Preventing “Overfitting” of Cross-Validation Data

Andrew Y. Ng
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
Andrew.Ng@cs.cmu.edu

Abstract

Suppose that, for a learning task, we have to select one hypothesis out of a set of hypotheses (that may, for example, have been generated by multiple applications of a randomized learning algorithm). A common approach is to evaluate each hypothesis in the set on some previously unseen cross-validation data, and then to select the hypothesis that had the lowest cross-validation error. But when the cross-validation data is partially corrupted such as by noise, and if the set of hypotheses we are selecting from is large, then “folklore” also warns about “overfitting” the cross-validation data [Klockars and Sax, 1986, Tukey, 1949, Tukey, 1953]. In this paper, we explain how this “overfitting” really occurs, and show the surprising result that it can be overcome by selecting a hypothesis with a *higher* cross-validation error, over others with lower cross-validation errors. We give reasons for not selecting the hypothesis with the lowest cross-validation error, and propose a new algorithm, LOOCVCV, that uses a computationally efficient form of leave–one–out cross-validation to select such a hypothesis. Finally, we present experimental results for one domain, that show LOOCVCV consistently beating picking the hypothesis with the lowest cross-validation error, even when using reasonably large cross-validation sets.

1 Introduction

A basic problem in learning is that of selecting a hypothesis out of a set of hypotheses – that is, determining which in a given set of hypotheses is “best.” When given such a set of hypotheses (that was, for example, generated by multiple applications of a randomized learning algorithm to a set of training data, or equivalently, applying a deterministic algorithm multiple times but using randomly re-chosen parameters each time), a common approach is to test all of them on some set of previously unseen cross-validation (CV) data, and then to pick the hypothesis that had the smallest CV error [Mosteller and Tukey, 1968, Stone, 1974, Stone, 1977]. For example, we may generate a set of hypotheses by training multiple Neural Networks with Backpropagation on a fixed set of training data, with each Neural Network having different initial weights, and then pick the Neural Network that has the lowest error on a set of hold-out CV data. But if the CV data is partially corrupted by noise, then “folklore” also warns that if we used too small a set of CV data to test too large a number of hypotheses, we may end up picking a poor hypothesis that had fit the corrupted CV data well “just by chance” [Klockars and Sax, 1986, Tukey, 1949, Tukey, 1953].

In this paper, we examine this problem of “overfitting” of CV data. This is an unusual form of overfitting because, unlike overfitting by single applications of learning algorithms such as Decision Trees and Neural Networks, *this overfitting comes from having tested too many hypotheses, rather than from having chosen a too complex a hypothesis.* For example, within the set of hypotheses we are selecting from, there is no notion of a structure or a sequence of nested hypothesis classes of increasing complexity such as assumed in some models [Kearns, 1996, Vapnik and Chervonenkis, 1971], or of some hypotheses having been trained using a more complex hypothesis class. Because of this, even thought the literature treating overfitting is rich with algorithms for selecting/limiting the *complexity* of the output hypothesis, this literature does not generalize to this other important problem of “overfitting” through having tested too *many* hypotheses with too small a CV set. (The mentioned literature is too wide to survey here, but for a sample, see [Baum and Haussler, 1989, Judd, 1990, Kearns et al., 1995, Miller, 1990, Rissanen, 1978].) And, while there are interesting similarities between "too many" and "too complex," the relationship between them is not obvious, and we defer a further discussion of them to Section 8, when we will have developed a notation for addressing these issues.

This paper examines this problem of "overfitting" of cross-validation data. Our contributions are two-fold: First, we explain how this overfitting really occurs, and show the surprising result that we can do better than picking the hypothesis with the lowest CV error. Note that this is a much *stronger* result than merely that the hypothesis with the lowest CV error is unlikely to be the hypothesis with the lowest generalization error; we are claiming that it is sometimes possible to actually *find* a hypothesis that we can expect to have lower generalization error than the minimum-CV-error hypothesis. And second, we will present an algorithm, *Leave-one-out Cross-Validated Cross-Validation* (LOOCVCV), to choose such a hypothesis, and show experimentally that LOOCVCV does consistently beat picking the minimum–CV–error hypothesis in one domain.

2 Definitions

Henceforth, for simplicity, let us assume a boolean target function f. Let $H = \{h_i\}_{i=1}^n$ be the set of n hypotheses that we are choosing from, and let us also assume that each hypothesis was drawn *i.i.d.* from some fixed distribution of hypotheses D_H (possibly a distribution induced by application of a randomized learning algorithm). We will also consider the case when we are also given access to an oracle that samples $h \in D_H$, but will not distinguish between this and the case where we do not have access such an oracle, because of the two cases' vast similarities. For example, when n is large, statements pertaining to when we have access to an oracle can naturally be approximately translated to when we do not, by "constructing" our own oracle that samples, possibly with replacement, from H. Continuing, let $S = \{(x_i, y_i)\}_{i=1}^m$ be the set of m samples forming the CV data, where each x_i is an arbitrary-dimensioned input vector drawn *i.i.d.* from some distribution D, and $y_i \in \{0, 1\}$, the labels of the data, have independently been corrupted with some fixed but unknown *noise rate* η, so that $y_i = f(x_i)$ with probability $1-\eta$, and $y_i = \neg f(x_i)$ with probability η. We shall also write $p_i = (x_i, y_i)$, so that $S = \{p_i\}_{i=1}^m$, and let us also define $S_{-i} = S \setminus \{p_i\}$. Furthermore, we shall say that hypothesis h_i *apparently misclassifies* some element of the CV set (x_j, y_j) if $h_i(x_j) \neq y_j$, and that it *truly misclassifies* it if $h_i(x_j) \neq f(x_j)$. Let the CV error $\hat{\varepsilon}_S(h_i) \in [0, 1]$ be the fraction of the CV set S that hypothesis h_i apparently misclassifies, and let the generalization error (with respect to uncorrupted data) of each hypothesis h_i be $\varepsilon_{D,f}(h_i) = \Pr_{x \in D}[h_i(x) \neq f(x)]$, following the PAC framework (although we will, for notational brevity, often drop the subscripts in $\hat{\varepsilon}_S$ and $\varepsilon_{D,f}$). Finally, the hypothesis h_i in a set H of size n that has the lowest CV error $\hat{\varepsilon}_S(h_i)$ will be called the "best–of–n hypothesis."

Note that by CV, we mean what is sometimes called "simple cross-validation," where there is a fixed cross-validation set, rather than m-fold or leave-one-out cross-validation. And in our notation, the goal of hypothesis selection, informally, is then to choose a hypothesis h such that $\varepsilon_{D,f}(h)$ is expected to be "small," and the conventional approach is choosing the best–of–n hypothesis.

3 Overfitting by a Simulated Learning Algorithm

As mentioned, when we choose the hypothesis h that has the lowest $\hat{\varepsilon}(h)$, we may be choosing a poor hypothesis that had fit the CV data well "just by chance." For example, if $\hat{\varepsilon}(h) = 0$ so that it apparently classifies the CV data perfectly, but some of the CV data's labels had been corrupted, then the hypothesis must actually be *truly misclassifying* the corrupted samples in the CV data, and therefore have non-zero (and possibly large) generalization error.

3.1 A Simulated Learning Algorithm

To examine the overfitting effect, let us first look at a particular model of a learning algorithm and learning task:

- We have a black box that draws successive hypotheses h_i *i.i.d.* from D_H (say, by applying a randomized algorithm to a set of training data), such that the random variable $\varepsilon_{D,f}(h_i)$ (induced by drawing $h_i \in D_H$), is distributed $\varepsilon_{D,f}(h_i) \sim \text{Uniform}(0, 1)$.
- We fix S to have $m = 100$ CV samples, and assume that our noise process happened to corrupt exactly 20 of this set's labels. (Note that, in this example, it is only to make our exposition cleaner that we have "fixed" the noise, which should technically be random.)
- Each hypothesis h_i has an *independent* $\varepsilon(h_i)$ chance of truly misclassifying each of the m CV samples, where the independence is across both hypotheses and CV samples.

So, to simulate testing a hypothesis with a given generalization error $\varepsilon(h_i)$, we would flip a 0–1 coin with bias $\varepsilon(h_i)$ 100 times, negate the result of (say) the last 20 flips, and calculate the fraction of resulting '1's to get the CV error $\hat{\varepsilon}(h_i)$. Note also that of the assumptions

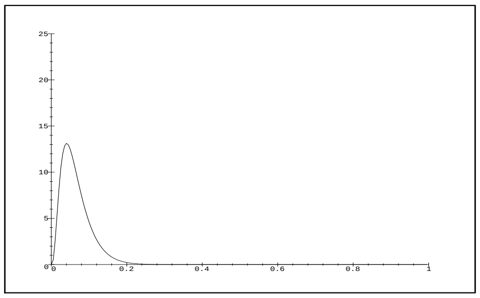

Figure 1: posterior distribution for $\varepsilon(h)|\hat{\varepsilon}(h) = 0.17$. $\mathrm{E}_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = 0.17] \approx 0.0648$.

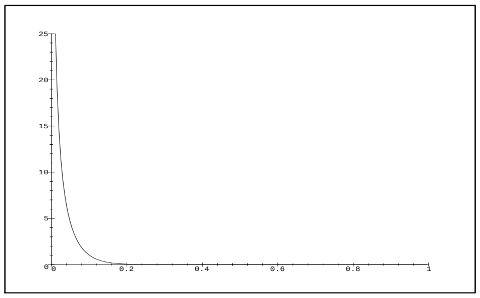

Figure 2: posterior distribution for $\varepsilon(h)|\hat{\varepsilon}(h) = 0.20$. $\mathrm{E}_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = 0.20] \approx 0.0252$.

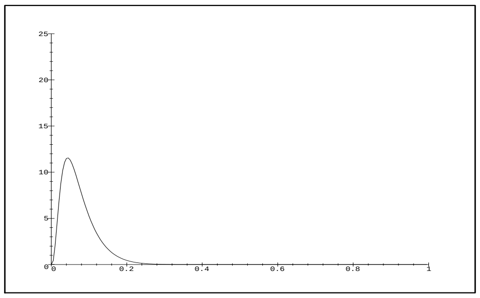

Figure 3: posterior distribution for $\varepsilon(h)|\hat{\varepsilon}(h) = 0.23$. $\mathrm{E}_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = 0.23] \approx 0.0727$.

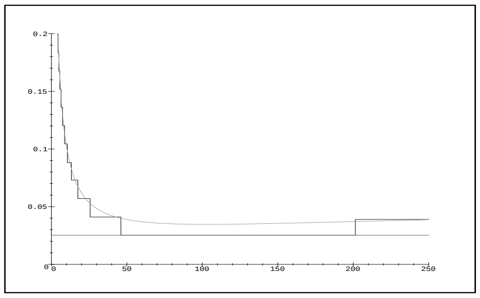

Figure 4: n on x-axis, generalization error on y-axis. Top curve is best–of–n, middle, stepped, line is k-th percentile hypothesis, with $k = 100(1 - (1/(n+1)))$, bottom line is LOOCVCV.

above, the one of uncorrelated errors across hypotheses is probably the most unrealistic, but we include it to have a simple and analyzable example of a learning algorithm. The goal, then, is to generate some number of hypotheses, test them on the 100-sample CV data, and then to somehow pick a hypothesis h with "small" expected generalization error.

3.2 Why Picking the "Best" is Bad

In this section, we will explain why picking the "apparent best" hypothesis may not be optimal when the CV data's labels have been corrupted by noise, and this will lead to results that will be the cornerstone of the rest of this work. While, for the purpose of clarity in exposition, we will continue to assume that exactly 20 CV data labels had been corrupted, it is worth keeping in mind that our main results will readily apply for the same problem using any other fixed CV sample S with some corrupted labels, whatever the level of label corruption may be. Naturally, the algorithms we propose later in this paper will also be applicable without requiring knowledge of the level of label corruption in S.

To motivate the rest of this discussion, let us first consider a simpler distribution of hypotheses than described above, that clearly shows why one might not want to pick the "apparent best" hypothesis. Let us continue to assume that exactly 20 of the 100 CV data labels were corrupted, but that D_H is now such that each hypothesis $h \in D_H$ has either $\varepsilon(h) = 0.00$ or $\varepsilon(h) = 0.10$. Now, consider the particular case of choosing between two hypotheses h_1 and h_2, where $\hat{\varepsilon}(h_1) = 0.17$, and $\hat{\varepsilon}(h_2) = 0.20$. Suppose we chose h_1. Then, we know that we must necessarily have chosen a hypothesis with $\varepsilon(h) = 0.10$, because if $\varepsilon(h) = 0.00$, then h would have classified all of the CV sample correctly with respect to the true target function, and would therefore have had $\hat{\varepsilon}(h) = 0.20$ because 20 elements of the CV sample were corrupted. On the other hand, if we chose h_2 with $\hat{\varepsilon}(h_2) = 0.20$, then we have a non-zero chance of picking a hypothesis with $\varepsilon(h) = 0.00$. Hence, $E_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = 0.20] < E_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = 0.17]$ for this case.

The previous example might have seemed slightly contrived, but a more general explanation for the hypothesis with lower $\hat{\varepsilon}(h)$ not having lower expected generalization error is in the differing *variances* of $\hat{\varepsilon}(h)$. Given S, D, D_H and f, then $\hat{\varepsilon}_S(h)$, as a random variable dependent with $\varepsilon(h)$ (which is in turn induced by drawing $h \in D_H$), is such that $\mathrm{E}_{h \in D_H}[\hat{\varepsilon}_S(h)|\,\varepsilon(h)]$ monotonically increases with $\varepsilon(h)$

as expected; but, its variance $\mathrm{Var}_{h\in D_H}(\hat{\varepsilon}(h)|\,\varepsilon(h))$ also increases as a function of $\varepsilon(h)$, at least for $\varepsilon(h) \in [0, 0.5]$. For example, if $\varepsilon(h) = 0$, then $\hat{\varepsilon}(h) = 0.20$ always, so that $\mathrm{Var}_{h\in D_H}(\hat{\varepsilon}(h)|\,\varepsilon(h) = 0) = 0$; but $\mathrm{Var}_{h\in D_H}(\hat{\varepsilon}(h)|\,\varepsilon(h) = 0.5) = 0.25 > 0$. So, despite hypotheses with lower $\varepsilon(h)$'s having lower $\mathrm{E}_{h\in D_H}[\hat{\varepsilon}(h)|\varepsilon(h)]$'s, it is possible for them to have have a *smaller* probability of having "extremely low" $\hat{\varepsilon}(h)$'s "by chance," because their distributions of $\hat{\varepsilon}(h)$ are less spread-out (lower variance). In words, *hypotheses with lower generalization errors do have lower expected CV errors, but may have a smaller chance of having "extremely low" CV errors because their distributions of CV errors are less spread–out than those of hypotheses with higher generalization errors (and higher expected CV errors).* And for certain "priors" such as the simplified D_H above, this leads to the effect of low $\overline{\hat{\varepsilon}} \not\Rightarrow$ low $\mathrm{E}_{h\in D_H}[\varepsilon(h)|\,\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}]$, which causes the overfitting phenomenon.

Finally, let us return to the model of Section 3.1. Recall that exactly 20 of the 100 CV data labels were corrupted. Then, using our "prior" for $h \in D_H$ that is such that $\varepsilon_{D,f}(h) \sim \mathrm{Uniform}(0, 1)$, we can calculate, for each value of $\overline{\hat{\varepsilon}}$, the posterior distribution $\varepsilon(h)|\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}$, which we denote by $f_{\varepsilon|\hat{\varepsilon}}$:

$$\begin{aligned}
f_{\varepsilon|\hat{\varepsilon}}(\varepsilon(h)|\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}) &= \frac{f_{\hat{\varepsilon}|\varepsilon}(\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}|\varepsilon(h))f_{\varepsilon}(\varepsilon(h))}{f_{\hat{\varepsilon}}(\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}})} \\
&= k f_{\hat{\varepsilon}|\varepsilon}(\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}|\varepsilon(h)) \\
&= k \sum_{i=0}^{100\overline{\hat{\varepsilon}}} (B(80, \varepsilon(h), i) \\
&\qquad B(20, 1 - \varepsilon(h), 100\overline{\hat{\varepsilon}} - i))
\end{aligned}$$

where k is a normalizing constant, and $B(n,p,x) = \binom{n}{x} p^x (1-p)^{n-x}$ is the probability of a $\mathrm{Binomial}(n, p)$ distribution returning x (so that i in the summation is the number of apparent misclassifications that are also true misclassifications). These posterior distributions of $\varepsilon(h)$ for $\overline{\hat{\varepsilon}} = 0.17$, 0.20, and 0.23 are shown in Figures 1–3. From the figures, we see that if we picked a hypothesis with $\hat{\varepsilon}(h) = 0.17$ or $\hat{\varepsilon}(h) = 0.23$, we can expect the hypothesis to have higher generalization error than one with $\hat{\varepsilon}(h) = 0.20$. It was, of course, expected that picking a hypothesis with CV error 0.20 beat picking one with 0.23, but it was surprising that picking a hypothesis with a *higher* error on the CV data ($\hat{\varepsilon}(h) = 0.20$ rather than 0.17) could, on average, result in a hypothesis with *lower* generalization error.

This was unexpected because we expect hypotheses with lower generalization error to have lower CV error. The point is that *the reverse is not necessarily true,* and hypotheses with lower CV error do not necessarily have lower expected generalization error. That is, given fixed S, D_H, and f for this problem, low $\overline{\varepsilon} \Rightarrow$ low $\mathrm{E}_{h\in D_H}[\hat{\varepsilon}(h)|\,\varepsilon(h) = \overline{\varepsilon}]$, but low $\overline{\hat{\varepsilon}} \not\Rightarrow$ low $\mathrm{E}_{h\in D_H}[\varepsilon(h)|\,\hat{\varepsilon}(h) = \overline{\hat{\varepsilon}}]$.

4 Not Choosing the "Best"

Because of this "overfitting" of CV data, "folklore" suggests generating only a limited number of hypotheses from D_H before picking the one that has the lowest CV error. The rationale is that if we generate too few hypotheses, then we might not generate any good hypotheses; whereas if we generate too many, we increase the risk of generating poor hypotheses that fit the CV data well "by chance," and overfit. So, the optimal number of hypotheses to generate should be between the two extremes, and we will call this number n_{opt}. But, such practice seems sub-optimal. For example, it is not clear how n_{opt} can be found. Moreover, if we knew the value of n_{opt}, but have already generated more than n_{opt} hypotheses, then it is unclear if we should throw-away all but n_{opt} of them, or if there is a better way to use the extra hypotheses. Finally, we would also like an algorithm that asymptotically improves as it is given more hypotheses.

The problem with generating "too many" hypotheses was that those with very low CV errors were actually "overfitting," and had high generalization error. Because, of any large set of hypotheses drawn from the fixed distribution of hypotheses, the fraction of them that "overfit" should be approximately the same, we propose the following, which we call *percentile-cv*:

1. Generate as many hypotheses as is computationally feasible.
2. Test them all on our CV data S, and sort them in descending order of $\hat{\varepsilon}_S(h_i)$.
3. For some k, pick the hypothesis in the k-th percentile in the list of sorted hypotheses. (i.e. if we have n hypotheses, then choose the one that was ranked $\lceil kn/100 \rceil$ in the sorted list.)

If multiple hypotheses fall into the k-percentile by having the same CV error, then we pick uniformly from them. Note also that we have introduced a new variable, k, that we have not specified a way to choose yet, but also that we no longer need to guess the value of n_{opt}.

The motivation behind percentile-cv is that when we sort the hypotheses in order of $\hat{\varepsilon}$'s as in Step 2 above, we hope that the hypotheses are approximately sorted into order of "truly bad" (high ε, high $\hat{\varepsilon}$), followed by "good" (low ε, low $\hat{\varepsilon}$), followed by "bad, but overfit CV data by chance" (high ε, very low $\hat{\varepsilon}$). Then, if we chose k well, we would pick something between the "truly bad" and "bad, but overfit CV data by chance," and choose a good hypothesis. Note, however, that we will be *choosing a hypothesis with higher CV error over some others with lower CV errors.*

Also, as the number of hypotheses generated tends towards ∞, since the hypotheses are drawn *i.i.d.* from a fixed distribution, we expect fixed proportions of them to be "truly bad," "good," and "bad, but overfit," so we will still be choosing a hypothesis that is "good." And as an example, in the domain of the simulated algorithm described in Section 3, we would hope that k is chosen such that the k-th percentile hypothesis is one that has $\hat{\varepsilon}(h) = 0.20$.

We now present a result that further justifies using percentile-cv, which generates a large set of hypotheses and picks some k-th percentile hypothesis from it, instead of best–of–n, which generates some finite set of n hypotheses, and picks the minimum CV-error hypothesis in it. The following theorem states that, with a well chosen k, percentile-cv always does at least well as best–of–n (i.e. $\exists k$ such that k-th percentile beats best–of–n, for all n).

Theorem 1 *For any D_H, f, and S, $\exists k$ such that, $\forall n$, choosing the k-th percentile hypothesis from a (sufficiently) large set of hypotheses drawn i.i.d. from D_H, gives an expected generalization error no higher than picking the "apparent best" of n randomly generated hypotheses.*

Proof (Sketch): For any fixed n, picking the best–of–n is equivalent to the following in terms of performance: (1) Generating a set H of n hypotheses, noting down the lowest CV error $\hat{\varepsilon}_0 = \min_{h \in H} \hat{\varepsilon}(h)$, and then (2) generating fresh hypotheses until one has the same CV error of $\hat{\varepsilon}_0$, and outputting that hypothesis, which we shall call h_{f}. Now, step (2) is simply drawing h_{f} from the conditional distribution $h \in D_H | \hat{\varepsilon}(h) = \hat{\varepsilon}_0$; and of the possible $\hat{\varepsilon}_0$'s that could be chosen, one is "best" in terms of giving the conditional distribution that minimizes $\mathrm{E}_{h_{\text{f}} \in D_H}[\varepsilon(h_{\text{f}}) | \hat{\varepsilon}(h_{\text{f}}) = \hat{\varepsilon}_0]$ during this step. Call this $\hat{\varepsilon}_{\text{opt}}$. Then, there exists k such that the k-th percentile hypothesis will have $\hat{\varepsilon}(h) = \hat{\varepsilon}_{\text{opt}}$ with probability 1, meaning that this k-th percentile hypothesis has an expected generalization error of $\mathrm{E}_{h \in D_H}[\varepsilon(h) | \hat{\varepsilon}(h) = \hat{\varepsilon}_{\text{opt}}]$. So, for all n, $\exists k$ such that k-th percentile (not strictly) beats the apparent best of n. This implies that some optimal k must beat *all* n, which is the statement of the theorem.

Corollary 2 *If $\mathrm{E}_{h_{\text{f}} \in D_H}[\varepsilon(h_{\text{f}}) | \hat{\varepsilon}(h_{\text{f}}) = \hat{\varepsilon}_0]$ is not constant over all possible $\hat{\varepsilon}_0$, $\exists k$ such that $\forall n$, the expected generalization error of the k-th percentile hypothesis is strictly less than that of the best–of–n hypothesis.*

5 Choosing a Percentile

Using the conventional hypothesis selection method, we had to estimate n_{opt}, the number of hypotheses to generate before picking the one that had the lowest CV error. In this section, we describe *Leave-one-out Cross-Validated Cross-Validation* (LOOCVCV), a new algorithm that first finds an estimate $\hat{n}$ of n_{opt}, then chooses k from that, and finally applies percentile-cv using that value of k. We will first show how to choose k from n_{opt} or an estimate $\hat{n}$ of it, and then how to estimate $\hat{n}$.

5.1 Choosing k from n_{opt}

For a fixed CV set S, since the hypotheses h_i's are drawn from a fixed distribution, their CV errors $\hat{\varepsilon}_S(h_i)$ are also drawn from some fixed distribution. Hence, if we generate n_{opt} hypotheses and pick the one with the lowest CV error, then to a "good approximation," we expect to pick a hypothesis whose CV error just falls into the bottom $1/(n_{opt}+1)$ fraction of all CV errors drawn from our distribution of $\hat{\varepsilon}$'s. (The proof of this is from the literature on the statistical theory of Extreme Value Distributions. See [Leadbetter et al., 1980], for example.) We therefore propose using the following value of k:

$$k = 100 \cdot \left(1 - \frac{1}{n_{opt}+1}\right) \qquad (1)$$

Using notation from the proof of Theorem 1, we can also think of picking the best–of–n hypothesis as a way of choosing $\hat{\varepsilon}_0$ (and then picking a hypothesis h with $\hat{\varepsilon}(h) = \hat{\varepsilon}_0$). Now, suppose we already have a good estimate $\hat{n}$ of n_{opt}; then a natural alternative to picking the apparent best of $\hat{n}$ is to pick the k-th percentile hypothesis, where $k = 100(1 - (1/(\hat{n}+1)))$, with the advantage of doing this being reducing the *variance* of $\hat{\varepsilon}_0$, and hopefully the "variance" of the hypothesis we pick as well. Later, we will also give experimental results that demonstrate that this can do significantly better than picking the best–of–$\hat{n}$.

5.2 Estimating n_{opt}

We will only sketch the main ideas of the algorithm here, and leave the details to the Appendix. We want to use Leave-one-out Cross-Validation (LOOCV) as follows: For each value of $\hat{n}$, we want a measure of the generalization error of the best–of–$\hat{n}$ hypothesis. So, for each sample $p_i \in S$, we shall "leave-out" p_i and use S_{-i} only, choosing the hypothesis $h^*_{i,\hat{n}}$ that has the minimum empirical error on S_{-i} among a set of $\hat{n}$ randomly generated hypothesis; and then test $h^*_{i,\hat{n}}$ on p_i and see if $h^*_{i,\hat{n}}(x_i) \stackrel{?}{=} y_i$. Then, the fraction of values of i for which $h^*_{i,\hat{n}}(x_i) \neq y_i$, which we call the *LOOCV-error*, is a measure of the generalization error of the best–of–$\hat{n}$ hypothesis. And running this process on different $\hat{n}$'s, we can get a measure of the LOOCV-error for each value of $\hat{n}$, and pick the value of $\hat{n}$ that minimizes the LOOCV-error.

The problem with the current formulation of this algorithm is that choosing the best–of–$\hat{n}$ hypothesis is a noisy process, as it involves generating randomly chosen hypotheses. So, the entire process has to be repeated many times in a Monte Carlo fashion for each value of $\hat{n}$, to get estimates of the LOOCV-error. This will, unfortunately, generally require generating an intractably large number of hypotheses. So instead, we will generate only a large but finite set H of hypotheses, and choose uniformly and with replacement from this set whenever we need to generate a hypothesis (this, from the Statistics literature, is very similar to non-parametric bootstrap [Efron, 1979]). However, even doing this, we found that we would still require averaging over an intractable or otherwise very large number of repetitions to get a smooth curve for the LOOCV-error as a function of $\hat{n}$. But, there is fortunately a way to calculate explicitly, from $\hat{n}$, H, and S, the LOOCV-error value we would get if we really repeated this bootstrap-like process an infinite number of times, and this is what LOOCVCV does. (Details in appendix.)

So, for every value of $\hat{n}$, we now have a tractable way of estimating the generalization error of the best–of–$\hat{n}$ hypothesis, and we can use the value of $\hat{n}$ that minimizes the LOOCV-error as our estimate of n_{opt}. With this, we can use Equation 1 to derive k, our desired percentile, which we can then use to apply percentile-cv. This completes our formulation of the LOOCVCV algorithm.

6 Results Using the Simulated Learning Algorithm

We ask the reader to refer to Figure 4, which shows results of different hypothesis selection methods applied to the Simulated Learning Algorithm described in Section 3. The value of n varies along the x-axis, and the top-most curve is the expected generalization error if we draw n hypotheses and pick the one with the lowest CV error. The middle, stepped, line is the expected generalization error if we picked the k-th percentile hypothesis, with $k = 100(1 - (1/(n+1)))$, using an "infinitely" large set of hypotheses. Finally, the bottom horizontal line is the generalization error that LOOCVCV achieves, also using an "infinitely" large set of hypotheses.

For this particular problem, $n_{opt} \approx 101$. Notice, however, that if we generated exactly n_{opt} hypotheses and picked the one with the lowest CV error, our expected generalization error would be about 0.035, which is higher than if we picked the $k = 100(1 - (1/(101+1))) \approx 99.0$-th percentile hypothesis from an "infinitely" large sample of hypotheses, which gives an expected generalization error of about 0.025. The reason for this is that with best–of–n, the CV error $\hat{\varepsilon}(h)$ of the hypothesis we choose in the end has non-zero variance, and therefore performs worse than percentile-cv, by our argument of Section 5.1. That is, $\mathrm{E}_{h \in D_H}[\varepsilon(h)|\hat{\varepsilon}(h) = \bar{\hat{\varepsilon}}]$ is minimized when $\bar{\hat{\varepsilon}} = 0.20$. However, in the process of drawing n_{opt} hypotheses and picking the one with lowest CV error, we may occasionally pick hypotheses with other values of $\hat{\varepsilon}(h)$, which is why its expected generalization error is higher. On the other hand, in the "infinitely" large sample of hypotheses, the 99.0-th percentile hypothesis has $\hat{\varepsilon}(h) = 0.20$ with probability 1, which is why picking the 99.0-th percentile hypothesis results in a lower expected generalization error than best–of–101: There is less variance in the CV error of the hypothesis we pick.

Similarly, note too that, for most values of the curve, k-th percentile (the middle curve) beats picking the apparent best of n (the top curve). We argue that this is due to the same reason: using k-th percentile results in lower variance in the CV error of the chosen hypothesis. This is further justification for our earlier argument that, if we did not want to apply LOOCVCV, but already had an estimate $\hat{n}$ of n_{opt}, then we may be better off picking the $100(1-(1/(\hat{n}+1)))$-th percentile hypothesis, instead of the best–of–$\hat{n}$ hypothesis.

Finally, we note that when LOOCVCV is given an "infinitely" large set of hypotheses for this problem, it will always achieve an expected generalization error of about 0.025 which, as is also suggested by the stepped line in Figure 4 having a lowest value of 0.025, is provably the best that any algorithm using percentile-cv can achieve on this problem. The best–of–∞ hypothesis, in contrast, has a perhaps surprisingly high expected generalization error of about 0.206, which is the value that the topmost curve of Figure 4 eventually asymtotes at.

7 Experimental Results

To examine a slightly more realistic hypothesis selection problem, we considered a small learning task involving decision trees. The target function was a boolean function over binary attributes x_0 to x_4, with $f(x_0, x_1, x_2, x_3, x_4) = (x_0 \wedge x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. The input distribution D was uniform over the instance space of size 32, and $n = 1000$ hypotheses were generated to form the set H. Finally, D_H was such that each hy-

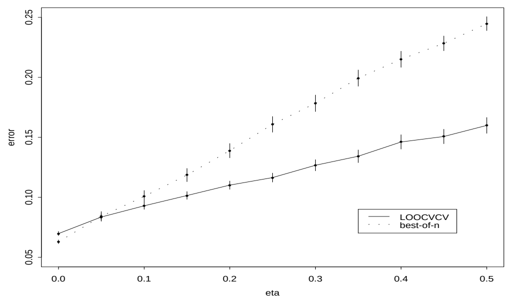

Figure 5: Noise rate on x-axis, generalization error on y-axis. m=20. Vertical dashes show 95% confidence intervals for means.

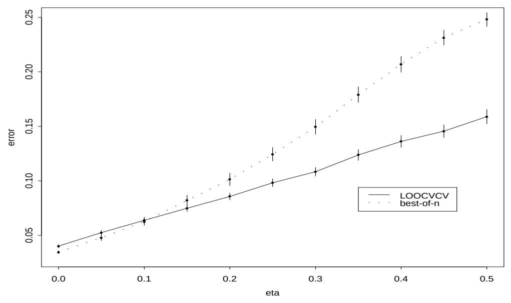

Figure 6: Noise rate on x-axis, generalization error on y-axis. m=40. Vertical dashes show 95% confidence intervals for means.

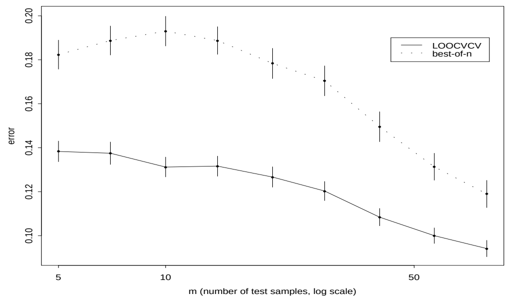

Figure 7: m (number of test-data samples, log scale) on x-axis, generalization error on y-axis. Upper curve is best–of–1000, lower curve is LOOCVCV.

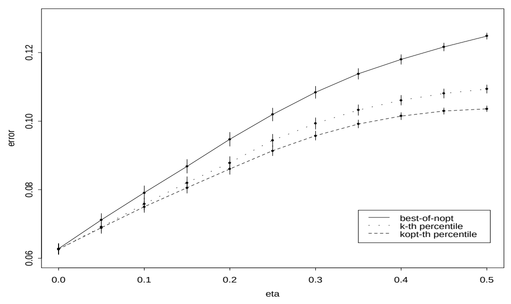

Figure 8: Noise on x-axis, error on y-axis. Uppermost curve is best–of–n_{opt}, middle curve is k-th percentile hypothesis, with $k = 100(1-(1/(n_{opt}+1)))$, bottom curve is k_{opt}-th percentile

pothesis was generated by drawing 20 samples from D with their correct labels, and running an ID3-like decision tree algorithm using a greedy information-gain heuristic to choose splits [Quinlan, 1986]. (Of course, no practitioner would ever see such a distribution of hypotheses; but, we are primarily interested in the problem of selecting a hypothesis out of a set, and this was a convenient way of creating such a set of hypotheses.)

All experimental results reported in this section are averages of 400 trials. First, using $m = 20$ CV samples, the results for different noise rates η in the CV data are in Figure 5. We see that for noise rates of about $\eta = 0.1$ and higher, picking the best–of–1000 hypothesis loses to LOOCVCV. Note too that picking the apparent best could do even worse than these results suggest, if we had used more than the 1000 hypotheses we had restricted ourselves to for computational reasons.

One might think that this effect would go away with a larger CV sample. However, it was still significant when the size of the CV sample was increased to 40 (and note that the size of the instance space is 32), keeping everything else the same as before (Figure 6). Also, we tried varying the CV-sample size, across the larger range of 5 to 80, using a fixed noise rate η of 0.3, and the results (Figure 7) also show LOOCVCV doing better consistently.

Next, to allow us to perform certain calculations exactly such as finding n_{opt}, we carried out the next few experiments by first drawing 1000 hypotheses from D_H, and then sampling with replacement from this pool of 1000 each time we needed to generate a hypothesis, instead of drawing a new one from D_H. (We also verified that these results are close to if we had used D_H.)

Recall that in Section 5.1, we claimed that if we had an estimate of a good $\hat{n}$ to use in picking best–of–$\hat{n}$, we might instead wish to use the k-th percentile hypothesis instead, with $k = 100(1 - (1/(\hat{n}+1)))$. We see this effect again in our next experiment. Using $m = 20$ CV samples again, the top line of Figure 8 is the expected error if we knew exactly what the best n, n_{opt} were and chose the best–of–n_{opt}. The middle curve is the error of the k-th percentile hypothesis, where $k = 100(1 - (1/(n_{opt}+1)))$, and we see that it consistently beats best–of–n_{opt}. Finally, the lowermost curve is if we knew the optimal k, k_{opt}, and chose the k_{opt}-th percentile hypothesis, which, as predicted by Theorem 1, beats best–of–n_{opt} also.

8 Discussion

The the results we have presented so far generally show LOOCVCV beating best–of–n, by picking a hypothesis other than the one with the smallest CV error. But best–of–n, especially with large n, is like using $k = 100$, so if there were insufficient data to find an accurate estimate k of k_{opt}, or if k_{opt} is close to 100, then we might expect best–of–n to beat LOOCVCV's using a possibly noisy estimate of k_{opt}. In fact, at the noise rate $\eta = 0$, LOOCVCV generally does lose to best–of–n (Figures 5 and 6). Also, some preliminary results (not reported) involving artificial neural networks, trained by backpropagation on a fixed set of training data and with randomly re-initialized weights each time, also showed LOOCVCV losing slightly to best–of–n.

Leaving considerations of a loss function of k aside, we might therefore decide not to use LOOCVCV unless we had reason to believe that the "overfitting" problem is significant in the distribution of hypotheses in our hypothesis selection problem, or that LOOCVCV will find a good $\widehat{k_{opt}}$ (such as when the CV sample is large). However, if we do have an exceedingly large number of hypotheses to choose from, and have a reasonably large CV-set, then LOOCVCV may give much better performance than simply selecting the hypothesis with the lowest CV error.

Before closing, one reformulation of our problem is worth commenting on here. Let us call try calling H a "hypothesis class," and S "training data." With this formulation, one might then have thought that for any fixed input space, n would have been a good surrogate measure for the "complexity" of H, and might have tried to apply conventional complexity regularization techniques using, say, $\log n$ as some "measure" of the "complexity" of H. But this turns out to be unsound as a general approach, because it fails to take into account the effect of D_H. For example, one can easily produce two different D_H's such that the "expressiveness" or richness of, say, $n = 1000$ hypotheses drawn from the first D_H is vastly greater then the expressiveness of 1000 hypotheses drawn from the second. However, with this "hypothesis class" and "training data" formulation, one point of interest is in noting that the arguments of this paper still apply exactly as before, and suggest the surprising result that it might not be optimal to pick the hypothesis from a hypothesis class H that has the lowest training error, even when there is no notion of complexity within H in that H is not, for example, partitioned into a nested sequence of hypothesis classes of increasing complexity. This possibility will be a subject of our future research.

Finally, these results might have implications for the practitioner as well. If cross-validation is used to decide how many epochs to train a backpropagation neural network (which, while not falling within our assumed framework of *i.i.d.h*'s, is similar), and if the practitioner does not somehow use the common assumption that hypotheses trained with more epochs are more "complex"; or if cross-validation is used to select from a large pool of learning algorithms, then our results suggest it may *not* be advisable to pick the minimum cross-validation error hypothesis. However, how strongly these results of this paper will bear out in future practice remains to be seen.

9 Conclusions

This paper explained how overfitting of CV data occurs despite there being no notion of *complexity* within the set of hypotheses, and presented percentile-cv and LOOCVCV, that try to correct this problem. It also proved that percentile-cv with an appropriately chosen percentile will always do at least as well as best–of–n, and demonstrated experimentally that LOOCVCV consistently beats best–of–n in one domain. However, there are also domains where overfitting of CV data is not a significant problem, in which LOOCVCV loses slightly to best–of–n; and the characterization of such domains is an open problem.

Acknowledgements

I give warm thanks to Michael Kearns and Andrew Moore for many interesting and informative discussions, that contributed much to this work.

Appendix: The LOOCVCV Algorithm

Please refer to Section 5 for the details of the LOOCVCV algorithm. Here, we describe how it finds an estimate $\hat{n}$ of n_{opt} using H, from which we can derive k, for picking the k-th percentile hypothesis.
Input: $\hat{n}$, $H[1..n]$, S, i
Output: leave-one-out error with p_i "left-out," for best–of–$\hat{n}$ hypothesis drawn from H
Notes: The key idea is sorting H in order of leave-one-out errors and then calculating, for each $h \in H$, the *probability* (first term in the summation in step 12) that it will be the lowest-LOOCV-error hypothesis in a set of $\hat{n}$ drawn from H. These probabilities can then be used to find the probability that the selected hypothesis will apparently misclassify p_i. $\overline{e}$ is used to accomodate multiple hypotheses in the set of $\hat{n}$ having the same lowest LOOCV-error, in which case we pick uniformly from them.

```
1. function loocv-error($\hat{n}$, $H[1..n]$, $S$, $i$)
2.     create array $H'[1..n].\{\hat{\varepsilon}_{-1}, e\}$
3.     for ($j$ = 1 to $n$) do begin
```

```
4.          $H'[j].\hat{\varepsilon}_{-1} = |\{H[j](x_k) \neq y_k | (x_k, y_k) \in S_{-i}\}|$
5.          $H'[j].e = 1$ if $H[j](x_i) \neq y_i$, 0 otherwise
6.          end
7.       Sort $H'[]$ in ascending order of $H'[].\hat{\varepsilon}_{-1}$
8.       for ($x = 0$ to $|S| - 1$) do begin
9.          $A = \{j | H'[j].\hat{\varepsilon}_{-1} = x\}$
10.         let $\overline{e}(x) = \frac{1}{|A|} \sum_{j \in A} H'[j].e$ if $|A| \neq 0$,
                 undefined otherwise.
11.         end
12. return $\sum_{j=1}^{n} \left( (\frac{n-j+1}{n})^{\hat{n}} - (\frac{n-j}{n})^{\hat{n}} \right) \overline{e}(H'[j].\hat{\varepsilon}_{-1})$
```

Input: H, S
Output: Recommended $\hat{n}$

```
13. function loocvcv-estimate-nopt($H$,$S$)
14. return
         $\mathrm{argmin}_{\hat{n} \in [1,\infty)} \{\sum_{i=1}^{m}$ loocv-error$(\hat{n}, H, S, i)\}$
```

References

[Baum and Haussler, 1989] Baum, E. and Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, 1(1):151–160.

[Efron, 1979] Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Anns. Statist.*, 7:1–26.

[Judd, 1990] Judd, J. S. (1990). *Neural Network Design and the Complexity of Learning.* MIT Press.

[Kearns, 1996] Kearns, M. J. (1996). A bound on the error of Cross Validation using the approximation and estimation rates, with consequences for the training-test split. In *Advances in Neural Information Processing Systems 8*, pages 183–189. Morgan Kaufmann.

[Kearns et al., 1995] Kearns, M. J., Mansour, Y., Ng, A. Y., and Ron, D. (1995). An experimental and theoretical comparison of model selection methods. In *Proceedings of the Eighth ACM Conference on Computational Learning Theory*, pages 21–30. ACM Press.

[Klockars and Sax, 1986] Klockars, A. J. and Sax, G. (1986). *Multiple Comparisons.* Sage Publications.

[Leadbetter et al., 1980] Leadbetter, M. R., Lindgren, G., and Rootzén, H. (1980). *Extremes and Related Properties of Random Sequences and Processes.* Springer Verlag.

[Miller, 1990] Miller, A. J. (1990). *Subset Selection in Regression.* Chapman and Hall.

[Mosteller and Tukey, 1968] Mosteller, F. and Tukey, J. W. (1968). Data analysis, including statistics. In Lindzey, G. and Aronson, E., editors, *Handbook of Social Psychology Vol. 2*, pages 1–26. Addison-Wesley.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

[Stone, 1974] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *J. Royal Statistical Society B*, 36:111–147.

[Stone, 1977] Stone, M. (1977). Asymtotics for and against cross-validation. *Biometrika*, 64(1):29–35.

[Tukey, 1949] Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 9:99–114.

[Tukey, 1953] Tukey, J. W. (1953). The problem of multiple comparisons. *Unpublished manuscript*, Princeton University.

[Vapnik and Chervonenkis, 1971] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.