A Simple Algorithm for Learning Stable Machines

Savina Andonova[1] and **Andre Elisseeff**[2] and **Theodoros Evgeniou**[3] and **Massimiliano Pontil**[4]

Abstract. We present an algorithm for learning stable machines which is motivated by recent results in statistical learning theory. The algorithm is similar to Breiman's bagging despite some important differences in that it computes an ensemble combination of machines trained on small random sub-samples of an initial training set. A remarkable property is that it is often possible to just use the empirical error of these combinations of machines for model selection. We report experiments using support vector machines and neural networks validating the theory.

Keywords: Machine Learning, Statistical Learning Theory, Bagging.

1 Introduction and Notation

An important theoretical approach to analyzing the performance of learning machines is through studying their stability [7, 11, 3]. Various notions of stability have been proposed in the past [7, 11, 3], and have been used to study how combining machines can influence the generalization performance [4]. There has also been a lot of experimental work showing that combining learning machines, for example using Boosting or Bagging methods [4, 13], very often leads to improved generalization performance, and a number of theoretical explanations have been proposed [13, 4, 10].

Bagging [4] is a particular ensemble architecture consisting of a voting combination of a number of learning machines. Each machine is obtained by training the same underline learning algorithm, e.g. a decision tree, on a dataset drawn randomly with replacement from an initial training set. The size of this sub-sampled dataset is equal to the size of the original training set, but repetitions of points occur. Although there does not exist a well-accepted theory of Bagging, there are some theoretical and experimental indications that Bagging works better when the individual machine is "unstable" [4]. Instability in this case means that the machine changes a lot with changes of the training set.

In this paper we present a simple algorithm for learning stable machines which is based on recent results in statistical learning theory [8]. The algorithm is similar to Breiman's Bagging despite some important differences: (i) we let each machine use only a small part (i.e. 5-20%) of the original training data formed by random sampling with replacement, (ii) we consider the case where classification is done after thresholding the combination of the real valued outputs of each of the machines. The algorithm is named Subagging and is summarized in Figure 1.

[1] Boston University - Boston, USA, E-mail:*savina@math.bu.edu*
[2] Biowulf Technologies, New-York, NY, USA, E-mail: *andre@barnhilltechnologies.com*
[3] INSEAD, Fontainebleau, France E-mail: *theodoros.evgeniou@insead.fr*
[4] Dept. of Information Engineering, Univ. of Siena, Siena, Italy, E-mail: *pontil@dii.unisi.it*

Input: - A set of i.i.d. data $D_\ell = \{(\mathbf{x}_i, y_i) \in X \times \{-1,1\}\}_{i=1}^{\ell}$
- A learning algorithm $\mathcal{A} : (X \times \{-1,1\})^{\ell} \to (X)^{\mathbb{R}}$
(f_{D_ℓ} denotes the solution of $\mathcal{A}$ trained on D_ℓ)

For $t = 1, \ldots, T$
- Sub-sample r points from D_ℓ
  Let $D_r(t)$ the obtained set.
- Train $\mathcal{A}$ on D_ℓ to compute $f_{D_r(t)}$

Output: $\frac{1}{T}\sum_{t=1}^{T} f_{D_r(t)}$

Figure 1. Subagging algorithm.

In a Related work [8] we analyzed the characteristics of this algorithm. In particular, we developed probabilistic bounds on the distance between the empirical and the expected error that, unlike in the standard analysis of Bagging [4], are not asymptotic. These bounds formally suggest that Subagging can increase the stability of the learning machines when these are not stable and decrease otherwise. These results are reviewed in Section 2. In Section 3 we present experiments validating this theory using support vector machines and neural networks as the base machine. The results indicate that Subagging can significantly increase the stability of these machines whereas the test error is close (and sometimes better) to that of the same machines trained on the full training set.

Another important practical consequence of our theoretical results is that, since Subagging improves the stability, it can be easier to control the generalization error. In Section 4 we present experiments suggesting that the empirical error can indeed be a good estimate of the predictive performance of Subagging. This is a remarkable property which shows the advantage of our ensemble method over standard learning algorithms based on empirical risk minimization.

2 Theory

Let $D_\ell = \{(\mathbf{x}_i, y_i) \in X \times \{-1,1\}\}_{i=1}^{\ell}$ be the training set. A learning algorithm is a function $\mathcal{A} : (X \times \{-1,1\})^{\ell} \to (X)^{\mathbb{R}}$ which, given a training set D_ℓ, returns a real valued classifier $f_{D_\ell} : X \to \mathbb{R}$. Classification of a new input $\mathbf{x}$ is done by taking the sign of $f_{D_\ell}(\mathbf{x})$. For simplicity we consider only deterministic and symmetric algorithms although the theory can be extended to general algorithms by adding a randomizing preprocessing step. We denote by D_ℓ^i the training set obtained by removing point $(\mathbf{x}_i, y_i)$ from D_ℓ,

that is the set $D_\ell \setminus \{(\mathbf{x}_i, y_i)\}$. We use the following notion of stability from [3]:

Definition: We say that a learning algorithm is β_ℓ-stable with respect to a training sets of size ℓ if the following holds:

$$\forall i \in \{1, \ldots, \ell\}, \forall D_\ell, \forall (\mathbf{x}, y) : \ |f_{D_\ell}(\mathbf{x}) - f_{D_\ell^i}(\mathbf{x})| \leq \beta_\ell$$

Roughly speaking the output of a learning machine on a new (test) point $\mathbf{x}$ should not change more than β_ℓ when we train the machine with any training set of size ℓ and when we train the machine with the same training set but one training point (any point) removed. Bounds on stability for many algorithms were proved in [3].

It is possible to relate the stability of a learning algorithm to his generalization error, $E_{(\mathbf{x},y)}[\theta(-yf_{D_\ell}(\mathbf{x}))]$, where the expectation is taken w.r.t the unknown probability distribution generating the data and θ is the Heavyside function, $\theta(x) = 1$, if $x > 0$, and zero otherwise (for a discussion, see [7, 11, 3]). In order to state the results we also define for any given constant δ the empirical error R^δ_{emp} on function f to be: $R^\delta_{\text{emp}}(f) = \frac{1}{\ell}\sum_{i=1}^{\ell} \pi_\delta(-y_i f(\mathbf{x}_i))$, where the function $\pi_\delta(\xi)$ is 0 for $\xi < -\delta$, 1 for $\xi > 0$, and $\frac{\xi}{\delta} + 1$ for $-\delta \leq \xi \leq 0$ (a soft margin function). The following theorem from [3] expresses the relation between generalization and stability.

Theorem 1 *For any given δ, with probability $1 - \eta$ the generalization misclassification error of an algorithm that is β_ℓ stable is bounded by:*

$$R^\delta_{\text{emp}}(f_{D_\ell}) + 2\frac{\beta_\ell}{\delta} + \sqrt{\frac{1}{2\ell}\left(2\frac{\ell\beta_\ell}{\delta} + 1\right)^2 \ln(\frac{1}{\eta})}$$

We now discuss a similar generalization bound for an ensemble of machines where each machine uses only r points drawn randomly with the uniform distribution from the training data. To study such a system, let us introduce some more notations that incorporates the "random sub-sampling" effect. Let $F^r_{D_\ell} = E_{D_r}[f_{D_r}]$ be the expected combination of machines trained on sub-samples of size r. The expectation is taken with respect to the training data D_r of size r drawn uniformly from D_ℓ. The next theorem is from [8].

Theorem 2 *For any given δ, with probability $1 - \eta$ the generalization misclassification error of the expected combination $F^r_{D_\ell}$ of machines each using a sub-sample of size r of the training set and each having a stability β_r is bounded by:*

$$R^\delta_{\text{emp}}(F^r_{D_\ell}) + \frac{4r}{\ell}\frac{\beta_r}{\delta} + \sqrt{\frac{1}{2\ell}\left(\frac{4r\beta_r}{\delta} + 1\right)^2 \ln(\frac{1}{\eta})}$$

Proof (Sketch): The proof consists of bounding the stability of the Subagging procedure. To this end, it is sufficient to show that the stability of $F^r_{D_\ell}$ is upper bounded by $2\frac{r\beta_r}{\ell}$ and apply Theorem 2.1 afterwards.

This theorem holds for ensemble combinations that are theoretically defined from the expectation $E_{D_r}[f_{D_r}]$. It does not apply to a finite combination of machines computed by Subagging, $\frac{1}{T}\sum_{t=1}^{T} f_{D_r(t)}$. Yet, when T is large $R^\delta_{\text{emp}}(F^r_{D_\ell}) \approx R^\delta_{\text{emp}}(\frac{1}{T}\sum_{t=1}^{T} f_{D_r(t)})$ and when T increases the stability of the combined learning system tends to the stability of the expectation $E_{D_r}[f_{D_r}]$ which does not improve after T has passed a certain value. This value may correspond to the convergence of the finite sum $\frac{1}{T}\sum_{t=1}^{T} f_{D_r(t)}$ to its expectation w.r.t. D_r - See [8] for a formal discussion. Thus, the control of the test error by the empirical error is mainly due to the sub-sampling effect and not to the number of machines used in the combination. *That means that increasing T shouldn't improve this control as much as reducing the value of r*, a fact which we experimentally verified - See Section 3.

A consequence of Theorem 2 is that, since the stability of $F^r_{D_\ell}$ is smaller than $\frac{2r\beta_r}{\ell}$, if this stability is better than the stability of a single machine (which equals β_ℓ), the average of functions $f_{D_r(t)}$ provides a better bound. However, in the other case, the bound is worse and Subagging should be avoided. We have the following corollary:

Corollary 2.1 *If a learning machines is β_ℓ stable and $\frac{\beta_\ell}{\beta_r} < \frac{2r}{\ell}$, then combining these learning machines via Subagging does not provide a better bound on the difference between the test and empirical error. Conversely, if $\frac{\beta_\ell}{\beta_r} > \frac{2r}{\ell}$, then combining these learning machines leads to a better bound on the difference between the test and empirical error.*

This corollary says that combining machines via Subagging should not be used if the stability of the single machine is very good. However, it is not often the case to have a highly stable single machine, so typically Subagging improves stability. In such a situation, the bounds presented in this paper show that we have better control of the generalization error for combination of machines in the sense that *the empirical error is closer to the test error.*

At last notice that the bounds presented *do not* necessarily imply that the generalization error of Subagging is less than that of single machines (the r.h.s. of the bounds include both the empirical error and the stability), a remark which also been made for Bagging [4].

3 Experimental Results

We conducted a number of experiments using five datasets from UCI[5]: Breast-Cancer, Diabetis, German, Image, and Flare-Solar. We focused mainly on two learning machines: Support Vector Machines (SVMs) and Neural Nets. The goal of the experiments was to study how the test error and the absolute difference between test and training error behave with the "free parameters" of Subagging, i.e. with the number T of machines and the percentage P of sub-samples used by each machine, $P = \frac{r}{\ell} \times 100$. For both SVM and Neural Nets we found that Subagging decreases the above difference, thus increasing their stability. We now discuss these results and link them to the theoretical findings.

3.1 Subagging SVMs

We used SVMs [14] by using a Gaussian kernel. Each SVM was trained with a fixed value of the regularization parameter and variance of the Gaussian which were previously computed using 10-fold cross validation. Table 1 shows the average test error (top row in each cell) and the average absolute difference between test and training error (bottom raw in each cell) with their standard deviation as a function of the percentage P. The average was computed over 10 random splits of the overall datasets in testing and training. The number of machines combined by Subagging was equal to 30. The last column in Table 1 shows the results of the SVM trained on the whole dataset.

These results indicate that the difference between test and training error of one SVM using all training data is, except that for the Flare-Solar dataset, about double the same difference obtained by

[5] The dataset along with a short description are available from http://www.ics.uci.edu/~mlearn/MLRepository.html

Subagging with $P = 10\%$. At the same time, the test performance of one SVM alone is close on four datasets and better than Subagging on the Image dataset. In other words *the test performance is about the same, but the difference between test and training error is significantly smaller for Subagging than for a single SVM*. Note that the parameters of the SVM were optimized for the single SVM, so the performance of Subagging is not the optimal one.

The fact that the test error gets closer to the training error for Subagging can be explained by Theorems 1 and 2. Indeed if we denote by K the kernel of the SVM and by C the regularization parameter, the stability of an SVM can be bounded by $\frac{C \cdot \kappa}{2}$ where $\kappa = \sup_{x \in X} K(\mathbf{x}, \mathbf{x})$. Thus, because C was fixed to a constant in our experiments, the stability does not depend on the size of the training set[6]. In this case Theorem 2 gives a much tighter bound than Theorem 1.

Finally, we noticed that the number T of machines is not very important: combining only 10 SVMs gives already a good approximation of the "true average" (We will add a figure showing this in a future version of the paper).

Table 1. Subagging SVMs. Average test error and (below it) the average absolute difference between test and training error with their standard deviations for different values of the percentage P of the sub-samples. The last column shows the results for the SVM trained on the full dataset.

Dataset	5%	10%	20%	1SVM
Breast	28.5 ± 4.8	27.1 ± 4.6	27.0 ± 4.7	26.6 ± 4.8
	5.3 ± 4.4	5.6 ± 3.4	7.2 ± 4.3	9.0 ± 5.0
Diabetis	24.6 ± 1.9	23.5 ± 2.0	23.5 ± 2.0	23.3 ± 2.3
	2.6 ± 1.5	2.8 ± 1.4	3.1 ± 1.7	5.4 ± 1.8
German	26.2 ± 2.7	24.3 ± 1.9	23.8 ± 2.2	23.4 ± 1.7
	2.7 ± 1.4	2.6 ± 1.6	3.1 ± 1.8	6.7 ± 2.2
Image	8.9 ± 0.8	7.1 ± 0.8	4.7 ± 0.7	3.0 ± 0.6
	0.8 ± 0.6	0.7 ± 0.8	0.7 ± 0.7	1.7 ± 0.6
Solar	33.8 ± 2.3	34.0 ± 1.9	33.5 ± 2.7	34.9 ± 3.0
	2.5 ± 2.0	2.4 ± 1.9	2.5 ± 2.2	3.1 ± 1.9

3.2 Subagging Neural Nets

Neural Nets were trained with the conjugate gradient (see [12] for details). We used a three layers network architecture with ten hidden units. The initial momentum, the learning rate, the attenuate learning rate, the error limit and the iteration limit were fixed for all of the machines. Also, we did not apply any heuristics in order to find the initial parameters for the Neural Network - they were chosen randomly a priori. Table 2 shows the experimental results (the setting is the same as in Table 1).

Like in the case of SVMs, the results indicate that Subagging decreases the difference between test and training error. The theoretical explanation is more delicate in this case, because we do not know theoretically the stability of Neural Nets. However, this does not reduce the interest of the discussion since the former theoretical results hold for very general algorithms as soon as uniform stability can be defined.

We also noticed in a separate series of experiments that the above difference is more sensitive to the size of the sub-sample than to the number of the machines. In particular the test error decreases and it stabilizes when the number of machines combined is greater than 50 (We will add a figure showing this in a future version of the paper).

[6] The discussion becomes more tricky if we let C be a function of the size of the training set ℓ, see [8].

Table 2. Subagging Neural Nets. Average test error and (below it) the average absolute difference between test and training error with their standard deviation for different values of the percentage P of the sub-samples. The last column shows the results for the Neural Net trained on the full dataset.

Dataset \ P	5 %	10%	20%	1NN
Cancer	26.7 ± 5.8	27.9 ± 3.7	28.6 ± 3.4	32.6 ± 5.7
	5.3 ± 4.5	6.4 ± 3.6	11.0 ± 5.2	30.1 ± 5.5
Diabetis	24.3 ± 2.0	24.2 ± 2.5	24.3 ± 2.6	28.6 ± 1.3
	3.2 ± 2.3	5.2 ± 2.9	8.2 ± 2.5	24.3 ± 1.7
German	24.5 ± 2.2	24.6 ± 2.8	23.7 ± 1.9	29.9 ± 2.7
	2.9 ± 2.0	4.9 ± 3.0	8.2 ± 3.3	27.7 ± 2.9
Image	8.8 ± 0.8	5.7 ± 0.6	4.5 ± 1.8	9.6 ± 18.2
	1.5 ± 1.6	1.5 ± 2.3	1.8 ± 2.5	7.8 ± 18.9
Solar	35.4 ± 1.7	35.4 ± 2.5	35.0 ± 1.6	33.8 ± 1.7
	3.0 ± 1.8	3.7 ± 2.0	3.6 ± 2.0	2.8 ± 2.2

3.3 Applying Subagging to Other Machines

We also carried our experiments with Decision Trees and $k-$Nearest Neighbors (not shown here). For $k-$Nearest Neighbors the experiments done showed that Subagging does not decrease the difference between test and training error. The $k-$Nearest Neighbors is known to be a stable algorithm [4] and his stability scales as $\frac{1}{\ell}$ [7], which means (with Corollary 2.1) that Subagging shouldn't help. Experiments and theory are thus consistent. In the case of decision trees we found the same trend as with SVMs and Neural Nets.

4 Model Selection Using Small Sub-samples

The bound in Theorem 2 implies that the empirical error of Subagging can be a good indicator of the expected error of the combinations of the machines, especially for small sub-sample sizes. For example, in the case of SVMs the stability does not depend on the variance of the kernel. Thus, if we believe Theorem 2 is predictive, we may just try to select the best value of the variance by minimizing the empirical error.

To further explore how the empirical error of Subagging can be used for model selection, we performed one more series of experiments. We focused only on two datasets, namely Breast-Cancer and Diabetis.

We first compared one single SVM and a Subagging combination of 30 SVMs each using a small percentage of the original training set. Each of these two architectures was trained on different values of the regularization parameter and variance of the Gaussian kernel.

Figures 2 and 3 show test and training error as a function the variance of the Gaussian kernel (for the same fixed value of the regularization parameter used in Section 3.1) for Breast-cancer and Diabetis datasets respectively - notice that the scales of the y-axis of the plots are different for single and Subagging SVMs. For both datasets, the training error of the SVM alone monotonically increases with the variance of the Gaussian. Then, as expected, in this case the training error cannot be used for model selection. On the other hand, in the case of Subagging, the training error has a minimum fairly close to the minimum of the test error.

Figures 4 and 5 show test and training error for our two datasets as a function of the logarithm of the regularization parameter. In these figures the test error present a large "U" shape. However, in the case

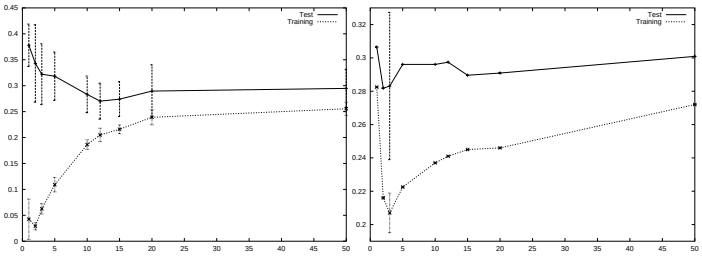

Figure 2. *Breast cancer dataset: Test and Training error as a function of the variance of the Gaussian kernel. (Left) One single SVM, and (Right) Subagging 30 SVMs trained on 10% of the data.*

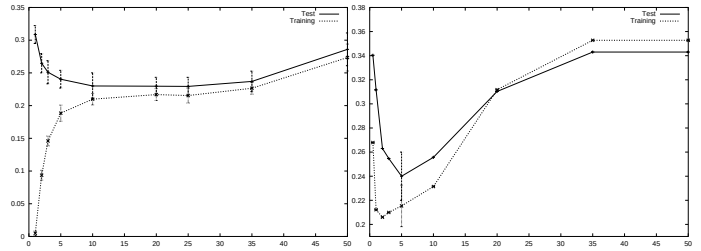

Figure 3. *Diabetis dataset: Test and Training error as a function of the variance of the Gaussian kernel. (Left) One single SVM, and (Right) Subagging 30 SVMs trained on 5% of the data.*

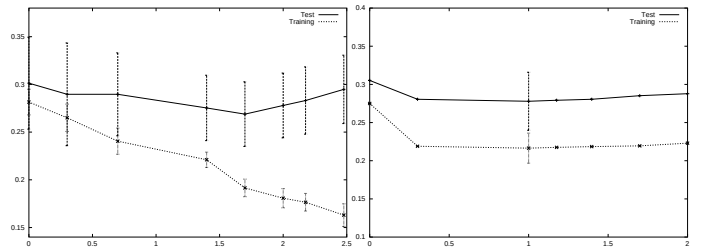

Figure 4. *Breast cancer dataset: Test and Training error as a function of the logarithm of the regularization parameter. (Left) One single SVM, and (Right) Subagging 30 SVMs trained on 10% of the data.*

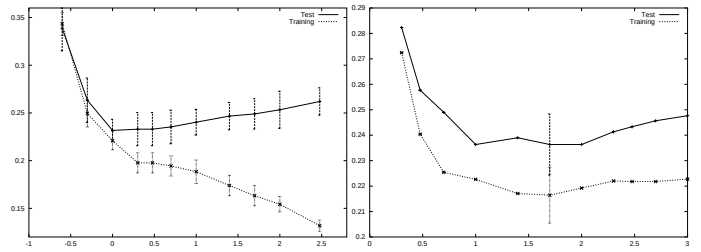

Figure 5. *Diabetis dataset: Test and Training error as a function of the logarithm of the regularization parameter. (Left) One single SVM, and (Right) Subagging 30 SVMs trained on 5% of the data.*

of Subagging, it is still possible to well locate the minimum of the test error by looking at the minimum of the training error. Thus, the experiments indicate that *for Subagging by small sub-samples the training error can effectively be used for model selection*. Notice also, that in all cases the test error for Subagging is only slightly bigger than the test error of a single SVM.

In the case of Neural Nets, we tried to select the number of hidden units of the Neural Nets. Table 3 shows the average test and training error for one single Neural Nets and Subagging Neural Nets with $P = 5\%$ for different numbers of hidden units. Although this experiment is only preliminary (we used only four different values for the hidden unites) it indicates that the training error can predict the best choice of the parameter i for Subagging.

The fact that, in the case of Subagging with small sub-samples, the empirical error can be used for model selection is, as far as we know, a quite surprising result. For standard learning algorithms, instead, the empirical error is an increasing function of the model complexity, i.e. the complexity of the hypothesis space used for learning (measured for example through the VC-dimension of that space). As a consequence, to perform model selection, one typically needs to use other error estimates such as the leave-one-out error [7]. However, the computation of these error estimates is, in general, very expensive. Then, the fact that just the training error is good enough to perform model selection is a very pleasant property of out learning algorithm.

Table 3. Results on Neural Nets with different trained on different numbers of hidden units. Each cell shows the test error and (below it) empirical error with their standard deviations.

H. Units	0	2	5	10
Cancer	28.8 ± 3.3	30.1 ± 2.5	35.5 ± 4.3	32.6 ± 5.7
(1NN)	21.9 ± 1.3	17.1 ± 1.8	5.9 ± 0.9	25.0 ± 0.9
Cancer	$\mathbf{26.6} \pm 3.1$	32.5 ± 3.2	28.4 ± 3.4	26.7 ± 5.8
(Suggabing)	$\mathbf{22.4} \pm 1.6$	24.4 ± 1.3	23.4 ± 1.6	23.3 ± 1.5
Diabetis	23.6 ± 2.5	26.2 ± 3.2	28.4 ± 1.0	28.6 ± 1.3
(1NN)	20.4 ± 2.3	19.6 ± 5.6	10.7 ± 2.3	4.3 ± 1.5
Diabetis	24.6 ± 2.4	25.2 ± 1.6	25.0 ± 1.9	$\mathbf{24.3} \pm 2.0$
(Subagging)	22.8 ± 2.5	22.6 ± 1.7	21.9 ± 2.5	$\mathbf{21.6} \pm 2.1$

5 Conclusions and Future Work

We presented Subagging, a learning algorithm which combines the output of real-valued classifiers each trained on small random subsamples of the original training set.

We reported experiments using a number of learning machines and datasets where we studied the characteristics of this algorithm. The results indicate that Subagging improves the stability of unstable machines while the test error is close (and sometimes better) than that of these machines trained on the whole dataset. We also showed that, in the case of Subagging, the empirical error of can be used for model selection, unlike the case of single machines trained using all the training data. These experimental findings support the theoretical results discussed in Section 2 and suggest that Subagging is a viable technique for learning stable machines.

In the future it could be interesting to further study how the difference between test and training error of Subagging depends on the size of the sub-sampling, and how to experimentally infer from this the form of the stability of the underline learning machine as a function of the size of the training data. Another future direction for re-

search is to explore how to extend our theoretical framework to other ensemble methods like Boosting.

REFERENCES

[1] E. Bauer and R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning*, 36:105–142, 1999.

[2] S. Boucheron, G. Lugosi, and P. Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 16:277–292, 2000.

[3] O. Bousquet and A. Elisseeff. Stability and generalization. To be published in *Journal of Machine Learning Research*, 2002.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[5] L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24(6):2350–2383, 1996.

[6] P. Cunningham, J. Carley and S. Jacob. Stability Problems with Artificial Neural Network and the Ensemble Solution. , 1999.

[7] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. Springer, New York, 1996.

[8] T. Evgeniou, M. Pontil, and A. Elisseeff. . Leave one out error, stability, and generalization of voting combinations of classifiers. Preprint, 2001.

[9] Y. Freund and R. Schapire. A decision-theoretic generalization of online learning and an application to Boosting. *Journal of Computer and System Sciences*,55(1):119–139, 1997.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Tech. report, Department of Statistics, Stanford University, 1998.

[11] M. Kearns and D. Ron. Algorithmic stability and sanity check bounds for leave-one-out cross validation bounds. *Neural Computation*, 11(6):1427–1453, 1999.

[12] W.H. Press et al.. Numerical Recipes in C. *Cambridge University Pressb*, 1986.

[13] R. Shapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 1998.

[14] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.