On the Dangers of Cross-Validation. An Experimental Evaluation

R. Bharat Rao
IKM CKS Siemens Medical Solutions USA

Glenn Fung
IKM CKS Siemens Medical Solutions USA

Romer Rosales
IKM CKS Siemens Medical Solutions USA

Abstract

Cross validation allows models to be tested using the full training set by means of repeated resampling; thus, maximizing the total number of points used for testing and potentially, helping to protect against overfitting. Improvements in computational power, recent reductions in the (computational) cost of classification algorithms, and the development of closed-form solutions (for performing cross validation in certain classes of learning algorithms) makes it possible to test thousand or millions of variants of learning models on the data. Thus, it is now possible to calculate cross validation performance on a much larger number of tuned models than would have been possible otherwise. However, we empirically show how under such large number of models the risk for overfitting increases and the performance estimated by cross validation is no longer an effective estimate of generalization; hence, this paper provides an empirical reminder of the dangers of cross validation. We use a closed-form solution that makes this evaluation possible for the cross validation problem of interest. In addition, through extensive experiments we expose and discuss the effects of the overuse/misuse of cross validation in various aspects, including model selection, feature selection, and data dimensionality. This is illustrated on synthetic, benchmark, and real-world data sets.

1 Introduction

In a general classification problem, the goal is to learn a classifier that performs well on unseen data drawn from the same distribution as the available data [1]; in other words, to learn classifiers with good generalization. One common way to estimate generalization capabilities is to measure the performance of the learned classifier on test data that has not been used to train the classifier. When a large test data set cannot be held out or easily acquired, resampling methods, such as cross validation, are commonly used to estimate the generalization error. The resulting estimates of generalization can also be used for model selection by choosing from various possible classification algorithms (models) the one that has the lowest cross validation error (and hence the lowest expected generalization error).

A strong argument in favor of using cross validation is the potential of using the entire training set for testing (albeit not at once), creating the largest possible test set for a fixed training data set. Essentially, the classifier is trained on a subset of the training data set, and tested on the remainder. This process is repeated systematically so that all the points in the training set are tested. There has been much study on the empirical behavior of cross-validation for error estimation and model selection, and more recently theoretical bounds on the error in the leave-one-out cross-validation (LOOCV) estimate. Much of the focus has been on the expected value of this error over all training sets of a given sample size and the asymptotic behavior as the sample size increases. In this paper we empirically address the pitfalls of using cross validation error to select among a *large number* of classification algorithms.

Resampling methods, such as bootstrapping or cross validation (Stone, 1977; Kohavi, 1995a; Weiss & Kulikowski, 1991; Efron & Tibshirani, 1993) have typically been used to measure the generalization performance of a chosen algorithm, or possibly to select between a *limited* set of algorithms. Until the last decade, cross validation experiments could reasonable be performed only on a small set of algorithms or possible models; a k-fold or LOOCV run for a single algorithm, even on a small dataset, typically ran for several hours, if not days. As computers have become more powerful and due to recent advances regarding the computational efficiency of popular classification algorithms and techniques (for example: linear training time for SVMs (Joachims, 2006) and $n \log(n)$ kernel computation (Raykar & Duraiswami, 2005)), cross validation

[1] We concentrate on performance on data drawn from the same distribution but performance on a different distribution is also a (less explored) problem of interest.

performance can be quickly computed on several thousands or even millions of algorithms. Recent developments in grid computing now allow computers distributed in a large geographic area to be harnessed for a specific task, exponentially increasing the computing power at hand.

It is a commonly held believe that cross validation, like any other tool or metric, can be abused (Ng, 1997). Some basic heuristic procedures have been employed to avoid these problems. For example, when possible a sequestered test set is kept aside. This set is finally used only after training to verify that the chosen classifier indeed has superior generalization. Any modeling decisions based upon experiments on the training set, even cross validation estimates, are suspect, until independently verified.

Despite certain general knowledge about the drawbacks attached to cross validation, there has not been a sufficiently clear experimental (practical) investigation on the behavior of the estimate of generalization error for a fixed data set.

In this paper we provide an empirical reminder of a fact that is known but usually underestimated: when the set of algorithms to be considered becomes large, cross validation is no longer a good measure of generalization performance, and accordingly can no longer be used for algorithm or feature selection. In addition we experimentally show the impact of cross validation as the data dimensionality increases and for feature selection. We provide experimental results on synthetic, standardized benchmark (from the UCI repository), and a real-world dataset related to clinical diagnosis in virtual colonoscopy.

2 Related Research

A fundamental issue in machine learning is to obtain an accurate estimate of the generalization error of a model trained on a finite data set. Precisely estimating the accuracy of a model is not only important to examine the generalization performance of an algorithm, but also for choosing an algorithm from a variety of learning algorithms. Empirical estimators based upon *resampling*, which include bootstrap (Efron & Tibshirani, 1993), cross validation (Stone, 1977) estimates are popular, and *Holdout* estimates where a test set is sequestered until the model is frozen are also used.

A fair amount of research has focused on the empirical performance of leave-one-out cross validation (LOOCV) and k-fold CV on synthetic and benchmark data sets. Experiments by (Breiman & Spector, 1992) show that k-fold CV has better empirical performance than LOOCV for feature selection for linear regression. (Kohavi, 1995b) also obtains results in favor of 10-fold cross validation using decision trees and Naive Bayes, and demonstrates the bias-variance trade-off for different values of k on multiple benchmark data sets. (Kohavi & Wolpert, 1996) discuss the bias-variance trade-off for classifiers using a misclassification loss function. Our work, while not directly related to the bias-variance trade-off is closely related to the notion of variance.

From a theoretical perspective, the most general theoretical results for training error estimates are provided by (Vapnik, 1982) who proved that the training error estimate is less than $O(\sqrt{VC/n})$ away from the true test error where VC is the VC dimension of a hypothesis space. More recently, the task of developing upper bounds on the LOOCV error for a specific methodology has drawn the attention in the learning theory community. For example, (Zhang, 2003) has derived upper bounds on the expected LOOCV error to show consistency for a set of kernel methods. These consistency bounds focus on examining the uniform convergence to the infimum error risk for a given data distribution and a given methodology (Vapnik, 1998). Thus, they cannot help us investigate how close the leave-one-out estimate is to the true error when a large number of algorithms are trained on a fixed finite data set. Perhaps the theoretical work, closest to the empirical focus of this paper is proposed in (Kearns & Ron, 1997), where PAC-style bounds are provided on the difference between the LOOCV estimate and the true error; these bounds hold for any given deterministic algorithm and any sample of size n with relatively week assumption on error stability. However, it turns out that these kinds of bounds are not tight enough to guide our examination of the accuracy of the LOOCV estimate in practice.

3 Methodology

We begin by stating some needed preliminaries and defining the quantity, $\Omega(\mathcal{A})$, the *cross validation underestimate of true error* for a specific algorithm $\mathcal{A}$ (in the context of a fixed training set, S).

We then describe our methodology for randomly generating a large (potentially infinite) number of classification algorithms. To further speed up our experiments, we make use a closed-form solution for efficiently evaluating the cross-validation error for regularized least-squares methods for classification similar to the one proposed in (Cawley & Talbot, 2003) for kernel Fisher discriminants; this covers a very large family of commonly-used classification algorithms.

3.1 Definitions Let f be a target function from domain X to range Y, and let $\mathcal{D}$ be a distribution over X. Let $S =< \mathbf{x}_1, y_1 >, \ldots, < \mathbf{x}_n, y_n >$ be a training data set of n samples, where each $\mathbf{x}_i$ is a vector in $\Re^d$ drawn randomly and independently according to $\mathcal{D}$, and $y_i = f(\mathbf{x}_i) + \sigma$, where σ represents additive noise. For any index $i \in [n]$, denote by S^i the training set S with the i^{th} sample $< \mathbf{x}_i, y_i >$ removed.

For the classification problems we discuss here, y is the classification label taking values in $\{1, -1\}$, but many of the results in this section hold for both classification and regression problems, and also when $\mathbf{y}$ is a vector. Let $\mathcal{B}=\{\mathrm{A}_1, \ldots, \mathrm{A}_M\}$ be a collection of M learning models or algorithms for classification.

In leave-one-out cross validation, a classification algorithm $\mathcal{A}$ is trained on S^i to produce a hypothesis $h^i = \mathcal{A}(S^i)$ and the prediction $h^i(x_i)$ is compared with y_i. This process is repeated n times for all S^i. Then, $\hat{\epsilon}^{\mathcal{A}}_{cv}(S)$, the cross validation estimate of the error of the hypothesis $h = \mathcal{A}(S)$, is simply the fraction of samples in S that have been misclassified during the cross validation experiments. We are interested in Ω, the *cross validation underestimate of the true error*:

$$(3.1)\ \Omega(\mathcal{A}) = true\ err - CV\ err = \epsilon(\mathcal{A}(S)) - \hat{\epsilon}^{\mathcal{A}}_{cv}(S)$$

where $\epsilon(\mathcal{A}(S))$ is the *true generalization error* of the hypothesis $h = \mathcal{A}(S)$ over $\mathcal{D}$.

In addition to estimating generalization error, cross validation is used to select from a set of algorithms. Let $\mathcal{A}^{\star}(S) \in \mathcal{B}$ be the algorithm that minimizes the cross validation error on S. Then:

$$(3.2) \qquad \Omega^{\star} = \Omega(\mathcal{A}^{\star})$$

is the *under-estimate of the cross validation error of the selected algorithm*, i.e., the amount that the cross validation error of the selected algorithm, under-estimates the true error.

The above discussion can equally apply to k-fold cross validation. (For k-fold CV, the training data is randomly split into k mutually exclusive subsets, and each fold uses one of the k subsets as test and the remaining data for training. LOOCV is k-fold with $k = n$.) Note that whereas the LOOCV error, $\epsilon^{\mathcal{A}}_{cv}(S)$, is dependent only on S and $\mathcal{A}$, the k-fold CV error, $\epsilon^{\mathcal{A}}_{k-cv}(S)$ depends also upon the specific partitioning of S into k subsets.

3.2 Regularized least-squares for classification

We now describe the methods we use for generating a large number of classification algorithms, and state an extremely efficient closed-form solution for computing LOOCV, and in general for k-fold CV.

The experiments described in this paper use a family of regularized least squares methods (Ridge regression) for classification. Least squares methods are traditionally used for solving classification and regression problems (Johnson & Wichern, 2002). In the recent past, much research has been devoted to incorporating the concept of kernels and new forms of regularization in these methods (Saunders et al., 1998; Mika et al., 1999; Suykens & Vandewalle, 1999; Fung & Mangasarian, 2001).

Let $\mathbf{A} \in \Re^{n \times d}$ be the data matrix representing the training samples, S. Each row represents a sample and each column represents a feature. Notice that if we use the kernel extension of least squares methods, the features in $\mathbf{A}$ may either be original features in $\mathbf{x}$ or columns introduced by kernel matrices. We construct linear models f using the features $\mathbf{a}$ contained in $\mathbf{A}$, i.e., $f = \mathbf{a}^T\mathbf{w} + b$ where $\mathbf{w}$ is the weight vector of the linear function, and b is the offset. In some context, b can be incorporated into kernels and may not explicitly appear in the model.

Typically, in regularized least squares methods, we optimize an unconstrained quadratic program. For instance, in kernel ridge regression, the following objective function is minimized:

$$(3.3)\ \frac{1}{2}\left(\mathbf{y} - \mathbf{A}\mathbf{w} - b\mathbf{1}\right)^T\left(\mathbf{y} - \mathbf{A}\mathbf{w} - b\mathbf{1}\right) + \lambda\mathbf{w}^T\mathbf{w}.$$

where the first term calculates the least squares training error, and the second term is a regularization factor for capacity control and λ is called regularization parameter that needs to be tuned in training.

The optimal solution of a least squares method of the form in Equation (3.3) can be analytically calculated. In (Xu et al., 2001) it was shown that the optimal solution for a variety of least squares methods can be obtained by solving a set of linear equations as follows :

$$(3.4)\ \begin{pmatrix} \mathbf{A}\mathbf{A}^T + \lambda\mathbf{I} & \mathbf{A}\mathbf{1} \\ (\mathbf{A}\mathbf{1})^T & n \end{pmatrix}\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{A}^T\mathbf{y} \\ \mathbf{1}^T\mathbf{y} \end{pmatrix}$$

where $\mathbf{I}$ is an identity matrix of appropriate dimension, and $\mathbf{1}$ is the n-dimensional vector of ones. Now, in order to obtain Equation (3.3), we denote $\mathbf{Z} = (\mathbf{A}\ \ \mathbf{1})$, and $\mathbf{D} = diag([\lambda\mathbf{I}\ 0])$, then we have:

$$(3.5) \qquad \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} = \left(\mathbf{D} + \mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{y}.$$

In our experiments, we adopted RBF kernels $\exp(-\frac{||\mathbf{x}-\mathbf{c}||}{\sigma^2})$ where $\mathbf{x}$ is a feature vector in the original distribution, $\mathcal{D}$. For each given $\mathbf{c}$ which we call the center of a RBF function, a feature vector can be generated by calculating the RBF function over all training points $\mathbf{x}$. The RBF kernel matrix of dimensionality $n \times n$ is typically calculated for each pair of training examples.

However, for the experiments in this paper, we need to generate a large number of learning algorithms. We do so by ***randomly*** assigning K_{dim} RBF centers, and generate the matrix $\mathbf{A}$, an $n \times K_{dim}$ kernel matrix.

As discussed in Section 1, we evaluate our classification algorithms via LOOCV and k-fold CV. LOOCV is a special case of k-fold CV where $k = n$. Let $\mathbf{Z}_i$ be the test sub-matrix in the i^{th} fold of the cross validation containing n_i rows of matrix $\mathbf{Z}$, and let $\mathbf{Z}_{-i}$ be the training matrix formed by omitting $\mathbf{Z}_i$ from $\mathbf{Z}$. We derive a closed-form solution for each fold given the inverse of matrix $\mathbf{D} + \mathbf{Z}^T\mathbf{Z}$ by employing the following lemma (Golub & Loan, 1996):

Lemma 1 [A *Sherman-Morrison-Woodbury* Formula] Given the inverse matrix of $\mathbf{A} \in R^{n\times n}$ and $\mathbf{U} \in R^{n\times k}$,

$$\begin{aligned}&\left(\mathbf{A} + \mathbf{U}\mathbf{U}^T\right)^{-1} = \\ &\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{I} + \mathbf{U}^T\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^T\mathbf{A}^{-1}.\end{aligned}$$

In each fold, the solution depends on all training examples except the n_i test examples. Hence the solution in the i^{th} fold is

$$(3.6) \qquad \begin{pmatrix}\mathbf{w}_i \\ b_i\end{pmatrix} = \left(\mathbf{D} + \mathbf{Z}_{-i}^T\mathbf{Z}_{-i}\right)^{-1}\mathbf{Z}_{-i}^T\mathbf{y}_{-i}.$$

and $\mathbf{D} + \mathbf{Z}_{-i}^T\mathbf{Z}_{-i} = \mathbf{D} + \mathbf{Z}^T\mathbf{Z} - \mathbf{Z}_i^T\mathbf{Z}_i$. Denote $\mathbf{C} = \mathbf{D} + \mathbf{Z}^T\mathbf{Z}$, and applying Lemma 1 yields

$$(3.7) \qquad \begin{aligned}&(\mathbf{D} + \mathbf{Z}_{-i}^T\mathbf{Z}_{-i})^{-1} = \\ &\mathbf{C}^{-1} + \mathbf{C}^{-1}\mathbf{Z}_i^T\left(\mathbf{I} + \mathbf{Z}_i\mathbf{C}^{-1}\mathbf{Z}_i^T\right)^{-1}\mathbf{Z}_i\mathbf{C}^{-1}.\end{aligned}$$

Notice that in (3.7), we need to compute the inverse of an $n_i \times n_i$ matrix. In contrast, the inverse takes place on a $d \times d$ matrix as shown in equation (3.6). Hence when d is much larger than n_i, we gain a lot computational efficiency by using the closed-form formula. Especially in the LOOCV where $n_i = 1$, the closed form solution becomes

$$(3.8) \quad (\mathbf{D} + \mathbf{Z}_{-i}^T\mathbf{Z}_{-i})^{-1} = \mathbf{C}^{-1} + \frac{\mathbf{C}^{-1}\mathbf{Z}_i^T\mathbf{Z}_i\mathbf{C}^{-1}}{1 + \mathbf{Z}_i\mathbf{C}^{-1}\mathbf{Z}_i^T},$$

so there is no need to calculate any inverse in the closed-form formula of Equation (3.8) for LOOCV.

Figure 1 shows the distribution of cross validation accuracy for 1,000,000 distinct learning algorithms. For each of these 10^6 runs, we verified that the error as measured by the traditional iterative LOOCV computation is exactly equal to the error computed via Equation (3.8).

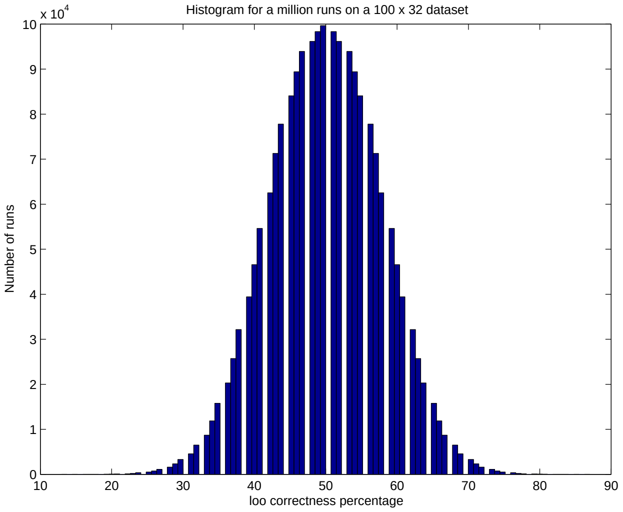

Figure 1: LOOCV accuracy distribution for 10^6 algorithms.

4 Experiments on Synthetic Data

4.1 Synthetic data for experiments Let ∇ be a distribution over $\Re^d$, specifically in $[-1, 1]^d$. Create training set S by drawing n training samples randomly and independently according to ∇. Let y, the class label be independent of the sample and be assigned randomly to 1 or -1 with probability $\frac{1}{2}$. Therefore, the *true error for every hypothesis* on ∇ is precisely 0.5. In other words, the under-estimate of the cross-validation error, Ω, depends only on the cross validation estimate, and we can modify Equation (3.1) to write

$$(4.9) \qquad \Omega_{\nabla}(\mathcal{A}) = 0.5 - \hat{\epsilon}_{cv}^{\mathcal{A}}(S) = a\hat{c}c_{cv} - 0.5$$

where, $a\hat{c}c_{cv} = 1 - CVerror$ is the *cross validation accuracy.* Similarly, $\Omega_{\nabla}^{\star}$ is the corresponding value for the algorithm with the lowest cross validation error..

4.2 Cross validation as an estimate for generalization We use the simple methodology shown below for the experiments in this sub-section.

- Given n,d, and r do:
    1. Sample n examples from synthetic distribution ∇ to create data set S of dimension d
    2. Compute LOOCV error for all M algorithms
    3. determine $\mathcal{A}^*$ with the minimum LOOCV error and the corresponding Ω^*(= LOOCV error + 0.5)
- Repeat steps (1-3) r times, and average Ω^* over r runs
- Report accuracy = 0.5+Ω^* = 1 - LOOCV error

Table 1: Variation of best LOOCV accuracy as the M the number of learning algorithms is increased.

M	10^1	10^2	10^3	10^4	10^5	10^6
Best CV	61.9	69.0	74.8	79.6	83.2	85.6
Expected	50	50	50	50	50	50

These experiments examine how good the cross validation estimate is of the true error when a large number of algorithms are executed on a single data set. As described in Section 3, we randomly generate centers for RBF kernels which allows us to quickly generate a virtually unlimited number of learning models. Essentially, we would hope that $\Omega^\star_\nabla$ would be close to 0, and the cross validation accuracy of the selected algorithm would be fairly close to 0.5, the true error. Note that, Ω, the CV under-estimate of true error is identical to the *CV over-estimate of true accuracy.*

Impact of number of learning algorithms

Figure 1 displays the distribution of CV accuracy for 1,000,000 distinct algorithms on a 100-sample 16-dimension synthetic data set. It is important to note the following surprisingly extreme disturbing result: the maximum CV accuracy achieved is in excess of 85%. (For this experiment, $\Omega^\star_\nabla = 0.35$, corresponds to the amount that the CV accuracy of the chosen algorithm *over-estimates the true accuracy.*)

Table 1 reports the best cross validation accuracy (corresponding to $\Omega^\star_\nabla$) as we systematically increase the number of algorithms we try on a single data set. Accuracy values shown are the average of $r = 500$ trials of M algorithms, except for $M = 10^5$ ($r = 10$) and $M = 10^6$ ($r = 1$). The CV over-estimate for much smaller numbers of learning algorithms is a caution for blindly executing a relatively large number (e.g. $M = 1000$) of algorithms and simply selecting the best.

Impact of data size and dimensionality Figure 2 displays the best cross validation accuracy as the dimensionality, d, of the training set is increased. The 4 curves correspond to executing $M = 1000$ algorithms each of training sets of size 100 and 1000 respectively, and with using a fixed ($K_{dim} = 20$) or varying kernel ($K_{dim} = d$) dimensionality. As expected, the CV over-estimate of accuracy is most pronounced for small data sets with large dimensionality. This phenomenon has been widely studied and it is usually referred to as the "curse of dimensionality".

It is generally accepted that increasing model complexity increases the likelihood of overfitting the data; it is interesting to note that the same applies to overfitting, as it were in the "cross validation space". Note that for the varying kernel dimensionality curves, the kernel dimensionality is less than the fixed amount (20) for $d = 4, 8$. All estimates are averaged over $r = 10$ trials.

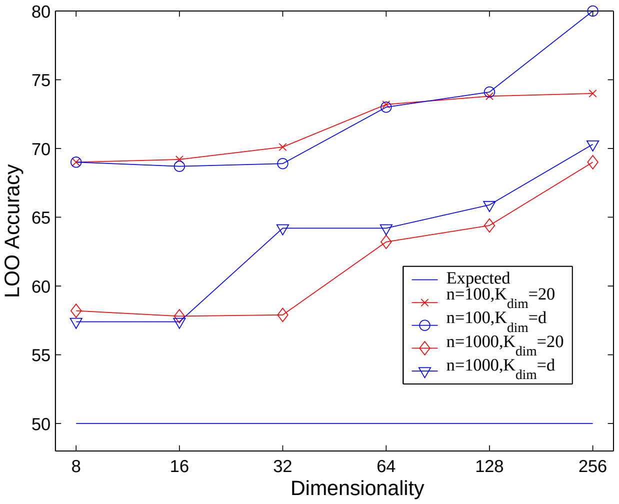

Figure 2: Variation of best CV accuracy for $M = 1000$ algorithms with dimensionality, sample size, and kernel dimensionality.

4.3 Cross validation for feature selection

For much the same reasons that cross validation is a good estimate of generalization performance, it is also believed to be a good method for selecting a subset of features (Breiman & Spector, 1992). In these series of experiments, we demonstrate that feature selection is also unreliable when a large number of features are selected using cross validation. These results empirically confirm and relate to statements made by previous work (Ng, 1998; Reunanen, 2003; Reunanen, 2004).

Our experimental methodology is completely different from that described in Section 4.2. Since we are interested in feature selection, it makes no sense to use random kernels; instead we use linear models (Fisher's linear discriminant (Mika et al., 1999)) for classification. As advocated in (Kohavi & John, 1997), we use the widely used greedy "wrapper" method for feature selection, wherein the learning algorithm is run iteratively on the training data to select the feature that minimizes the LOOCV error, $\hat{\epsilon}_{cv}$. This greedy feature selection continues until the $\hat{\epsilon}_{cv}$ converges. Therefore, with "wrapper" feature selection, the classifier itself determines the relevant features.

In our experiments, we wish to draw n samples from ∇ with dimensionality d, and evaluate the impact on Ω

when we select *exactly* δ out of d features. Thus, instead of continuing until convergence as suggested above, we either stop after precisely $\delta \leq d$ features are selected, or if $\hat{\epsilon}_{cv}$ has already converged, force the selection of the feature that least increases $\hat{\epsilon}_{cv}$. We repeat this r times with a different data set each time, and report the average value of Ω.

Note that $M = 1$ for these experiments. However, we could think of every linear model depending on every possible subset of the original features as a different model. Unlike the previous experiments where we were observing the behavior of $\Omega^\star_\bigtriangledown$, here we evaluate the behavior of $\Omega_\bigtriangledown$ itself. The average value of $\Omega_\bigtriangledown$ that we obtain gives some intuition into exactly how much we can *under-estimate* the true error in feature selection.

Impact of number of features selected In these experiments, we randomly draw n samples with dimensionality $d = 256$ from $\bigtriangledown$ and select exactly δ features out of these 256. Table 2 shows that for a small data set, cross validation accuracy $(0.5 + \Omega_\bigtriangledown)$ continues to increase even after selecting 32 redundant (by definition) features. However, for large sample sizes ($n = 10000$) forcing the classifier to use additional features makes very little difference after a very small number of features have been selected.

Impact of number of features to select from One of the standard defenses against overfitting is to permit only a small number of features in our model – this is the basic motivation for feature selection. In these experiments, we select only the best $\delta = 4$ features, but allow the greedy feature selection access to an increasingly large pool of features to select from. Table 3 shows that allowing access to more and more features, dramatically overfits the training set for small sample sizes ($n = 100$), even when cross validation is used for feature selection and the classifier is restricted to a very small number of features. See Section 7 for a discussion about our experiences with a real-world data set where we had similar experiences.

Table 2: Variation of $acc_{cv} = 0.5 + \Omega_\bigtriangledown$ with n and greedy selection of δ =4, 8, 16, 32 features from $d = 128$ features. For each experiment, the mean (upper) and standard deviation (lower) of acc_{cv}. For $n = 100$, $\Omega_\bigtriangledown$ is averaged over $r = 1000$ trials, and for all other n's over $r = 100$ trials. Note that $M = 1$.

N/δ	4	8	16	32
100	70.5	72.2	72.7	76
	2.9	4.9	5.4	6.4
1000	60.7	61.2	59.6	56.3
	5	4	3	3.6
10000	59.7	60.1	60.1	59.5
	1.3	1.5	1.5	1.7

Table 3: Variation of $acc_{cv} = 0.5{+}\Omega_\bigtriangledown$ with n and greedy selection of $\delta = 4$ features from d =8,16,32, 64, and 128 features. For each experiment, the mean (upper) and standard deviation (lower) of acc_{cv}. For $n = 100$, $\Omega_\bigtriangledown$ is averaged over $r = 1000$ trials, and for all other n's over $r = 100$ trials. Note that $M = 1$.

N/δ	8	16	32	64	128
100	59.2	63.3	66.5	68.8	70.6
	4.6	3.9	3.7	3.6	2.8
1000	53.3	54.7	56.2	57.8	60.3
	1.7	1.6	2.5	3.2	4.5
10000	51.9	53.3	55.5	57.1	59.3
	1.8	2.6	2.5	2.2	1.4

5 Experimental Results on Benchmark data sets

For synthetic data sets, we can usually compute the *true error*, $\epsilon(h)$ of a hypothesis h, either analytically (as shown above) or accurately estimate $\epsilon(h)$ by generating a large test dataset). Thus, for synthetic data, we can compute the value of Ω from Equation (3.1) for any $\mathcal{A}_i$ and S_n, (and accordingly determine, $\mathcal{A}^\star$, $\mathcal{A}^{max}$ and compute the values of $\Omega^\star$ and Ω^{max}).

Unfortunately, the distribution is unknown for real-world data sets, and all we observe is $\hat{\epsilon}_{cv}^{\mathcal{A}_j}(S_n)$ (and so can determine $\mathcal{A}^\star$ and $\hat{\epsilon}_{cv}^{\mathcal{A}^\star}(S_n)$). However, if we can draw m additional samples randomly and independently from D to create a hold out test set, T_m, then we can *estimate* the value of Ω (from Equation (3.1)) as:

$$\hat{\Omega}(\mathcal{A}_|) = \hat{\epsilon}^{\mathcal{A}_j}(S_n)(T_m) - \epsilon_{cv}^{\mathcal{A}_j}(S_n) \tag{5.10}$$

where $\hat{\epsilon}^h(T)$ is the *misclassification error* of the hypothesis h on a data set T (i.e., number of incorrect predictions by h on T). We can also modify Equations (3.2) and (3.2) accordingly:

$$\hat{\Omega}^\star = \hat{\Omega}(\hat{\mathcal{A}}^\star) \tag{5.11}$$

where, $\hat{\mathcal{A}}^\star = \text{argmin} \ldots.$

$$\hat{\Omega}^{max} = \hat{\Omega}(\hat{\mathcal{A}}^{max}) = \max \hat{\Omega}(\mathcal{A}_j) \tag{5.12}$$

We investigate the efficacy of cross validation as an estimate for generalization when a large number of algorithms are tried on a single data set, but this time on real instead of synthetic data; on 5 benchmark data sets from the UCI repository (Newman & Merz, 1998). We split each benchmark data set Q into a training set S_n, and a test data set T_m, such that $n = m = N/2$, where N is the number of samples in Q; we do a stratified split,

Table 4: $\hat{\Omega}^\star$ is the estimate of Ω for the algorithms with the best $\hat{\epsilon}_{cv}$ for M algorithms. The first two rows provide the sample size and dimensionality for the 5 benchmark data sets. The first column (or the remaining rows) provides M and r. are for different values of M/r In each cell, the top number is $\hat{\Omega}^\star$ and the bottom number is $\hat{\Omega}^{max}$ (averaged over the r runs).

DATA	BUPA	CLEVE	BOSTON	PIMA	WPBC60
n	345	297	506	728	110
d	6	13	13	8	32
	5.0	4.5	3.5	10.0	13.0
2000/1	9.7	12.7	7.4	13.6	20.4
	4.98	4.02	3.4	9.8	13.0
1000/5	9.7	11.7	7.4	13.4	22.4
	4.97	3.94	3.21	9.9	12.2
500/5	9.67	11.5	6.8	14.1	22.3
	4.87	3.48	2.98	9.37	9.89
100/10	10.7	9.13	6.48	14.6	23.2

so that the ratio of positive and negative examples is the same in S_n and T_m. For $\mathcal{A}_j \in\mathcal{B}$ (generated as described in Section 3), we compute $\hat{\epsilon}_{cv}$ on S_n, and the error of hypothesis $h = \mathcal{A}_j(S_n)$ on T_m to compute $\hat{\Omega}(\mathcal{A}_|)$ (from Equation (5.10)); compute $\hat{\Omega}^\star$ from Equation (5.11). We repeat these experiments $r = 10$ times, and report the value of $\hat{\Omega}^\star$ averaged over these r trials.

Table 4 shows the value of $\hat{\Omega}^\star$, the *estimated underestimate of the CV error* for $\mathcal{A}^\star \in\mathcal{B}$ that minimizes $\hat{\epsilon}_{cv}$. This value is of particular significance, because is represents our estimate of Ω for the algorithm we would pick, if S_n were our training set. We also show the value of $\hat{\Omega}^{max}$ (also averaged over r trials) which shows how badly the worst estimates of Ω can be off (the highest values of $\hat{\Omega}^{max}$ we achieved were about 1.5 to 2 times higher than the average $\hat{\Omega}^{max}$ values shown in Table 4). This is of academic interest only, as in real life we would have no interest in $\mathcal{A}^{max}$, only in $\mathcal{A}^\star$.

6 Experimental Results on a real world data set: The ColonCAD dataset

These results also match our experiences on a real-world data set. Recently, we developed classifiers to detect colon polyps from computed tomography images of the colon (also called virtual colonoscopy) (Fung et al., 2006). For the system to be deemed clinically useful (both in a clinical trial and also in the opinions of physicians), high sensitivity is critical while maintaining a low number of false positives per patient (usually no more than 3 or 4 is clinically acceptable). We used a simple shape filter to identify candidates, from which we extracted 66 features in all based on moments of tissue intensity, volumetric and surface shape and texture characteristics. It is important to note that all features were based (at least loosely) on visual characteristics of colon polyps based on the advice of experts. Using a variety of techniques to prevent overfitting, including LOOCV, we achieved acceptable LOOCV performance (94% sensitivity, 4 false positives / image) with a classifier that used just 14/66 features; more importantly, experiments on newly collected data sets showed virtually identical performance.

In an attempt to further boost performance, we extracted an additional 20 features (66 in all), which perhaps were not quite as motivated by medical factors, but on extracting certain statistics from the candidates. Using these features in feature selection to build a new classifier that also coincidentally had 14 features, we dramatically improved LOOCV performance, now obtaining 100% sensitivity with similar false positive rate; this was particularly significant, because we had never before attained perfect sensitivity with any classifier without a huge false positive rate. However, performance on test sets was significantly poorer. Essentially, adding those 20 new features into the training set, resulted in the same overfitting shown on synthetic data in Table 3.

7 Discussion

We have demonstrated, using controlled numerical experiments, that when the number of algorithms is large, LOOCV ceases to be an effective estimate of generalization for the algorithm that has the best cross validation performance. This is because running a large number of algorithms effectively overfits in cross validation space. Experiments on synthetic data demonstrate, as expected, that this behavior worsens as the sample size decreases, and the dimensionality and number of algorithms increase. The phenomenon of underestimating cross validation error is also demonstrated on some benchmark data sets, and is seen to be worse for datasets with the higher dimensionality. In addition, we analyzed a real clinical dataset and observed consistent behavior.

We also investigated the use of LOOCV for feature selection. As expected, as we select more features, LOOCV grossly underestimates true error and could lead to selecting unnecessary features; this phenomenon is well known, and many researchers guard against this by using a test set and limiting the number of selected features. The other experiments on feature selection show that as we increase the number of available features, restricting the number of features to a very small number (4 in the experiments shown in Table 3) does not guard against overfitting.

While the general problems with cross validation have been noticed in the past, we believe this paper offers, through a careful numerical evaluation, a much needed guidance on how this happens in synthetic and real data and on the severity of the problem in general (including tasks like feature selection)

Several reasons can be attached to the behavior demonstrated in this paper. In general, one inherent limitation of the various resampling approaches is that not all data can be used for training at once. The amount of available training data is in general an important factor affecting the performance of the learned classifier; having more data available for training is in general beneficial. Similarly, the amount of data used for testing (data not used for training) can have a major effect on the accuracy of our estimates of the generalization error. Small amounts of test data will generally result in a higher variance in the estimate. In the case of LOOCV, while the amount of training data used to obtain each algorithm is large relative to the total data, only one data point is used for testing at a time. This in general will imply a large variance in the cross validation error; as a consequence generating unreliable estimates.

Although, this requires much further study, we offer a few heuristics to guide the user. Whenever possible use a sequestered test set that is only used when the classifier has been frozen. Ideally, the performance on the test set will be very similar to the cross validation error, i.e., $\hat{\Omega} \rightarrow 0$. Some foresight should be used, particularly in selecting models that perform extraordinarily well (even when measured by cross validation) on the training set. Furthermore, when using cross validation for feature selection, the number of original features should be limited, and new features should be added with care. This is particularly important for small data sets. Similarly, it is important to be aware of the dangers of fitting a very large number of algorithms on a single data set.

A final note of warning: experienced machine learning researchers know not to tune a classifier by continuously observing the classifier performance on the test data until a desirable performance is achieved. When a classifier is tuned according to its performance on the test data, then the test results lose all their credibility since the classifier may no longer simulate real-world settings. More importantly, such a classifier loses its ability to generalize on new data, which is the key reasons to use cross validation in the first place.

8 Acknowlegments

Many people have contributed to this tis paper, in a variety of ways. We would like to thank the following persons: Jinbo Bi, Murat Dundar, Sathyakama Sandilya.

References

Breiman, L., & Spector, P. (1992). Submodel selection and evaluation in regression: The x-random case. *International Statistical Review*, *60*, 291–319.

Cawley, G. C., & Talbot, N. L. C. (2003). Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. citeseer.ist.psu.edu/cawley03efficient.html.

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. London, UK: Chapman and Hall.

Fung, G., Dundar, M., Krishnapuram, B., & Rao, R. B. (2006). Multiple instance learning for computer aided diagnosis. *NIPS 2006: Advances in Neural Information Processing Systems*.

Fung, G., & Mangasarian, O. L. (2001). Proximal support vector machine classifiers. *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA* (pp. 77–86). New York: Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations*. Baltimore: The Johns Hopkins University Press.

Joachims, T. (2006). Training linear svms in linear time. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 217–226). New York, NY, USA: ACM Press.

Johnson, R. A., & Wichern, D. W. (2002). *Applied multivariate statistical analysis*. New Jercey: Prentice Hall.

Kearns, M. J., & Ron, D. (1997). Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Computational Learing Theory* (pp. 152–162).

Kohavi, R. (1995a). A study of cross-validation and bootstrap for accuracy estimation and model selection. *IJCAI* (pp. 1137–1145).

Kohavi, R. (1995b). A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence IJCAI* (pp. 1137–1145).

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 275–283). Morgan Kaufmann.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., & Müller, K.-R. (1999). Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing IX* (pp. 41–48). IEEE.

Newman, C. B. D., & Merz, C. (1998). UCI repository of machine learning databases.

Ng, A. Y. (1997). Preventing overfitting of cross-validation data. *Proc. 14th International Conference on Machine Learning* (pp. 245–253). Morgan Kaufmann.

Ng, A. Y. (1998). On feature selection: learning with exponentially many irrelevant features as training examples. *Proc. 15th International Conf. on Machine Learning* (pp. 404–412). Morgan Kaufmann, San Francisco, CA.

Raykar, V. C., & Duraiswami, R. (2005). The improved fast gauss transform with applications to machine learning. *NIPS 2005 workshop on Large scale kernel machines.*

Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, *3*, 1371–1382.

Reunanen, J. (2004). A pitfall in determining the optimal feature subset size. *Proc. of the 4th Int. Workshop on Pattern Recognition in Information Systems (PRIS 2004)* (pp. 176–185). Porto, Portugal.

Saunders, G., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables. *Proc. 15th International Conf. on Machine Learning* (pp. 515–521). Morgan Kaufmann, San Francisco, CA.

Stone, M. (1977). Asymptotics for and against cross-validation. *Biometrika*, *64*, 29–35.

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*, 293–300.

Vapnik, V. N. (1982). *Estimation of dependencies based on empirical data.* New York: Springer-Verlag.

Vapnik, V. N. (1998). *Statistical learning theory.* New York: John Wiley & Sons, Inc.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn.* Morgan Kaufmann.

Xu, J., Zhang, X., & Li, Y. (2001). Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR. *Proc. of IJCNN-01* (pp. 1486–1491).

Zhang, T. (2003). Leave-one-out bounds for kernel methods. *Neural Computation*, *15*, 1397–1437.