Complete Cross-Validation for Nearest Neighbor Classifiers

Matthew Mullin MDM@JUSTRESEARCH.COM
Just Research, 4616 Henry Street, Pittsburgh, PA 15213 USA

Rahul Sukthankar RAHULS@CS.CMU.EDU
Just Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Cross-validation is an established technique for estimating the accuracy of a classifier and is normally performed either using a number of random test/train partitions of the data, or using k-fold cross-validation. We present a technique for calculating the complete cross-validation for nearest-neighbor classifiers: i.e., averaging over all desired test/train partitions of data. This technique is applied to several common classifier variants such as K-nearest-neighbor, stratified data partitioning and arbitrary loss functions. We demonstrate, with complexity analysis and experimental timing results, that the technique can be performed in time comparable to k-fold cross-validation, though in effect it averages an exponential number of trials. We show that the results of complete cross-validation are biased equally compared to subsampling and k-fold cross-validation, and there is some reduction in variance. This algorithm offers significant benefits both in terms of time and accuracy.

1. Introduction

In supervised learning (Mitchell, 1997), a pool of labeled data, S, is used to predict the labels of unseen data. Since the distribution of data is unknown, the generalization accuracy (the probability that an unseen data point will be correctly classified) cannot be directly determined. Using S, we can calculate an empirical accuracy and this can be used as an estimate of the generalization accuracy.

Two accepted techniques for estimating the generalization accuracy are *subsampling* and *k-fold cross-validation*. In the former, S is partitioned into a training set, T, and a test set $S \backslash T$.[1] The empirical accuracy is given by the fraction of test set items labeled correctly by a classifier trained on T. In the latter, S is partitioned into k equally-sized subsets. Each subset is used as a test set for a classifier trained on the remaining $k-1$ subsets. The empirical accuracy is given by the average of the accuracies of these k classifiers. Both techniques may employ a stratified partitioning in which the subsets contain approximately the same proportion of classes as S.

The empirical accuracy obtained using either of these techniques depends on the items from S that appeared in a given subset, and is therefore a random variable,[2] and any given trial of subsampling or k-fold cross-validation is equivalent to a single observation. The best estimate of the generalization accuracy is given by the expectation of this random variable, termed *complete cross-validation* (Kohavi, 1995). However, directly calculating this expectation requires averaging over all permissible partitions of S and is generally impractical since the number of such partitions grows exponentially with $|S|$. Therefore, it is customary to create an estimate by averaging the accuracies from a manageable number of partitions (Mitchell, 1997). Unfortunately, the variance of the random variable can be significant and obtaining an estimate with sufficiently low variance can require a large number of time-consuming trials.

This paper presents a technique for calculating the complete cross-validation (CCV) for the nearest neighbor family of classifiers (including the popular K-nearest-neighbor algorithm). Nearest neighbor methods (Dasarathy, 1991) frequently appear at the core of sophisticated pattern recognition and information retrieval systems. For instance, a diversity of face recognition methods such as elastic bunch-graph matching (Wiskott et al., 1997), eigenfaces (Turk & Pentland, 1991), and the support vector machine face recognizer (Phillips, 1998) all fall into this family. Similarly, wavelet-based image retrieval (Jacobs et al., 1995), vector space model for text-retrieval (Salton, 1971) and latent se-

[1] $S \backslash T$ denotes the *set difference* of S and T: the elements of S that are not in T.

[2] The random variables corresponding to the two techniques are generally different.

mantic indexing (Deerwester et al., 1990) also employ variants of nearest-neighbor methods in their matching phase. Not only is our complete cross-validation accuracy identical to that obtained by averaging over the exhaustive enumeration of partitions, it can be computed very efficiently.

The remainder of the paper is organized as follows. Section 2 describes the general technique, applies it to several classifier types and provides a complexity analysis for each variant, showing that complete cross-validation is efficient. Section 3 shows that the theoretical bias of CCV is equivalent to k-fold cross-validation, and that the variance is typically lower. Section 4 presents empirical results on standard datasets that support these theoretical claims. Section 5 concludes with an outline of future research.

2. Expectation of Empirical Accuracy

This section describes a general method for efficiently calculating the complete cross-validation accuracy for the nearest-neighbor family of classifiers. Our goal is to calculate the *expectation of the empirical accuracy*, as given by either k-fold cross-validation or subsampling. We define the following terms:

N	Size of S ($\|S\|$)
$c(x)$	The class label associated with x
C	Number of class labels in S
N_i	Number of elements in class i. $N_i = \|\{x : c(x) = i\}\|$
k	Number of folds in k-fold cross validation (CV).

First, we simplify the problem by demonstrating that the accuracy obtained using *complete* k-fold cross validation can easily be obtained by evaluating the equivalent *complete* subsampling where $|S \setminus T| = N/k$. Let $P = (p_1, p_2, \ldots, p_k)$ be a partition of S for cross-validation, Subsampling(T) be the empirical accuracy computed by subsampling with training set T and CV(P) be the empirical accuracy computed by k-fold cross-validation with partition P. Then by definition we have:

$$\text{CV}(P) = \frac{1}{k}\sum_{i=1}^{k} \text{Subsampling}(S \setminus p_i).$$

The expectation is, by substitution and linearity:

$$\begin{aligned} E[\text{CV}] &= \frac{1}{k}\sum_{i=1}^{k} E[\text{Subsampling}(S \setminus p_i)] \\ &= \frac{1}{k}\sum_{i=1}^{k} E[E[\text{Subsampling}(S \setminus p_i)|p_i = p]], \end{aligned}$$

by Proposition 6.1 in Ross, 1988 (p. 285). Now:

$$\begin{aligned} E[\text{CV}] &= \frac{1}{k}\sum_{i=1}^{k} E[\text{Subsampling}(S \setminus p)] \\ &= E[\text{Subsampling}(S \setminus p)], \end{aligned}$$

because $E[\text{Subsampling}(S \setminus p)]$ is independent of i. And

$$E[\text{CV}] = E[\text{Subsampling}(T)]$$

by a simple correspondence of a test set p and the training set $T = S \setminus p$. Thus, the remainder of the paper focuses on complete subsampling. Let $\mathcal{T}$ be the set of permissible training sets. The expectation of the empirical accuracy using complete subsampling is simply the proportion of possible classifications that are correctly classified (over all $\mathcal{T}$). The number of possible classifications is $\sum_{T\in\mathcal{T}} |S \setminus T|$, while the total number of *correct* classifications is:

$$A = \sum_{T\in\mathcal{T}} \sum_{x\notin T} \text{correct}(x, T), \tag{1}$$

where the binary function, correct(x, T), returns 1 iff x is correctly labeled[3] by a classifier trained on T. Equation 1 is a large sum (with $|\mathcal{T}|$ terms) of small inner sums (each with fewer than N terms). We focus on efficiently computing this seemingly-intractable sum. We begin by manipulating Equation 1 into a more manageable form:

$$\begin{aligned} A &= \sum_{T\in\mathcal{T}} \sum_{x\notin T} \text{correct}(x, T) && (2) \\ &= \sum_{\substack{(x,T)\in S\times\mathcal{T} \\ x\notin T, T\in\mathcal{T}}} \text{correct}(x, T) \\ &= \sum_{x\in S} \underbrace{\sum_{\substack{T\in\mathcal{T} \\ T\not\ni x}} \text{correct}(x, T)}_{A_x}. && (3) \end{aligned}$$

This identity expresses the key insight of our technique: that the total number of correct classifications can be computed in two equivalent ways. In the direct method, the number of correct classifications for a particular test/train split is summed over all possible splits (Equation 2). Alternately, one can count the number of *splits*, A_x, for which a particular item in the test set, x, is correctly identified and sum over every item in the data set (Equation 3). Computationally, this transforms the problem into a small sum of large inner sums. Fortunately, in the case of nearest-neighbor classifiers, A_x can be efficiently calculated without explicit enumeration, enabling us to efficiently obtain our goal (expectation of empirical accuracy).

[3]It is implicit that $x \in S$, so that $x \notin T \Leftrightarrow x \in S \setminus T$.

The training phase of any classifier in the nearest-neighbor family simply consists of storing all of the labeled items in T. In the testing phase, to classify an item x from the test set, the similarity between x and each of the items in T is computed and x is assigned a label computed from the labels of the most similar items in T. The *similarity measure* between items need not satisfy metric properties such as symmetry or the triangle inequality. We require that the similarity between items be invariant over test/train splits so that a *fixed ordering* of items in S, sorted by similarity, can be created for every $x \in S$.[4] For a particular item, x, this ordered list looks like:

$$\begin{array}{ll} x: & x_1\ x_2\ x_3\ \ldots\ x_i\ \ldots\ x_{N-2}\ x_{N-1}. \\ & \leftarrow \text{More similar} \quad \text{Less similar} \rightarrow \end{array}$$

This list contains all of the items in S; for a given split, some of the x_i items will be in the training set and the remainder will be in the test set. A given split correctly classifies x iff its label is correctly computed from the elements in the training set most similar to x. Note that these elements are not necessarily the leftmost elements above, since the leftmost elements may have been assigned to the test set. We show below how the number of splits that predict a given label can be computed without enumeration and demonstrate how this enables us to efficiently evaluate A_x for several classifiers. We introduce the approach using 1-nearest-neighbor and then generalize to loss functions, stratified datasets, K-nearest-neighbor and rank-scoring.

2.1 Classifier 1: 1-Nearest-Neighbor with Fixed $|T|$

Despite its simplicity, the 1-nearest-neighbor (1-NN) classifier has good theoretical properties (Cover & Hart, 1967), and has been successfully applied to a broad range of problems. We first consider the case of complete subsampling where the training set is of a fixed size, $|T| = \alpha$, and the classifier is 1-nearest-neighbor. In the context of the discussion above, a given split correctly classifies x iff x and its nearest neighbor (most similar element in T) have the same class label.

Consider the set of training sets, $\mathcal{E} \subset \mathcal{T}$, where the element most similar to x's is denoted x_i. If $c(x) = c(x_i)$, then all classifiers using $T \in \mathcal{E}$ correctly classify x, else none of them do; thus, the number of total correct classifications is incremented by $|\mathcal{E}|$ iff $c(x) = c(x_i)$. By examining the additional constraints on $\mathcal{E}$, we obtain $|\mathcal{E}|$ without explicitly enumerating $\mathcal{E}$. Our constraints are: (1) the items $x_1, \ldots, x_{i-1}$ cannot be in T, since x_i is the most similar element in T to x; (2) aside from x_i, there are $\alpha - 1$ items in the training set, all of which must appear somewhere in the $N - i - 1$ available positions to the right of x_i (see the diagram above). Therefore, given x and x_i, there are $\binom{N-i-1}{\alpha-1}$ splits that will create training sets satisfying these constraints. Summing over all possible positions x_i gives us the following:

$$A_x = \sum_{i=1}^{N-1} I(x, x_i) \binom{N-i-1}{\alpha-1}, \tag{4}$$

where $I(x_i, x_j)$ is an indicator function that returns 1 if $c(x_i) = c(x_j)$, and 0 otherwise. Equation 4 reduces the number of terms in Equation 3's intractable inner sum from $|\mathcal{T}| = \binom{N}{\alpha}$ to only $N - 1$. Unless $\alpha = 1$, or $\alpha = N - 1$, this improvement is very significant.

2.1.1 Complexity analysis for 1-NN

The time required to calculate A_x using Equation 4 is composed of the following parts:

1. Computing similarity between x and each x_i: $O(N)$;
2. Sorting the $N - 1$ items: $(N \log N)$;
3. Summing the $N - 1$ terms in Equation 4: $O(N)$.

Therefore, the time required to obtain A_x is bounded by $O(N \log N)$. Since A_x is computed for each $x \in S$, the total time is:

$$O(N^2 \log N). \tag{5}$$

The binomial coefficients should be calculated once and then stored. In all cases presented, there will be at most a small multiple of N coefficients that need to be stored, and in practice this calculation takes a small fraction of the total running time.

When a significant fraction of the data is to be placed in the training set, A_x can be computed more efficiently as follows. In Equation 4, note that when $i > N - \alpha$, the binomial coefficient $\binom{N-i-1}{\alpha-1} = 0$. So, rather than summing $N-1$ terms, we need only sum the first $N-\alpha$ terms. Also, rather than sorting all $N - 1$ items, we need only determine the $N - \alpha$ nearest neighbors, which can be done with a priority queue (Knuth, 1973) in time $O(N \log(N - \alpha))$. This results in a total execution time of $O(N^2 \log(N-\alpha))$, which can be as low as $O(N^2)$ when $N - \alpha$ is a small constant.

By comparison, a single trial of the subsampling takes $O(\alpha\ (N - \alpha))$. The exhaustive computation (Equation 1) will therefore take $O(\binom{N}{\alpha}\alpha(N - \alpha))$, which is intractable for typical α. In fact, our algorithm is more efficient than averaging the results from $O(\log N)$ randomized trials.

2.2 Classifier 2: 1-NN with Arbitrary Loss Function

Loss functions (Vapnik, 1998) offer a generalization of accuracy by penalizing some misclassifications more than

[4] Any reasonable tie-breaking criterion may be applied.

others and by rewarding some correct classifications more than others. The technique for complete cross-validation described above can easily be adapted to employ loss functions as follows.

Consider again the set of training sets, $\mathcal{E} \subset \mathcal{T}$, where the element most similar to x in T is x_i. All classifiers using $T \in \mathcal{E}$ assign a loss of $L(x, x_i)$; thus, the total loss is incremented by $|\mathcal{E}|L(x, x_i)$. $|\mathcal{E}|$ is determined as in Section 2.1. Summing over all possible positions x_i gives us:

$$A_x = \sum_{i=1}^{N-1} L(x, x_i) \binom{N-i-1}{\alpha - 1}. \qquad (6)$$

The algorithms presented in subsequent sections can also be adapted to include loss functions as described above. The complexity analysis presented in Section 2.1.1 is also valid here.

2.3 Classifier 3: Stratified 1-NN

In some applications, the training set is constrained to contain a certain number of items from each class (typically to reflect the proportions of the classes in S); this is termed *stratification* (Kohavi, 1995). For instance, the ORL face dataset (Samaria & Harter, 1994) contains 10 images from each of 40 individuals. The face recognition experiments presented in (Lawrence et al., 1996; Sim et al., 2000) examine accuracy by varying the size of the training set using 1, 3, or 5 images for each of the 40 classes. In general, we define α_c to be the number of items required from class c to be in the training set. This section shows how complete cross-validation may be applied to this problem, by extending the algorithm described Section 2.1.

Consider the set of training sets, $\mathcal{E} \subset \mathcal{T}$, where the item most similar to x in T is x_i. We first define $f_i(c)$ to be the number of items from a given class, c, that are less similar than a given item, x_i in S (equivalent to the number of items of the given class appearing to the right of x_i in the diagram above):

$$f_i(c) = |\{j : j > i \text{ and } c(x_j) = c\}|. \qquad (7)$$

For each class, $c \neq c(x_i)$, we must select α_c items from a potential $f_i(c)$ candidates for inclusion into the training set T. For class $c = c(x_i)$, since we have already selected x_i, we only need an additional $\alpha_c - 1$ items from the $f_i(c)$ candidates. Since the choices for each class are independent, the number of training sets that correctly classify x is the product of the number of choices for each class. All possible training sets for item x can be generated by considering i from 1 to $N-1$, as above. In a manner analogous to Equation 4, we can efficiently calculate A_x as:

$$A_x = \sum_{i=1}^{N-1} I(x, x_i) \binom{f_i(c(x_i))}{\alpha_{c(x_i)} - 1} \prod_{\substack{c=1 \\ c \neq c(x_i)}}^{C} \binom{f_i(c)}{\alpha_c} \qquad (8)$$

Note that a loss function (see Section 2.2) can be incorporated into Equation 8 simply by replacing $I(x, x_i)$ with $L(x, x_i)$.

2.3.1 Complexity analysis

A similar analysis to the one presented in Section 2.1.1 can be performed to calculate the bounds on execution time for this algorithm. The time required to compute A_x using Equation 8 is composed of the following parts:

1. Similarity computation and sorting (identical to that in Section 2.1.1);
2. Computing $\forall c\ f_i(c)$ (discussed below);
3. Computing the $N-1$ non-zero terms, each requiring $O(C)$ to multiply the binomial coefficients together: $O(CN)$.

$\forall c\ f_i(c)$ can be efficiently computed since $f_{i+1}(c) = f_i(c) - 1$ when $c = c(x_{i+1})$, and is unchanged otherwise. Therefore, the cost of computing $\forall c\ f_i(c)$ is $O(N)$. and the total time needed to compute A_x is $O(N \log N + CN)$. To compute A, we must do this for each $x \in S$; the total time is given by:

$$\sum_{x \in S} O(CN + N \log N) = O(N^2(C + \log N)).$$

A single trial of the subsampling algorithm takes $O(\alpha(N - \alpha))$, where $\alpha \equiv \sum_{c=1}^{C} \alpha_k$. The analysis is very similar to that presented in Section 2.1.1.

2.4 Classifier 4: K-Nearest-Neighbor

The 1-nearest-neighbor algorithm has been criticized for being overly sensitive to labeling errors in the dataset. The K-nearest-neighbor algorithm (Mitchell, 1997) addresses this by considering the K most similar elements in the training set. To simplify discussion, this section focuses on the instance of binary K-nearest-neighbor classification where each of the K neighbors votes equally for its label, and x is assigned the label with the greatest number of votes.[5]

First, we define some additional notation. Consider the set of training sets, $\mathcal{E}_s \subset \mathcal{T}$, where x_i is the K-th most similar item to x in T, and there are exactly s correctly-labeled items in T that are at least as similar as x_i to x. Analogous to the definition of $f_i(c)$ in Section 2.3, we define:

$$g_i(c) = |\{j : j < i \text{ and } c(x_j) = c\}|.$$

[5] We use an odd K to prevent ties.

Without loss of generality, let $\Omega = c(x)$ and denote the other class by $\mho$. Now:

$$A_x = \sum_{i=1}^{N-1} \sum_{s=\frac{k+1}{2}}^{K} |\mathcal{E}_s|, \tag{9}$$

where $|\mathcal{E}_s|$ is given by:

$$\binom{g_i(\Omega)}{s - I_\Omega(x_i)} \binom{g_i(\mho)}{K - s - I_\mho(x_i)} \binom{N-i-1}{\alpha - K}, \tag{10}$$

and $I_\Omega(x_i) = 1$ iff $c(x_i) = \Omega$ and 0 otherwise, and $I_\mho(x_i)$ defined analogously. The expression for $|\mathcal{E}_s|$ consists of three parts:

1. the number of elements more similar than x_i to x of the correct class is $g_i(\Omega)$, from which we must select $s - I_\Omega(x_i)$;
2. the number of elements more similar than x_i to x of the incorrect class is $g_i(\mho)$, from which we must select $K - s - I_\mho(x_i)$;
3. there are a total of $N - i - 1$ elements that are less similar than x_i to x (of either class), from which we must select $\alpha - K$.

2.4.1 Complexity Analysis

The analysis is similar to that presented in Section 2.3.1 with the following differences. Step 2 requires computation of $\forall c\, g_i(c)$ rather than $\forall c\, f_i(c)$, which also takes $O(N)$ time. The summation in Step 3 consists of $O(KN)$ terms taking $O(KN)$ time. Thus, the total time needed to perform complete cross-validation for binary-class K-NN is $O(N^2(K + \log N))$.

2.5 Classifier 5: Stratified K-Nearest-Neighbor

The algorithm to compute complete cross-validation for stratified K-Nearest-Neighbor only requires small modifications to the approach given in Section 2.4. The third term in the expression for $|\mathcal{E}_s|$ (Equation 10) should be replaced by:

$$\binom{f_i(\Omega)}{\alpha_\Omega - s} \binom{f_i(\mho)}{\alpha_\mho + s - k}.$$

This is justified as follows. In the stratified case, (analogous to the discussion in Section 2.3), there are $f_i(\Omega)$ correctly-labeled elements in T that are less similar than x_i to x, of which we must select $\alpha_\Omega - s$. Similarly, out of $f_i(\mho)$ incorrectly-labeled elements in T that are less similar than x_i to x, we require $\alpha_\mho + s - k$.

By substituting $k = 1$ into the above equations, we can confirm that the formulae for K-NN correctly reduce to the equations presented in Sections 2.1 and 2.3 for 1-NN. The complexity analysis for stratified K-NN is almost identical to the analysis given in Section 2.4.1, resulting in the same asymptotic execution time.

2.6 Classifier 6: 1-NN with R-Rank-Scoring

In some domains with large numbers of classes, a classifier is judged to be "correct" iff at least one of the R most similar items in the training set matches the query item's class label. For instance, a face recognition system may be given a noisy photograph, x, and asked to return a list of the R best matches from its database. As long as the identity of one of the retrieved faces is correct, the system is judged to be correct (Phillips et al., 1998).

The technique presented above can be extended to accommodate such a scoring scheme as follows. Consider the set of training sets, $\mathcal{E}_r \subset \mathcal{T}$, where x_i is the most similar correctly-labeled item to x in T, and there are exactly $r - 1$ incorrectly-labeled items in T that are at least as similar as x_i to x. The expression for A_x is:

$$A_x = \sum_{i=1}^{N-1} I(x, x_i) \sum_{r=1}^{R} |\mathcal{E}_r|, \tag{11}$$

where:

$$|\mathcal{E}_r| = \binom{N-i-1}{\alpha - r} \binom{i - N_{c(x)} + f_i(c(x)) + 1}{r - 1}.$$

We justify the expression for $|\mathcal{E}_r|$ as follows. The first term counts the number of ways of selecting $\alpha - r$ elements from the $N - i - 1$ elements that are less similar than x_i to x. The second term counts the number of ways that $r - 1$ elements can be chosen from the incorrectly-labeled items that are more similar than x_i to x.

2.6.1 Complexity Analysis

The analysis is similar to that presented in Section 2.3.1 with the following difference. The summation in Step 3 consists of $O(RN)$ terms: $O(RN)$ time. Thus, the total time needed to perform complete cross-validation for 1-NN using a R-ranked-scoring is $O(N^2(R + \log N))$.

3. Statistical Analysis

In this section, we show that empirical accuracies obtained using k-fold cross-validation, subsampling, or complete cross-validation all exhibit the same statistical bias, but that the estimate obtained using complete cross-validation exhibits smaller variance.

For a fixed set, S, the empirical accuracy obtained using complete cross-validation was shown above to be equal to the expected value obtained using either of the standard

Table 1. Datasets and their relevant characteristics.

Dataset	Sample size/ $\lvert S\rvert$	Dim	Classes
Synthetic	—/100	3	4
Abalone-2	4177/600	8	2
Abalone-3	4177/600	8	3
Breast Cancer	699/140	8	2

techniques. Since these are all estimates of the same generalization accuracy, this implies that the bias is identical. For the given S, the complete cross-validation estimate is not a random variable; therefore, its variance is 0. k-fold cross-validation and subsampling, unless averaged over all partitions, exhibit a non-zero variance.

In reality, S should not be considered fixed, but is rather a sample of N elements drawn from some unknown probability distribution $\mathcal{F}$. In this case, the estimate from complete cross-validation is a function of S, and is therefore also a random variable, with statistical properties expressed by:

$$\begin{aligned} E[\text{CCV}|S] &= E[E[\text{CV}|S]] = E[\text{CV}] \\ &= E[E[\text{Subsampling}|S]] = E[\text{Subsampling}]. \end{aligned}$$

We observe that the empirical accuracies computed by all three methods are still equally biased. Since the complete cross-validation estimate is no longer constant, its variance is non-zero. However, by the conditional variance formula (Ross, 1988) we can write:

$$\begin{aligned} \text{Var}(\text{CV}) &= \text{Var}(E[\text{CV}|S]) + E[\text{Var}(\text{CV}|S)] \\ &= \text{Var}(\text{CCV}|S) + E[\text{Var}(\text{CV}|S)] \\ &\geq \text{Var}(\text{CCV}|S), \end{aligned}$$

showing that, the variance for complete cross-validation is generally smaller. The term, $E[\text{Var}(\text{CV}|S)]$ is the variance due to the random partitions used in creating the k-folds. The same argument can be used to show that complete cross-validation estimates exhibit lower variance than non-exhaustive subsampling. These theoretical results are confirmed by the experiments presented in Section 4.

4. Empirical Results

Table 1 summarizes the datasets used in the experiments reported in this section. *Abalone* and *Breast Cancer* were obtained from the UCI repository (Blake & Merz, 1998). *Synthetic* consists of 4 Gaussians with equal variance and significant overlap, and Bayes Error ≈ 0.504.[6] *Abalone* is a 29 class problem, however many of the classes have only very few instances. *Abalone-2* and *Abalone-3* are two- and three-class versions of the problem, where the adjacent classes were grouped so that data was divided evenly. *Abalone-3* was introduced in (Waugh, 1995). In the *Breast Cancer* dataset, the ID field was omitted, as was a field containing missing values.[7]

Since the aim of these experiments was not to improve classification accuracy but rather to compare estimation variance and timing for different cross-validation strategies, no effort was made to tune classification parameters. Therefore, all of the experiments employed a simple Euclidian distance function on unscaled features. The experimental methodology is summarized as follows.

Averaging over 50 independent runs using dataset D:
- Generate S by drawing without replacement from D
- CCV: $\alpha = 4|S|/5$ (only need to run once)
- Stratified CCV: $\alpha_c = 4N_c/5$ (only need to run once)
- 5-fold CV: averaged over 50 trials (random folds)
- 5-fold stratified CV: averaged over 50 trials
- Subsampling: $\alpha = 4|S|/5$ averaged over 50 splits
- Stratified subsampling: $\alpha_c = 4N_c/5$ averaged over 50
- Compute holdout accuracy: use S to train, $D \setminus S$ to test.

Because of the low dimension and the small S we have chosen, we are able to perform more runs with the *synthetic* dataset, as follows: 100 independent runs were performed, and the 5-fold cross-validations and subsampling trials were repeated 100 times, and data points were synthesized from a known probability distribution. The holdout accuracy was estimated using an additional 500 generated points.

Table 2 summarizes the experimental results. The first block demonstrates that CCV performs well on synthetic data: the empirical accuracy obtained using CCV is consistent with that obtained using either k-fold CV or subsampling. It is also close to the holdout accuracy, indicating that there is little bias. While the time taken to execute a single trial on CCV is greater than that required to execute a single trial using standard techniques, it is clear that a single run of CCV outperforms the average of 100 trials of the other techniques, both in terms of reduced variance and faster execution. Note that a given block of experiments also includes results from the stratified variant of the algorithm. The next block of results demonstrate that CCV also performs favorably on a real-world dataset. The third block compares the performance of the various techniques using the K-nearest-neighbor classifier. The last block compares the techniques for a rank-scored 1-nearest-neighbor classi-

[6] The Bayes error for the *Synthetic* data set was estimated using a Monte-Carlo simulation with 60000 samples.

[7] The breast cancer database is obtained from the University of Wisconsin Hospitals, Madison from Dr. William Wolberg (Mangasarian & Wolberg, 1990).

fier (no stratified variant here). Table 3 shows experimental results for CCV vs. CV and subsampling in the case of an asymmetric loss function. The penalty for misdiagnosing a malignant cancer as benign is set to be five times greater than the penalty for misclassifying a benign tumor as malignant.

From this data, we see that CCV can complete a run in the time taken to evaluate 3–20 *trials* of CV (a small fraction of the time taken to run one CV) and 5–50 trials of random subsampling.[8] In this time, neither of the established methods is able to substantially reduce its variance on estimated accuracy.

5. Conclusion

Nearest-neighbor methods appear frequently (in disguise) in real applications, typically as the final stage of a complex system. This paper presents an efficient technique for calculating the expected empirical accuracy of nearest-neighbor classifiers using complete cross-validation. The general method is specialized to several popular classifier variants: K-nearest neighbor, stratified data partitioning, asymmetric loss functions and ranked-scoring. Theoretical results demonstrate that our method exhibits desirable statistical properties and has favorable asymptotic complexity. Experimental results on a synthetic classification task and also on standard machine learning datasets supports the theoretical claims.

Our technique, which is straightforward to implement, is immediately applicable; for instance, Classifier 3 (Section 2.3) has reduced the time required for parameter optimization in a face recognition system (Sim et al., 2000) from several hours to a few minutes. Thus our technique may be used as a fast cross-validation component in other machine-learning algorithms. We are currently incorporating it into a gradient-descent-based method for feature weighting with encouraging preliminary results. Our plans for future research include:

- Extending the technique to allow statistical hypothesis testing, as in (Dietterich, 1998).
- Addressing a more general set of classifiers. The current system is limited to classifiers where the similarity relations between elements are indepedendent of the training set. We believe that our technique can be extended to kernel-weighted or distance-weighted classifiers.
- Working with very large datasets. The current implementations of our technique do not take advantage of specialized data structures developed for large

[8]Experiments were performed in Matlab 5.3.1 on a PII-300 machine running Linux 2.2.12.

Table 2. A summary of several experiments comparing complete cross-validation (CCV) with standard techniques (k-fold CV and random subsampling). The empirical accuracy (column 2) is multiplied by 10^3 for convenience, as is the variance on the estimate (column 3). The time per run sums up 50 trials for the standard methods on all datasets except for synthetic (where 100 trials are summed). Note that CCV's estimate of accuracy is consistent with that of other methods, and that its variance and execution time compare very favorably.

Experiment	Emp. Acc.	Var	Time (s) per trial	Time (s) per run
1-Nearest-neighbor on *synthetic*				
Holdout	371.2	0.89	0.84	0.8
CCV	369.8	3.68	0.14	0.1
CCV (stratified)	378.5	3.77	0.24	0.2
CV	370.0	4.46	0.06	6.4
CV (stratified)	379.3	4.53	0.08	7.5
Subsampling	361.7	12.25	0.01	1.2
Sub. (stratified)	372.5	12.23	0.02	1.6
1-Nearest-neighbor on *Abalone-3*				
Holdout	566.7	0.12	17.54	17.5
CCV	560.1	0.59	3.47	3.5
CCV (stratified)	561.1	0.59	6.91	6.9
CV	561.2	0.62	0.75	37.5
CV (stratified)	562.5	0.59	0.86	43.0
Subsampling	561.8	2.08	0.52	26.3
Sub. (stratified)	562.2	2.05	0.55	27.6
K-Nearest-neighbor (K=5) on *Abalone-2*				
Holdout	754.1	0.09	25.78	25.8
CCV	748.5	0.35	41.42	41.4
CCV (stratified)	749.3	0.35	21.33	21.3
CV	748.6	0.44	2.03	101.7
CV (stratified)	749.5	0.43	2.14	106.8
Subsampling	747.2	1.54	0.80	40.0
Sub. (stratified)	749.1	1.62	0.84	41.7
1-NN with Rank-scoring (R=10) on *Abalone-3*				
Holdout	970.7	0.04	26.16	26.2
CCV	973.6	0.02	20.95	21.0
CV	973.5	0.04	2.02	101.3
Subsampling	973.5	0.25	0.82	40.7

Table 3. Results of experiments using an asymmetric loss function: cost for missing a cancer case is set to be 5 times greater than the cost of a false alarm. The accuracy estimate produced by CCV is consistent with the standard methods and the execution time and variance compare very favorably.

Experiment	Exp. Loss	Var	Time (s) per trial	Time (s) per run
Holdout	225.4	4.87	0.47	0.5
CCV	246.1	6.89	0.30	0.3
CV	240.9	12.66	0.06	2.8
Subsampling	230.6	41.46	0.03	1.4

datasets, such as k-d trees (Friedman et al., 1977).

- Directly estimating higher-order statistics. This would enable us to report confidence intervals on the estimated accuracy.

Acknowledgments

Thanks to Rich Caruana, Dayne Freitag, Terence Sim, and Gita Sukthankar for valuable feedback on earlier drafts of this paper.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*, 21–27.

Dasarathy, B. (1991). *Nearest neighbor pattern classification techniques*. Los Alamitos, CA: IEEE Computer Society Press.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.

Dietterich, T. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*, 1895–1923.

Friedman, J., Bentley, J., & Finkel, R. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*, 209–226.

Jacobs, C., Finkelstein, A., & Salesin, D. (1995). Fast multiresolution image querying. *Proceedings of SIGGRAPH 95* (pp. 277–286).

Knuth, D. (1973). *The art of computer programming: Sorting and searching*. Reading, MA: Addison-Wesley.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of International Joint Conference on Artificial Intelligence* (pp. 1137–1143).

Lawrence, S., Giles, C., Tsoi, A., & Back, A. (1996). *Face recognition: A hybrid neural network approach* (Technical Report UMIACS-TR-96-16). University of Maryland.

Mangasarian, O., & Wolberg, W. (1990). Cancer diagnosis via linear programming. *SIAM News*, *23*, 1, 18.

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.

Phillips, P. (1998). Support vector machines applied to face recognition. *Advances in Neural Information Processing Systems 11* (pp. 803–809).

Phillips, P., Wechsler, H., Huang, J., & Rauss, P. (1998). The FERET database and evaluation procedure for face-recognition algorithms. *Image and Visual Computing*, *16*, 295–306.

Ross, S. (1988). *A first course in probability*. New York: Macmillan.

Salton, G. (1971). *The SMART information retrieval system: Experiments in automatic document processing*. Edgewood Cliffs, NJ: Prentice-Hall.

Samaria, F., & Harter, A. (1994). Parametrisation of a stochastic model for human face identification. *Proceedings of IEEE Workshop on Applications on Computer Vision* (pp. 138–142). ORL database is available at: `<www.cam-orl.co.uk/facedatabase.html>`.

Sim, T., Sukthankar, R., Mullin, M., & Baluja, S. (2000). Memory–based face recognition for visitor identification. *Proceedings of Face and Gesture* (in press).

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*, 71–86.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Waugh, S. (1995). *Extending and benchmarking cascade-correlation*. Doctoral dissertation, University of Tasmania.

Wiskott, L., Fellous, J., Krüger, N., & von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *Transactions on Pattern Analysis and Machine Intelligence*, *19*, 775–779.