

Chapter 3

Accuracy Estimation

FORECAST: A prediction of the future, based on the past, for which the forecaster demands payment in the present.
—Unix fortune

Estimating the accuracy of a classifier induced by a supervised learning algorithm is important not only in order to predict its future performance, but also in order to choose a classifier from a given set (model selection), or in order to combine classifiers. Although the study of accuracy estimation was originally motivated by the fact that the wrapper approach (Chapter 4) requires estimating the accuracy of hundreds of classifiers, the results and observations are not limited to use in wrappers and could be of general use.

We review accuracy estimation methods, compare cross-validation and the .632 bootstrap, and look at methods to stabilize the estimate of cross-validation. We report on a large-scale experiment—over a million runs of C4.5 and a Naive-Bayes algorithm—to estimate the effects of different variants of cross-validation on real-world datasets. For the .632 bootstrap, we vary the number of samples; for cross-validation, we vary the number of folds, the number of times the folds are generated, stratification, and trimming. The number of times cross-validation is executed has a large effect on the variance of k -fold cross-validation for k less than ten; in fact, the variance decreases as k decreases if repeated cross-validation is done; this result contrasts with the U-shape curved when a single cross-validation is done. We conclude that for model selection and datasets similar to ours, it is best to use k -fold cross-validation for low values of k (*i.e.*, $k \leq 10$), even if computation power allows using more folds (*e.g.*, leave-one-out).

3.1 Introduction

Today's data analyst can afford to expend more computation on a single problem than the world's yearly total of statistical computation in the 1920s.
—Efron & Tibshirani (1991)

Estimating the accuracy of a classifier induced by a supervised learning algorithm is important not only in order to predict its future performance, but also in order to choose a classifier from a given set (model selection) as in Schaffer (1993), or in order to combine classifiers (Wolpert 1992b, Breiman 1994a). For estimating the final accuracy of a classifier, we would like an estimation method with low bias and low variance. To choose a classifier or to combine classifiers, the absolute accuracies are less important, and we are willing to trade off bias for low variance, assuming the bias affects all classifiers similarly (*e.g.*, estimates are 5% pessimistic).

Computer power has grown to a point where computer intensive methods for accuracy estimation are used more often and on larger datasets. Methods such as cross-validation and bootstrap require parameters that determine the quality of the estimates. In this chapter, we explain some of the assumptions made by the different estimation methods and present concrete examples where each method fails. While it is known that no accuracy estimation can be correct all the time (Wolpert 1994b, Schaffer 1994), we are interested in identifying methods that are well suited for the biases and trends in typical real world datasets.

For years it was generally assumed that higher folds for cross-validation (up to leave-one-out) would yield better estimates, usually at the expense of longer computation time. For example, Weiss & Kulikowski (1991) write that “While leaving-one-out is a preferred technique, with large sample sizes it may be computationally quite expensive.” Mosteller & Tukey (1968) wrote the following: “Suppose that we set aside one individual case, optimize for what is left, then test on the set-aside case. Repeating this for every case squeezes the data almost dry.”

The use of incremental induction algorithms, however, allows cross-validation in time that is independent of the number of folds (Kohavi 1995a, Moore & Lee 1994, Utgoff 1994). With such algorithms, leave-one-out takes exactly the same time as ten-fold cross-validation; is it clear that leave-one-out should be preferred? While leave-one-out is almost unbiased (Lachenbruch 1967, Glick 1978, Efron 1983), it has high variance, leading to unreliable estimates. Recent results, both theoretical and experimental, have shown that it is not

always the case that increasing the number of folds is beneficial, especially if the variance is more important than the bias, as is the case with model selection.

On the theoretical side, using leave-one-out cross-validation for model selection of linear models is asymptotically inconsistent: the probability of selecting the model with the best predictive power does not converge to one as the total number of observations approaches infinity (Zhang 1992*b*, Shao 1993). On the experimental side, Breiman & Spector (1992) and Kohavi (1995*a*) found that five-fold and ten-fold cross-validation are better for model selection of feature subsets than leave-one-out.

In this chapter, we assess the different effects that parameters have on the bias and variance of the two best-known accuracy estimators: the .632 bootstrap and cross-validation. For the .632 bootstrap, we vary the number of samples; for cross-validation, we vary the number of folds, the number of times cross-validation is executed, and whether or not a trimmed mean is used.

This chapter is organized as follows. In Section 3.2, we describe the common accuracy estimation methods and ways of computing confidence bounds. In Section 3.3, we discuss the methodology underlying our experiments. In Section 3.4, we evaluate the bias and variance of cross-validation and the .632 bootstrap and in Section 3.5, we evaluate cross-validation variants in an attempt to stabilize the accuracy estimates. In Section 3.6, we describe a technique to extrapolate the learning curve in order to reduce the bias. In Section 3.7, we discuss related work and we conclude with a discussion of future work and a summary in Sections 3.8 and 3.9.

3.2 Methods for Accuracy Estimation

The term assessment is preferred to validation which has a ring of excessive confidence about it.
—Stone (1974)

Given a finite dataset, we would like to *estimate* the future performance of a classifier induced by the given inducer and dataset. A single accuracy estimate is usually meaningless without a confidence interval; thus we will consider how to approximate such an interval when possible. An excellent review of accuracy estimation (not including bootstrap) in the context of nearest-neighbor algorithms can be found in Devijver & Kittler (1982, Chapter 10). McLachlan (1992, Chapter 10) provides a review of more recent methods from a statistical perspective.

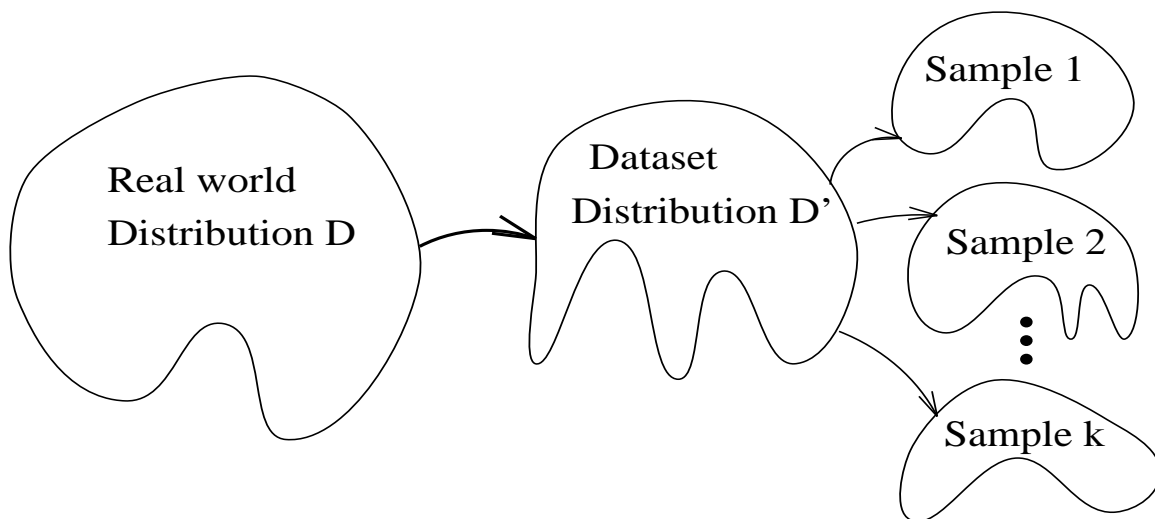


Figure 3.1: Accuracy estimation techniques, such as holdout, cross-validation, and bootstrap, are all based on the idea of resampling.

Except for the resubstitution estimate, all the non-parametric estimators that will be explained are based on the idea of *resampling*, depicted in Figure 3.1. The real world has some unknown distribution D . Our dataset has distribution D' , which is assumed to be similar to the distribution D . In order to estimate the accuracy of inducers trained based on D' , we create samples from D' , train on these samples, and test on instances from D' (usually out-of-sample instances). We thus simulate the sampling process that occurred in the real world, assuming that D' is the real world.

3.2.1 Resubstitution Estimate

All the present recommendations are predicated on small-to-moderate training sample sizes, perhaps in the range of 10-50. ... for large samples, there is no need to look further than the resubstitution estimator when seeking a robust method.

—Knoke (1986)

The resubstitution estimate of the accuracy, sometimes called apparent accuracy, tests the classifier on the same data given to the inducer. Formally, we have

$$\text{acc}_s = \frac{1}{m} \sum_{\langle \vec{x}_i, y_i \rangle \in \mathcal{D}} \delta(\mathcal{I}(\mathcal{D}, \vec{x}_i), y_i) \quad (3.1)$$

where $\delta(i, j) = 1$ if $i = j$ and 0 otherwise.

The resubstitution estimate is a highly optimistic estimate of accuracy because classification procedures attempt to minimize it. For restricted hypothesis spaces, *e.g.*, linear discrimination, where it is usually hard to fit large amounts of data, the resubstitution estimate is reasonable. For example, Knoke (1986) (quoted above) wrote that for large samples, the resubstitution estimate is the best accuracy estimation to use, but he assumes that the classification is based on linear discriminant functions. For many induction algorithms that perfectly fit the data, such as one nearest-neighbor or decision tree induction algorithms that do not prune, the resubstitution estimate is very optimistic; if there are no conflicting instances, the accuracy estimate will be 100%.

3.2.2 Holdout

The holdout method, sometimes called test sample estimation, randomly partitions the data into two mutually exclusive subsets called the training set and the test set, or holdout set. It is common to designate 2/3 of the data as the training set and the remaining 1/3 as the test set. The training set is given to the inducer, and the induced classifier is tested on the test set. Formally, let \mathcal{D}_h , the holdout set, be a subset of \mathcal{D} of size h , and let \mathcal{D}_t be $\mathcal{D} \setminus \mathcal{D}_h$. The holdout estimated accuracy is defined as

$$\text{acc}_h = \frac{1}{h} \sum_{\langle \vec{x}_i, y_i \rangle \in \mathcal{D}_h} \delta(\mathcal{I}(\mathcal{D}_t, \vec{x}_i), y_i), \quad (3.2)$$

where $\delta(i, j) = 1$ if $i = j$ and 0 otherwise. Assuming that the inducer's accuracy increases as more instances are seen, the holdout method is a pessimistic estimator because only a portion of the sample is given to the inducer for training. The more instances we leave for the test set, the higher the bias of our estimate; however, fewer test set instances will cause the confidence interval for the accuracy to be wide, as shown below.

The classification of each test instance can be viewed as a Bernoulli trial: either correct or incorrect labelling. Let S be the number of correct classifications on the test set, then S is distributed binomially (sum of Bernoulli trials). For reasonably large holdout sets, the distribution of S/h is approximately normal with mean acc (the true accuracy of the classifier) and a variance of $\text{acc} * (1 - \text{acc})/h$. Thus, by applying the De Moivre-Laplace

limit theorem, we have

$$\Pr \left\{ -z < \frac{\text{acc}_h - \text{acc}}{\sqrt{\text{acc}(1 - \text{acc})/h}} < z \right\} \approx \gamma, \quad (3.3)$$

where z is the $(1 + \gamma)/2$ -th quantile point of the standard normal distribution. To get a 100γ percent confidence interval, one determines z and inverts the inequalities.

Inversion of the inequalities leads to a quadratic equation in acc , the roots of which are the low and high confidence points:

$$\frac{2h \cdot \text{acc}_h + z^2 \pm z \cdot \sqrt{4h \cdot \text{acc}_h + z^2 - 4h \cdot \text{acc}_h^2}}{2(h + z^2)}. \quad (3.4)$$

The above equation is not conditioned on the dataset \mathcal{D} ; if background information is available about the probability of the given dataset, it must be taken into account.

A simpler expression can be derived by the plug-in-estimate acc_h for acc in the denominator of Equation 3.3, yielding the following low and high confidence points:

$$\text{acc}_h \pm z \cdot \sqrt{\frac{\text{acc}_h(1 - \text{acc}_h)}{h}}. \quad (3.5)$$

This simpler approximation is inaccurate for small values of h and for extreme accuracies. For example, if the estimated accuracy is 100%, a zero width confidence interval is obtained for any value of z when using this approximation.

The holdout estimate is a random number that depends on the division into a training set and a test set. In **random subsampling**, the holdout method is repeated k times, for different random partitions, and the estimated accuracy is derived by averaging the estimated holdout accuracies. The standard deviation of the accuracy can be estimated as the standard deviation of the accuracy estimations from each holdout run. Note, however, that one cannot compute the standard deviation of the mean by dividing the population mean by \sqrt{k} because the estimates are neither independent nor approximately so (the same test instances are used multiple times). Regrettably, many researchers in the field claim the significance of their results by increasing k until the difference, as small as it is, is made significant. To compute the variance, we recommend using the percentile method (Efron & Tibshirani 1993, pp. 168-176). A $1 - 2\alpha$ confidence interval is defined by taking the α and $1 - \alpha$ quantiles of the estimates. This procedure requires that k be large, say at least 50, so

that there is enough data to estimate the 5% mass in the tails if a 95% confidence interval is sought.

The main assumption that is violated in random subsampling is the independence of instances in the test set from those in the training set. Under repeated samplings, the union of the training set and test set is constrained to be equal to the given dataset. If the training set and test set are formed by a split of an original dataset, then an over-represented class in one subset will be under-represented in the other. To demonstrate the issue, we simulated a 2/3, 1/3 split of Fisher’s famous iris dataset (Fisher 1936) and used a majority inducer that builds a classifier predicting the prevalent class in the training set. The iris dataset describes iris plants using four continuous features, and the task is to classify each instance (an iris) as Iris Setosa, Iris Versicolour, or Iris Virginica. For each class label, there are exactly one third of the instances with that label (50 instances of each class from a total of 150 instances); thus we expect 33.3% prediction accuracy. However, because the test set will always contain less than 1/3 of the instances of the class that was prevalent in the training set, the accuracy predicted by the holdout method is 27.68% with a standard deviation of 0.13% (estimated by averaging 500 holdouts).

In practice, the dataset size is always finite, and usually smaller than we would like it to be. The holdout method makes inefficient use of the data: a third of the dataset is not used for training the inducer. The next method mitigates the problem of hiding a large portion of the dataset from the algorithm.

3.2.3 Cross-Validation, Leave-one-out, and Stratification

Since the basic philosophy of cross-validation is non-probabilistic and non-parametric, it is perhaps not surprising that supporting theory is rather meagre.
—Stone (1978)

In k -fold cross-validation, sometimes called rotation estimation, the dataset \mathcal{D} is randomly split into k mutually exclusive subsets (the folds) $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k$ of approximately equal size. The inducer is trained and tested k times; each time $t \in \{1, 2, \dots, k\}$, it is trained on $\mathcal{D} \setminus \mathcal{D}_t$ and tested on \mathcal{D}_t .¹ The cross-validation estimate of accuracy is the overall number of correct classifications, divided by the number of instances in the dataset. Formally, let $\mathcal{D}_{(i)}$ be the test set that includes instance $\langle \vec{x}_i, y_i \rangle$, then the cross-validation estimate of accuracy

¹The notation $A \setminus B$ indicates set A minus set B .

is

$$\text{acc}_{\text{cv}} = \frac{1}{m} \sum_{\langle \vec{x}_i, y_i \rangle \in \mathcal{D}} \delta(\mathcal{I}(\mathcal{D} \setminus \mathcal{D}_{(i)}, \vec{x}_i), y_i) . \quad (3.6)$$

The cross-validation estimate is a random number that depends on the division into folds. **Complete cross-validation** is the average of all $\binom{m}{m/k}$ possibilities for choosing m/k instances out of m , but it is usually too expensive (Geisser 1975). Except for leave-one-out (m -fold cross-validation), which is always complete, k -fold cross-validation is estimating complete k -fold cross-validation using a single split of the data into the folds. Repeating cross-validation multiple times using different splits into folds provides a better Monte-Carlo estimate to the complete cross-validation at an added cost.

Stone (1974) is credited with the first formal description of cross-validation and its uses for choosing statistical predictors; however, he credited Mosteller, Wallace, and Lachenbruch for the idea. Lachenbruch (1967) described leave-one-out as a method for obtaining confidence intervals. Linhart & Zucchini (1986) claimed that the idea was explicitly stated in psychometric literature in the 1930s.

In **stratified cross-validation**, the folds are stratified so that they contain approximately the same proportions of labels as the original dataset. Breiman *et al.* (1984, Section 8.7) claimed that, especially for regression, stratification is the preferred method for selecting the right-sized tree; the cross-validation used in CART and in the C4.5 cross-validation utility, `xval.sh`, is stratified; and Weiss, who has done much work in accuracy estimation, uses stratified cross-validation in his experiments (Weiss 1991, Weiss & Indurkha 1994a, Weiss & Indurkha 1994b). An asymptotic ($m \rightarrow \infty$) theoretical analysis done by Olshen, Gilpin, Henning, LeWinter, Collins & Ross (1985) showed that if the label is independent of the features (*i.e.*, random), then for building decision trees by recursive-partitioning, stratification helps a single-fold of cross-validation, in the sense that the mean-squared error is smaller. A later theoretical analysis done by Bai (1988) showed that under some assumptions (finite instance space, unequal class probabilities), stratification neither helps nor hurts two-fold cross-validation in terms of mean squared error because it increases the covariance between the folds.

The cross-validation accuracy is sometimes defined as the average of the estimated accuracies from the k runs, and not as in Equation 3.6. This estimation method is accurate for all but very small samples, and it is the version we use in our experiments because it allows us to test the effect of trimming and the normality assumption (see Section 3.5).