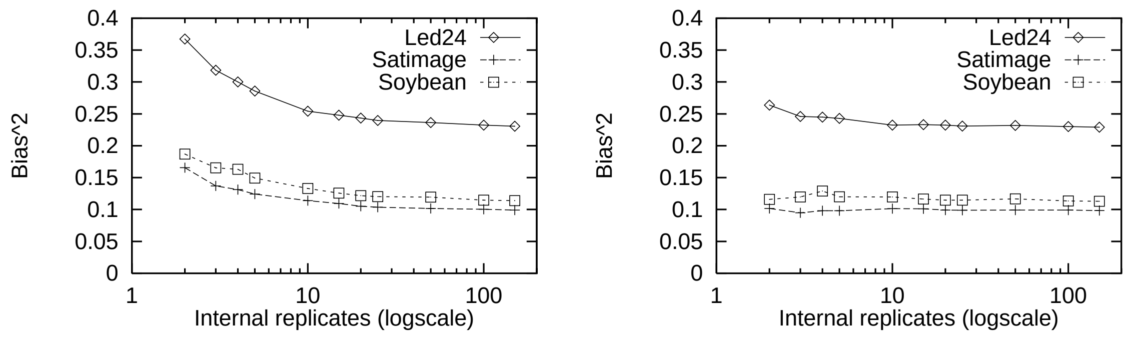

Figure 1: The uncorrected bias (left) and corrected (right)

An unbiased estimator for the variance of the probability in an N-sample Bernoulli process is the empirical variance multiplied by $N/(N-1)$. Furthermore, because T is fixed, the variance of U_N is equal to the variance of w_N. Thus we have

$$\mathrm{var}(U_N|T,n) = \mathrm{var}(w_N|n) = w_N(1-w_N)/(N-1) \quad (5)$$

But by definition of variance, we have

$$\mathrm{var}(U_N|T,n) = E[U_N^2|T,n] - (E[U_N|T,n])^2 \quad (6)$$

Equating the right sides of Equations 5 and 6, we have the desired result. ∎

The correction to the variance estimator is simply given by negative the correction to the bias^2 estimator. This can be shown using reasoning similar to the proof just above, but it also follows from the bias-variance decomposition itself and the fact that the frequency counts estimator of error gives an unbiased estimate of the zero-one loss.

It is important to realize that Proposition 2 implicitly assumes that training sets are formed by i.i.d. sampling a training-set-generating process. This is true for our experimental methodology even though each running of our training-set-generating process is required to produce training sets that contain no duplicate $x-y$ pairs; for the Proposition to apply it suffices to allow duplicate training sets.

We conclude this section with a few comments.

1. The assumption underlying Proposition 2 that we know T exactly is false, since we can only estimate it from the data. However, our estimate of T is a constant, independent of the number of training sets or the details of the learning algorithm. So errors in our estimate of T are not important.
2. A related point is that σ^2 is very difficult to estimate in practice. Using a frequency count estimator the estimate of σ^2 would be zero if all instances are unique (regardless of the true σ^2). Since in the UCI datasets, almost all instances are unique, we elected to define σ^2 to be zero by considering all instances to be unique. This can be viewed as a calculational convenience, since we are only concerned with the variation in expected cost as we vary the learning algorithm, and the estimate of σ^2 is algorithm-independent
3. There is a possibility that V_N will give a negative estimate of bias^2. This is to be expected, given that V_N is unbiased. If the true bias^2 equals zero, for an estimator with variance greater than zero to produce zero as its average estimate (as it must if it is an unbiased estimator), it must sometimes produce negative numbers.

   This potentially negative estimate should be contrasted with the negative variances accompanying the bias-variance decomposition in Kong & Dietterich (1995) and Breiman (1996). Unlike in their decomposition, in our decomposition the true bias^2 and variance are always non-negative; the potential for a negative value is a reflection of the more problematic aspects of unbiased estimators, rather than of the underlying quantity being estimated. In practice though, we always had enough data so that negative bias^2 estimates never occurred.

4.3 The Experiments

We now present experiments demonstrating the bias and variance of induction algorithms for datasets from the UCI repository (Murphy & Aha 1996). The datasets were chosen so that they contain at least 500 instances (to ensure accurate estimates of error) and

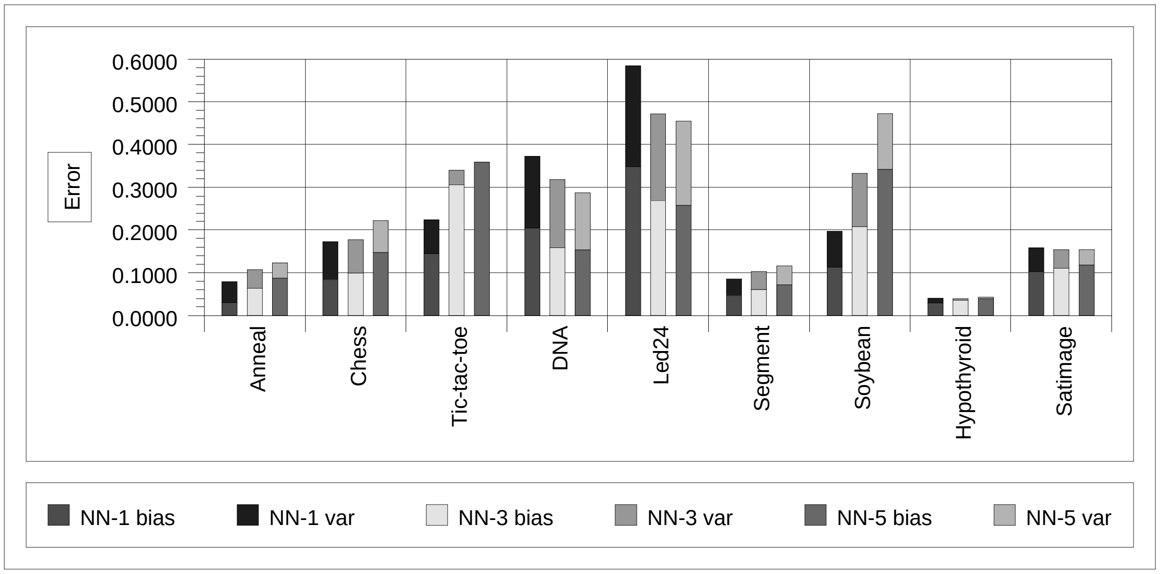

Figure 2: The bias plus variance decomposition for nearest-neighbor with varying number of neighbors. The notation NN-k indicates vote among the k closest neighbors.

Table 1: The Datasets and their characteristics

Dataset	No. features	Dataset Size	Train-set size
Anneal	38	898	100
Chess	36	3196	250
DNA	180	3186	100
LED-24	24	3200	250
Hypothyroid	25	3163	250
Segment	19	2310	250
Satimage	36	6435	250
Soybean-large	35	683	100
Tic-tac-toe	9	958	100

to demonstrate the range of potential bias-variance behaviors. Table 1 shows a summary of the datasets we use. We use small training sets to make sure the evaluation set is large. In general, we chose size 100 for datasets with less than 1,000 instances and 250 for those with over 1,000 instances (except DNA which has 180 features). We generated 50 training sets for each learning algorithm (*i.e.*, $N = 50$) and use the unbiased estimators discussed above.

4.4 Varying the Number of Neighbors in Nearest-Neighbor

Figure 2 shows the bias-variance decomposition of the error for the nearest-neighbor algorithm with a varying number of neighbors used.

In Anneal, Chess, and Tic-tac-toe, increasing the number of neighbors increases the bias and decreases the variance; however, the bias increases much more than the decrease in variance and the overall error increases. In Anneal, the bias more than doubles when going from one neighbor to three neighbors. In Tic-tac-toe, the variance drastically decreases (it is 0.0001 for five neighbors). The reason for this decrease in variance is that instances in Tic-tac-toe vary on one to nine squares; allowing neighbors with up to five differing squares causes NN-5 to let a large portion of the space vote, in effect predicting the majority class (a constant win for X) all the time.

In DNA and Led24, both bias^2 and variance decrease as the number of neighbors increases. For DNA, bias^2 shrinks by 0.07 and the variance by 0.05; for led24, bias^2 shrinks by 0.09 and the variance by 0.05.

In Segment and Soybean, both the bias^2 and variance increase as the number of neighbors is increased. We have observed a general increase in variance when the number of classes is large. Segment has seven classes and Soybean has 19 classes. Increasing the number of neighbors causes many ties which are broken arbitrarily, thus increasing the variance.

In Hypothyroid and Satimage, the changes in bias^2 and variance as the number of neighbors is changed are small and almost cancel. For Hypothyroid, increasing from one to three neighbors increases bias^2 from

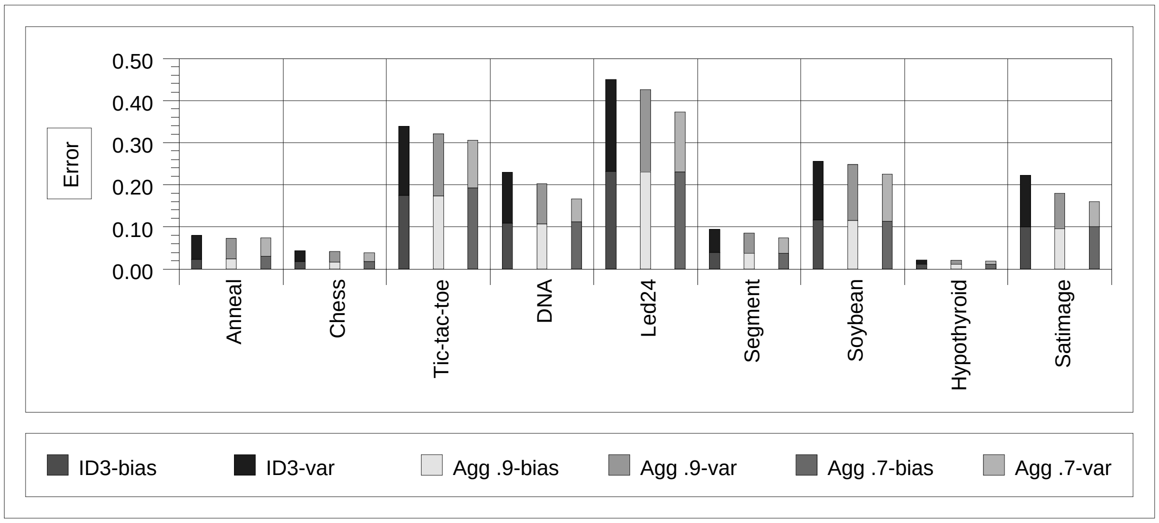

Figure 3: The bias and variance of ID3 and 50 aggregated ID3, with 0.9 and 0.7 of the training set. Aggregation increases the bias slightly but stabilizes ID3 thus reducing the variance more.

0.029 to 0.035 and decreases the variance from 0.011 to 0.004. For Satimage, bias^2 increases from 0.103 to 0.118 and the variance decreases from 0.055 to 0.037.

Although most datasets exhibit a bias-variance tradeoff where one quantity goes up and the other goes down as a parameter of the induction algorithm is varied, we can see examples where both change in the same direction.

4.5 Combining Classifiers

There has been a lot of work recently on combining classifiers, with the terms aggregation, averages, ensembles, classifier combinations, voting, and stacking commonly used (Wolpert 1992, Breiman 1994*a*, Perrone 1993, Ali 1996). In the simplest scheme, multiple classifiers are generated and then vote for each test instance, with the majority prediction used as the final prediction.

Figure 3 shows ID3 versus a combination of 50 aggregated trees. The 50 trees are generated by repeatedly sampling a subset of the training set and running ID3. In contrast to bagging as defined in Breiman (1994*a*), the samples used here were generated by uniform sampling without replacement. Two sample sizes are shown: training sets with 0.7 of the training set and with 0.9. Minor variations in the training set may cause a different split, which might change the whole subtree. As a consequence, decision trees are very unstable, and therefore they usually gain by aggregation techniques (Breiman 1994*b*). Note also that the smaller the internal sample, the more bias we potentially add (Gordon & Olshen 1984) but the more different the classifiers will be, leading to a more stable average.

Our results show that in this voting scheme, the reduction in error is almost solely due to the reduction in variance. While the bias goes up slightly, especially as smaller training sets are used (samples of size 0.7), the reduction in variance is significant and the total error usually decreases. For our datasets, voting samples of size 0.9 always reduced the error. Voting samples of size 0.7 reduced the error even more in all datasets but one (anneal).

5 Summary and Future Work

We presented a bias and variance decomposition for misclassification error, which is equivalent to zero-one loss, and showed that it obeys some desired criteria. This decomposition does not suffer from the shortcomings of the decompositions suggested by Kong & Dietterich (1995) and Breiman (1994*b*). We showed how estimating the terms in the decomposition using frequency counts leads to biased estimates and explained how to get unbiased estimators, which we later used. We then gave some examples of the bias-variance tradeoff using two machine learning algorithms and several UCI datasets.

In the future we plan to investigate many extensions of this work. In particular, we plan to investigate the following topics:

1. The decomposition in Equation 2 holds for essentially *any* conditioning event. It even holds if the conditioning event is (d, x), as in Bayesian point estimation; we have a bias-variance decomposition for a Bayesian quantity. The decomposition also holds if x is not in the conditioning event, so that no external average over x is required, as it is in this paper. We intend to explore these alternative conditioning events.
2. Since variance can be directly estimated using the unlabelled instances of the test set, estimating overall error of a learning algorithm hinges on estimating its bias^2. We would like to test whether better error estimates can be obtained by using cross-validation-style partitions of the training set to estimate that bias and using unlabelled instances from the test set to estimate the variance.

The bias and variance decomposition is an extremely useful tool in Statistics that has rarely been utilized before in Machine Learning because no decomposition existed for misclassification rate. We hope that the proposed decomposition will overcome this problem, and thereby help us improve our understanding of supervised learning.

Acknowledgments

We thank Jerry Friedman and Tom Dietterich for many discussions on the bias-variance tradeoff. The first author thanks Silicon Graphics Inc. The second author thanks the Santa Fe Institute and TXN Inc.

References

Ali, K. M. (1996), Learning Probabilistic Relational Concept Descriptions, PhD thesis, University of California, Irvine. http://www.ics.uci.edu/~ali.

Breiman, L. (1994*a*), Bagging predictors, Technical Report Statistics Department, University of California at Berkeley.

Breiman, L. (1994*b*), Heuristics of instability in model selection, Technical Report Statistics Department, University of California at Berkeley.

Breiman, L. (1996), Bias, variance, and arcing classifiers, Technical report, Statistics Department, University of California, Berkeley. Available at: http://www.stat.Berkeley.EDU/users/breiman/.

Cover, T. M. & Thomas, J. A. (1991), *Elements of Information Theory*, John Wiley & Sons, Inc.

Dietterich, T. G. & Kong, E. B. (1995), Machine learning bias, statistical bias, and statistical variance of decision tree algorithms, Technical report, Department of Computer Science, Oregon State University.

Efron, B. & Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Chapman & Hall.

Geman, S., Bienenstock, E. & Doursat, R. (1992), "Neural networks and the bias/variance dilemma", *Neural Computation* **4**, 1–48.

Gordon, L. & Olshen, R. (1984), "Almost sure consistent nonparametric regression from recursive partitioning schemes", *Journal of Multivariate Analysis* **15**, 147–163.

Kong, E. B. & Dietterich, T. G. (1995), Error-correcting output coding corrects bias and variance, *in* A. Prieditis & S. Russell, eds, "Machine Learning: Proceedings of the Twelfth International Conference", Morgan Kaufmann Publishers, Inc., pp. 313–321.

Murphy, P. M. & Aha, D. W. (1996), UCI repository of machine learning databases, `http://www.ics.uci.edu/~mlearn`.

Perrone, M. (1993), Improving regression estimation: averaging methods for variance reduction with extensions to general convex measure optimization, PhD thesis, Brown University, Physics Dept.

Quinlan, J. R. (1986), "Induction of decision trees", *Machine Learning* **1**, 81–106. Reprinted in Shavlik and Dietterich (eds.) *Readings in Machine Learning*.

Wolpert, D. (submitted), On bias plus variance, SFI TR 95-08-074 in ftp.santafe.edu/pub/dhw_ftp/bias.plus.ps.

Wolpert, D. H. (1992), "Stacked generalization", *Neural Networks* **5**, 241–259.

Wolpert, D. H. (1994), The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework, *in* D. H. Wolpert, ed., "The Mathemtatics of Generalization", Addison Wesley.