



**Validation of Voting Committee**

**Bax, Eric**

**Computer Science Department  
California Institute of Technology**

**Caltech-CS-TR-97-13**

# Validation of Voting Committees

Eric Bax\*

June 2, 1997

## Abstract

This paper contains a method to bound the test errors of voting committees with members chosen from a pool of trained classifiers. There are so many prospective committees that validating them directly does not achieve useful error bounds. Because there are fewer classifiers than prospective committees, it is better to validate the classifiers individually, then use linear programming to infer committee error bounds. We test the method using credit card data. Also, we extend the method to infer bounds for classifiers in general.

**Key words** machine learning, learning theory, fusion, committees, voting, Vapnik-Chervonenkis, linear programming.

---

\*Computer Science Department, California Institute of Technology 256-80, Pasadena, California, 91125 ([eric@cs.caltech.edu](mailto:eric@cs.caltech.edu)).

## 1 Introduction

Consider the following machine learning problem. There is an unknown boolean-valued target function, and there is a distribution over the input space of the function. For example, the input distribution could consist of images encountered by commuters in a city, and the target function could be 1 if the input image contains a yellow tram and 0 otherwise.

We have a set of in-sample data examples with inputs drawn according to the input distribution and outputs determined by the target function. We also have a set of test example inputs drawn according to the input distribution. Our goal is to find a classifier function with a low error rate on the test inputs. (The error rate is the fraction of examples for which the classifier and the target function disagree.)

We use a portion of the in-sample data to train a pool of candidate classifiers. We use the remaining in-sample data, called the validation data, to select a voting committee of candidate classifiers. For each input, the committee returns the result shared by the majority of its members. (To avoid confusion, we restrict our attention to committees with odd numbers of members.) In this paper, we develop a new method to bound the test error of the selected committee.

In the next section, which focuses on validation, we derive VC-type [Vapnik and Chervonenkis 1971] uniform bounds on test error. To achieve useful error bounds through validation, the number of classifiers must be small. Since the classifier pool is much smaller than the number of prospective committees, the classifier errors can be bounded more precisely than the committee errors.

Next, we show how to infer committee error bounds from member error bounds and test inputs. We use linear programming to find the maximum possible committee error given constraints imposed by member error bounds and the distribution of agreements among members.

In the following section, we present some small numerical examples to build intuition regarding the new method. Then we analyze the committee error bound. We compare the linear programming method to direct validation, exploring differences in asymptotic behavior and discussing tradeoffs between the methods.

In the next section, we extend the linear program to infer error bounds for classifiers in general – not just voting committees. This makes the new method a valuable tool to derive error bounds for stacking classifiers [Wolpert 1992, Sridhar, Seagrave, and Bartlett 1996, Kim and Bartlett 1995, Breiman 1992]. In stacking, the outputs from trained classifiers are fed into higher levels of classifiers. Generalization is improved through fusion and by estimating bias through resampling. Our method allows developers of stacked classifiers to derive test error bounds by the following process. Train the initial layer of classifiers, withholding some validation data. Use this data to compute uniform test error bounds for the classifiers. Use all in-sample data to train subsequent layers. Finally, infer a test error bound for the stacked classifier through linear pro-

gramming.

In the following section, we present tests to compare linear programming and direct validation as methods to compute committee test error bounds. Each example in the data set corresponds to a credit card applicant, and the task is to predict whether or not the applicant defaults. We find that linear programming produces superior bounds to direct validation.

## 2 Uniform Test Error Bounds and Validation

We develop two uniform upper bounds on the test errors of several classifiers. The first bound is weaker, but it is smooth. We will use it for analysis. The second bound is stronger. We will use it for tests on real data.

For the first bound, we modify a simplified treatment of VC error bounds [Vapnik and Chervonenkis 1971, Abu-Mostafa 1996]. Suppose we have  $M$  classifiers. Let  $d$  be the number of validation examples, and let  $d'$  be the number of test examples. Let  $\nu_m$  be the validation error of classifier  $m$ , and let  $\nu'_m$  be the test error. Let  $\pi_m$  be the error rate over the entire input distribution, i.e. the expected error rate for a random data set. For a single classifier selected without reference to the validation or test data, Hoeffding's inequality [Hoeffding 1963, Vapnik 1982] implies

$$\Pr\{\pi_m \geq \nu_m + \epsilon\} \leq e^{-2\epsilon^2 d} \text{ for } \epsilon > 0 \quad (1)$$

and

$$\Pr\{\nu'_m \geq \pi_m + \epsilon\} \leq e^{-2\epsilon^2 d'} \text{ for } \epsilon > 0 \quad (2)$$

Using both bounds,

$$\Pr\{\nu'_m \geq \nu_m + \epsilon\} \leq \Pr\{\nu'_m \geq \pi_m + \frac{\epsilon}{2}\} + \Pr\{\pi_m \geq \nu_m + \frac{\epsilon}{2}\} \leq 2e^{-\frac{1}{2}\epsilon^2 D} \quad (3)$$

where  $D = \min(d, d')$ . For a single classifier, the test error rate is less than  $\epsilon$  greater than the validation error rate with probability at least  $1 - 2e^{-\frac{1}{2}\epsilon^2 D}$ .

For uniform bounds over  $M$  classifiers selected without reference to the validation or test data, bound the probability union by the sum of probabilities.

$$\Pr\{\nu'_1 \geq \nu_1 + \epsilon \text{ or } \dots \text{ or } \nu'_M \geq \nu_M + \epsilon\} \leq 2Me^{-\frac{1}{2}\epsilon^2 D} \quad (4)$$

If validation data is used to choose a classifier, then the single classifier bound (3) does not apply. However, if the set from which the classifier is chosen is developed without reference to the validation data, then the uniform bound (4) applies to the chosen classifier, since uniform bounding implies bounding of the chosen classifier.

This is exactly the case when we train a pool of  $n$  classifiers, withholding validation data, then we use the data to select a committee of  $k$  classifiers. There are  $M = \binom{n}{k}$  prospective committees. The probability that no committee test error is more than  $\epsilon$  greater than its validation error is at least  $1 - 2ne^{-\frac{1}{2}\epsilon^2 D}$ . Hence, we have confidence  $1 - 2ne^{-\frac{1}{2}\epsilon^2 D}$  in the error bound for the chosen committee.

Using uniform bounds for a single classifier and bounding the probability union by the sum of probabilities in (4) seems quite wasteful. Our bound reduces the waste by using uniform bound (4) only over the pool of classifiers, then using linear programming to infer uniform bounds over the prospective committees.

Now we develop the second error bound, which is a generalization of the bound at the heart of the original VC bound proof [Vapnik and Chervonenkis 1971]. For a single classifier, condition the bound on a given multiset of  $b = d + d'$  inputs composing the validation and test inputs. Since the inputs are drawn i.i.d., each partition of the inputs into validation and test sets is equally likely. Let  $w$  be the number of inputs for which the classifier produces the incorrect output. The probability that the validation error is  $\frac{r}{d}$  is

$$\binom{b}{d}^{-1} \binom{w}{r} \binom{b-w}{d-r} \quad (5)$$

If the validation error is  $\frac{r}{d}$ , then the test error is  $\frac{w-r}{d'}$ . So

$$\Pr\{\nu'_m \geq \nu_m + \epsilon | w\} = \sum_{\{r | \frac{w-r}{d'} \geq \frac{r}{d} + \epsilon\}} \binom{b}{d}^{-1} \binom{w}{r} \binom{b-w}{d-r} \quad (6)$$

Bound by maximizing over  $w$ .

$$\Pr\{\nu'_m \geq \nu_m + \epsilon\} \leq \max_{w \in \{0, \dots, b\}} \Pr\{\nu'_m \geq \nu_m + \epsilon | w\} \quad (7)$$

Refer to the bound as  $B(\epsilon)$ . Note that the bound is constant over all multisets of  $b = d + d'$  inputs. Hence, integration over all possible input multisets removes the conditional nature of the bound.

For a single classifier chosen without reference to validation data

$$\Pr\{\nu'_m \geq \nu_m + \epsilon\} \leq B(\epsilon) \quad (8)$$

The uniform bound for  $M$  classifiers is

$$\Pr\{\nu'_1 \geq \nu_1 + \epsilon \text{ or } \dots \text{ or } \nu'_M \geq \nu_M + \epsilon\} \leq MB(\epsilon) \quad (9)$$

### 3 Committee Error Bound

We will use the statistics of the voting patterns among committee members to bound committee error. For each subset  $S$  of  $\{1, \dots, k\}$ , define  $a_S$  to be the fraction of test examples for which the classifiers indexed by  $S$  return 1 and the other classifiers return 0. For example,  $a_\emptyset$  is the fraction of examples for which every committee member returns 0. Likewise, if  $k = 5$ , then  $a_{\{1,3,4\}}$  is the fraction of examples for which members 1, 3, and 4 return 1 and members 2 and 5 return 0.

Given error bounds on the members and the distribution of agreements, we will use linear programming to find an upper bound for committee error. Let  $e_1, \dots, e_k$  be upper bounds for the error rates of members. For each subset  $S$  of  $\{1, \dots, k\}$ , define  $x_S$  to be the probability that the committee vote returns the incorrect value, given that the committee members indexed by  $S$  return 1 and the other committee members return 0. The committee error rate is  $\sum_S a_S x_S$ , the weighted average of errors over voting patterns.

For each member  $i$ , define  $W_i$  to be the set of sets  $S$  for which member  $i$  votes with the majority on the examples counted by  $a_S$ .

$$W_i = \{S | i \in S \text{ and } |S| > \frac{k}{2}\} \cup \{S | i \notin S \text{ and } |S| < \frac{k}{2}\} \quad (10)$$

The error rate of member  $i$  is

$$\sum_{S \in W_i} a_S x_S + \sum_{S \notin W_i} a_S (1 - x_S) \quad (11)$$

Hence, an upper bound for the committee error rate can be found by solving the following linear program.

$$\text{maximize } \sum_S a_S x_S \quad (12)$$

over  $x_\emptyset, \dots, x_{\{1, \dots, k\}}$  such that

$$\sum_{S \in W_i} a_S x_S + \sum_{S \notin W_i} a_S (1 - x_S) \leq e_i \quad \forall i \in \{1, \dots, k\} \quad (13)$$

$$0 \leq x_S \leq 1 \quad \forall S \subset \{1, \dots, k\} \quad (14)$$

To use the member bounds derived from validation, set  $e_i = \nu_i + \epsilon$  for each member, and solve the linear program to error bound the committee. With confidence  $1 - 2ne^{-\frac{1}{2}\epsilon^2 D}$ , the linear program solution is an upper bound for the committee error.

The linear program, as written, has many variables. In many cases, most of them are unnecessary. For each  $S$ , if there is no test example for which each classifier indexed by  $S$  returns 1, and each other classifier returns 0, then

$a_S = 0$ . For each such  $S$ ,  $x_S$  plays no role in the linear program. Rewrite the linear program to use only the variables  $x_S$  for which  $a_S \neq 0$ . Change all indices of summation to the intersection of the set  $\{S | a_S \neq 0\}$  and the present indices. Now the linear program has no more variables than the number of test data examples.

To further reduce the number of variables, note that  $a_S$  and  $a_{\bar{S}}$  count votes with the same members in the majority. Hence, coefficients and variables with complementary subscripts play the same role in the linear program. Rewrite to keep one of each pair – for each  $S$  with  $1 \in S$ , let  $a_S = a_S + a_{\bar{S}}$  and  $x_S = x_S + x_{\bar{S}}$ .

## Procedure

We have derived the following procedure to train classifiers, choose a committee using validation data, and compute a test error bound for the committee:

1. Partition the in-sample data into training and validation sets.
2. Train a pool of  $n$  classifiers using the training data.
3. Compute test error bounds for the classifiers using the validation data. Choose  $\epsilon$  to give the desired confidence ( $1 - 2ne^{\frac{1}{2}\epsilon^2 D}$  for Hoeffding-style bounds or  $1 - nB(\epsilon)$  for partition-based bounds.) Then compute validation errors. For each classifier, the test error bound is the sum of its validation error and  $\epsilon$ .
4. Use the validation data to select a committee of  $k$  classifiers from the pool. (The training data may also be used.) If feasible, simply evaluate the validation errors of all prospective committees, and choose the committee with minimum validation error. If there are too many prospective committees, select one by some other search method.
5. For the committee, let  $e_1, \dots, e_k$  be the member test error bounds  $\nu_1 + \epsilon, \dots, \nu_k + \epsilon$  from Step 3. Compute the distribution of agreements  $a_\emptyset, \dots, a_{\{1, \dots, k\}}$  over the test set inputs. Solve linear program (12), (13), and (14) for  $x_\emptyset, \dots, x_{\{1, \dots, k\}}$ . With confidence  $1 - 2ne^{\frac{1}{2}\epsilon^2 D}$  (or  $1 - nB(\epsilon)$  for partition-based bounds,) the committee test error is no greater than  $\sum_S a_S x_S$ .



## 4 Intuition and Analysis

The error bound for a prospective committee is a function of member error bounds and the distribution of agreements. Lower member error bounds produce a lower committee error bound. Also, as the frequency of unanimous and near-unanimous votes decreases, the committee error bound decreases. So linear programming gives low bounds for committees of intelligent and contentious classifiers.

To build intuition, examine the linear program error bound in a few cases. First, suppose all members agree on every test input ( $a_\emptyset + a_{\{1,\dots,k\}} = 1$ ). Then the committee error bound is the minimum member error bound. This makes sense; we assume every member bound is correct, so when the committee agrees with all members, we use the tightest member bound for the committee.

Now focus on the role of disagreement among classifiers. Suppose there are three committee members with equal error bounds. Suppose the votes are evenly distributed over voting patterns with a 2-1 split ( $a_1 = a_2 = a_3 = a_{1,2} = a_{1,3} = a_{2,3} = \frac{1}{6}$  and  $a_\emptyset = a_{1,2,3} = 0$ ). When the member error bounds are  $\frac{1}{3}$  or less, the committee error bound is zero. Each vote has  $\frac{1}{3}$  of the members in dissent, so the errors of each member are “consumed” by votes in which the member is overruled. For member error bounds  $b \in (\frac{1}{3}, \frac{1}{2})$ , the committee error bound is  $3b - 1$ . Over this domain, the committee error bound is lower than the member error bounds. When the member error bounds are  $\frac{1}{2}$ , the committee error bound is  $\frac{1}{2}$  – random members form a random committee.

As problem size grows, there is increasing advantage in bounding committee error by linear programming. Recall

$$\Pr\{\nu'_1 \geq \nu_1 + \epsilon \text{ or } \dots \text{ or } \nu'_M \geq \nu_M + \epsilon\} \leq 2Me^{-\frac{1}{2}\epsilon^2 D} \quad (15)$$

For fixed failure probability  $c = 2Me^{-\frac{1}{2}\epsilon^2 D}$ ,

$$\epsilon = \sqrt{\frac{2}{D}(\ln 2M - \ln c)} \quad (16)$$

So the tightness ( $\epsilon$ ) of the error bound is  $O(\sqrt{\ln M})$ . Validating individual classifiers to use the linear program gives  $O(\sqrt{\ln n})$ . Validating committees directly gives  $O(\sqrt{\ln \binom{n}{k}})$ . If  $k = \frac{n}{2} + 1$ , for example, then  $\binom{n}{k} \sim \sqrt{\frac{2}{\pi n}} 2^n$  (see [Feller 1968] p. 180), and  $\epsilon$  grows as  $O(\sqrt{n})$ .

If the committee size is fixed, then  $\binom{n}{k} \sim n^k \frac{1}{k!}$ . In this case, the ratio of  $\epsilon$  for direct committee evaluation to  $\epsilon$  for linear programming goes to  $\sqrt{k}$  as  $n \rightarrow \infty$ .

As a concrete example, suppose we wish to select a committee of  $k = 21$  classifiers from a pool of  $n = 100$ . We have 1000 validation examples, and we want to bound committee errors with no more than  $c = 5\%$  chance of failure. Then the linear programming error bounds have tightness  $\epsilon = .13$ , and the direct committee error bounds have tightness  $\epsilon = .32$ .

For small problems, the linear programming method will not always give lower or more accurate error bounds than the direct method. The linear program uses relatively tight member bounds and produces the worst-case error given the distribution of agreements over the test data. If the worst-case error is very different from the actual test error, then the linear program bound gives poor information about the test error. The direct method uses the error over the validation data to estimate the error over the test data, and the estimate comes with a relatively loose bound. If the bound is too loose, then the direct method gives poor information about the actual test error.

If the test set inputs are not known, then the distribution of agreements must be estimated. The in-sample data can be used for this purpose. Also, if the underlying input distribution is known, then the distribution of agreements can be estimated by random sampling of inputs. (Note that example outputs are not needed to compute the distribution of agreements.) Estimation errors will affect the solution of the linear program; the relationship between the errors and the solution can be calculated by perturbation methods [Franklin 1980].

## 5 General Classifier Bound

The linear programming method can be extended to compute a test error bound for any classifier, not just a voting committee. Let  $g_1, \dots, g_n$  be classifiers trained without reference to some validation data. Select a confidence level, and hence  $\epsilon$ . Then use the validation data to compute uniform test error bounds  $e_1, \dots, e_n$  for  $g_1, \dots, g_n$  using (9).

Let  $g'$  be the classifier for which we will compute a test error bound. Classifier  $g'$  may be selected with reference to the validation data. Let variables  $x_1, \dots, x_{d'}$  represent error "rates" of  $g'$  on test examples.

$$\forall j \in \{1, \dots, d'\} \quad x_j = \begin{cases} 0 & \text{if } g' \text{ produces the correct output for example } j \\ 1 & \text{if } g' \text{ produces the incorrect output for example } j \end{cases} \quad (17)$$

The test error of  $g'$  is  $\sum_{j=1}^{d'} \frac{1}{d'} x_j$ .

Let  $A_i \subseteq \{1, \dots, d'\}$  indicate the set of test inputs for which  $g_i$  and  $g'$  agree. The test error of  $g_i$  is

$$\sum_{j \in A_i} \frac{1}{d'} x_j + \sum_{j \notin A_i} \frac{1}{d'} (1 - x_j) \quad (18)$$

With confidence  $1 - nB(\epsilon)$ , all test error bounds  $e_1, \dots, e_n$  hold. Hence, the following program bounds the test error of  $g'$  with the same confidence.

$$\text{maximize } \sum_{j=1}^{d'} \frac{1}{d'} x_j \quad (19)$$

over  $x_1, \dots, x_{d'}$  such that

$$\sum_{j \in A_i} \frac{1}{d'} x_j + \sum_{j \notin A_i} \frac{1}{d'} (1 - x_j) \leq e_i \quad \forall i \in \{1, \dots, n\} \quad (20)$$

$$\text{and } x_j \in \{0, 1\} \quad \forall j \in \{1, \dots, d'\} \quad (21)$$

This is an integer linear program (ILP). It may be difficult to solve for large problems, since ILP is NP-hard [Garey and Johnson 1979, Karp 1972]. Relaxing the integer constraints on the variables to

$$0 \leq x_j \leq 1 \quad \forall j \in \{1, \dots, d'\} \quad (22)$$

produces a linear program. The feasible set of the linear program is a superset of the feasible set for the integer linear program, so the linear program solution is an upper bound. Hence, the linear program returns a valid upper bound for  $g'$  with confidence  $1 - nB(\epsilon)$ .

For voting committees, this linear program is equivalent to the one presented earlier, except in one respect. This program uses constraints based on

all classifiers in the pool, while the earlier program uses only constraints based on committee members. Hence, this program returns a bound at least as strong as the one returned by the earlier program.

## 6 Tests

This section outlines the results of tests on a set of credit card data. Each example corresponds to a credit card user. There are six inputs that correspond to user traits. The traits are unknown because the data provider has chosen to keep them secret. There is a single output that indicates whether or not the credit card user defaulted. The data were obtained from a machine-learning database site at the University of California at Irvine. The discrete-valued traits were removed, leaving the six continuous-valued traits. Of the 690 examples in the original database, 24 examples had at least one trait missing. These examples were removed, leaving 666 examples. The data were cleaned by Joseph Sill. For further information, see [Sill and Abu-Mostafa 1997].

In each test, the 666 examples were randomly partitioned into 444 training examples,  $d = 111$  validation examples, and  $d' = 111$  test examples. In each test, a pool of classifiers was trained by early stopping. For each classifier, the training data were randomly partitioned into 400 examples used for actual training and 44 examples used for early stopping. The classifiers are artificial neural networks with six input units, six hidden units, and one output unit. The hidden and output units have tanh activation functions. The initial weights were selected independently and uniformly at random from  $[-0.1, 0.1]$ . The networks were trained by gradient descent on mean squared error over training examples, using sequential mode weight updates with random order of example presentation in each epoch. A snapshot of the weights was recorded after each epoch. The snapshot with minimum error on the 44 early stopping examples was returned as the trained classifier.

Partition-based uniform error bounds (9) with 90% confidence were used in all tests. Committees were selected by evaluating validation error over all prospective committees. The linear programs used constraints based on error bounds for all classifiers in the pool, not just committee members, as outlined in the previous section.

Table 1 shows the results of 10 tests with pools of 15 classifiers and voting committees of 7 members. Since  $\epsilon = 0.172$  is the minimum value for which  $1 - 15B(\epsilon) \geq 0.90$ , this value was used in the uniform error bounds for pool classifiers in the linear program bound, i.e., the linear program constraints used error bounds  $e_1 = \nu_1 + 0.172, \dots, e_{15} = \nu_{15} + 0.172$ . For the direct bound,  $\epsilon = 0.280$  is the minimum value with  $1 - \binom{15}{7}B(\epsilon) \geq 0.90$ . Hence, the direct bound is the sum of committee validation error and 0.280. Note that the linear programming bound is superior to the direct bound for every test.

Table 2 shows the results of 10 tests with pools of 10 classifiers. In each test, the odd-sized committee with minimum validation error was selected. For the linear programming bound,  $\epsilon = 0.163$  is the minimum value for which  $1 - 10B(\epsilon) \geq 0.90$ . So this value is used in the uniform bounds over classifiers in

test	committee val. error	LP test error bound	direct test error bound
1	0.297	0.487	0.577
2	0.135	0.343	0.415
3	0.198	0.388	0.478
4	0.297	0.478	0.577
5	0.234	0.397	0.514
6	0.297	0.478	0.577
7	0.225	0.433	0.514
8	0.234	0.433	0.514
9	0.189	0.388	0.505
10	0.261	0.469	0.514
avg	0.245	0.429	0.525

Table 1: Test results for  $n = 15$  classifiers in the pool and committees with  $k = 7$  members.

the pool. For the direct bound,  $\epsilon = 0.244$  is the minimum value with

$$1 - \left[ \binom{10}{1} + \binom{10}{3} + \binom{10}{5} + \binom{10}{7} + \binom{10}{9} \right] B(\epsilon) \geq 0.90 \quad (23)$$

So this value is added to the validation error to produce the direct test error bound. Once again, note that the linear programming bound is superior to the direct bound for every test.

test	committee val. error	LP test error bound	direct test error bound
1	0.225	0.388	0.469
2	0.198	0.379	0.442
3	0.216	0.379	0.460
4	0.234	0.433	0.478
5	0.225	0.388	0.469
6	0.198	0.361	0.442
7	0.180	0.343	0.424
8	0.216	0.406	0.460
9	0.180	0.352	0.424
10	0.207	0.397	0.451
avg	0.208	0.383	0.452

Table 2: Test results for  $n = 10$  classifiers in the pool and committees with odd numbers of members.

## 7 Discussion

We have developed an algorithm to compute test error bounds for voting committees through linear programming. We extended the method to compute error bounds for any classifier, using constraints based on uniformly bounded classifiers. The extended method applies to classifiers constructed by stacking, or fusion of underlying classifiers. The extended method also applies to the classifier chosen by early stopping, through the following process. Partition the in-sample data into training and validation sets. Choose an initial classifier at random, and use an iterative method to fit the classifier to the training data. As the classifier evolves during training, record a sequence of snapshots. Select a subset of the snapshots without reference to the validation data, and uniformly bound their test errors using the validation data and (9). Now, identify the snapshot with minimum validation error. It will be delivered as the result of training. To bound its test error, use the linear program with constraints provided by the uniformly bounded classifiers. For suggested methods of choosing the constraint classifiers, see [Bax, Cataltepe, and Sill 1997].

The linear program yields strong bounds when the constraint classifiers have low error rates, so the method works best with trained constraint classifiers. Disagreement among constraint classifiers encourages strong bounds. Disagreement among classifiers is also a condition for improving performance through stacking, or fusion, so the linear programming bound is a natural fit for these classifiers. Also, having one or more constraint classifiers with high rates of agreement with the classifier being error-bounded yields strong bounds. This occurs when the constraint classifiers are drawn from the training sequence [Bax, Cataltepe, and Sill 1997] to bound the classifier chosen by early stopping. It also occurs for stacking and fusion, both through data-fitting and by design [Breiman 1992].

This paper focused on binary classification problems. It would be interesting to extend the error bounding method to other problem types, e.g., regression problems.



## 8 Acknowledgements

Thanks to Joel Franklin for advice, teaching, and encouragement. Thanks to Yaser Abu-Mostafa for teaching, and thanks to Amir Atiya, Zehra Cataltepe, Malik Magdon-Ismail, Alexander Nicholson, Sam Roweis, Joseph Sill, and Xubo Song for useful pointers and discussions.

## References

- [Abu-Mostafa 1996] Y. Abu-Mostafa, What you need to know about the VC inequality, Class notes from CS156, California Institute of Technology, 1996.
- [Bax, Cataltepe, and Sill 1997] E. Bax, Z. Cataltepe, and J. Sill, Alternative error bounds for the classifier chosen by early stopping, CalTech-CS-TR-97-08.
- [Breiman 1992] L. Breiman, Stacked regressions, Tech. Rep. No. 367, Statistics Dept., Univ. of California at Berkeley (1992).
- [Feller 1968] W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley and Sons, Inc. 1968.
- [Franklin 1980] J. Franklin, *Methods of Mathematical Economics* pp.68-79, Springer-Verlag New York, Inc. 1980.
- [Garey and Johnson 1979] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York 1979. p.245.
- [Hoeffding 1963] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Am. Stat. Assoc. J.*, (1963):13-30.
- [Karp 1972] R. M. Karp, Reducibility among combinatorial problems, in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York 1972. pp. 85-103.
- [Kim and Bartlett 1995] K. Kim and E. B. Bartlett, Error estimation by series association for neural network systems, *Neural Computation*, 7(1995):799-808.
- [Sill and Abu-Mostafa 1997] J. Sill and Y. Abu-Mostafa, Monotonicity hints, to appear in *Advances in Neural Information Processing Systems*, 9. 1997.
- [Sridhar, Seagrave, and Bartlett 1996] D. V. Sridhar, R. C. Seagrave, and E. B. Bartlett, Process modeling using stacked neural networks, *AIChE Journal*, 42(9) (1996):2529-2539.
- [Vapnik 1982] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data* p.31, Springer-Verlag New York, Inc. 1982.
- [Vapnik and Chervonenkis 1971] V. N. Vapnik and A. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theory Prob. Appl.*, 16(1971):264-280.
- [Wolpert 1992] D. H. Wolpert, Stacked generalization, *Neural Networks*, 5(1992):241-259.