Variance Optimized Bagging

Philip Derbeko[1], Ran El-Yaniv[1], and Ron Meir[⋆2]

[1] Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel, {philip,rani}@cs.technion.ac.il

[2] Electrical Engineering Department, Technion - Israel Institute of Technology, Haifa 32000, Israel, rmeir@ee.technion.ac.il

Abstract. We propose and study a new technique for aggregating an ensemble of bootstrapped classifiers. In this method we seek a linear combination of the base-classifiers such that the weights are optimized to reduce variance. Minimum variance combinations are computed using quadratic programming. This optimization technique is borrowed from Mathematical Finance where it is called Markowitz Mean-Variance Portfolio Optimization. We test the new method on a number of binary classification problems from the UCI repository using a Support Vector Machine (SVM) as the base-classifier learning algorithm. Our results indicate that the proposed technique can consistently outperform Bagging and can dramatically improve the SVM performance even in cases where the Bagging fails to improve the base-classifier.

1 Introduction

This paper is concerned with Bagging (**B**ootstrap **Agg**regation) of classifiers. Bagging works by applying a learning algorithm on a number of bootstrap samples of the training set. Each of these applications yields a classifier. The resulting pool of classifiers is combined by taking a uniform linear combination of all the constructed classifiers. This way a new (test) point is classified by the "master" classifier by taking a majority vote between the classifiers in the pool.

Since its introduction in [1] Bagging attracted considerable attention, and together with Boosting is considered to be among the most popular techniques for constructing and aggregating an ensemble of classifiers. A number of theoretical and experimental studies attribute the success of Bagging to its ability to reduce variance; see e.g. [2], [3] and [4].

We ask and attempt to answer the following question: Is it possible to improve the performance of Bagging by optimizing the combined classifier over all weighted linear combinations so as to reduce variance? In the context of regression such a scheme is particularly appealing since the bias of a normalized weighted combination is unchanged if the original biases are all the same. Although this result is not directly related to classification it may be suggestive, and if variance reduction of the base-classifier is one of the main effects of Bagging, one can expect that this question should be answered affirmatively.

⋆ The research was supported by the fund for promotion of research at the Technion and by the Ollendorff center.

Indeed, we provide strong evidence that a Variance Optimized Bagging, which we term for short ***Vogging***, consistently improves on Bagging. The main ideas behind the new technique are borrowed from Mathematical Finance. Specifically, we import the basic ideas of Markowitz *Mean-Variance Portfolio Theory* [5, 6] that is used for generating low variance portfolios of financial assets, and use it in our context to construct optimized "portfolios" of bootstrapped classifiers.[1]

This paper is organized as follows. In Section 2 we briefly overview the basic ideas of Markowitz portfolio theory. We then use these ideas and introduce the new Vogging technique in Section 3. In Section 4 we discuss our experimental design and present our results in Section 5. Related work is discussed in Section 6 and finally, in Section 7, we summarize our conclusions and suggest directions for further research.

2 Markowitz Mean-Variance Portfolio Optimization

In this section we provide a brief overview of the main ideas of the Markowitz Single-Period Mean-Variance Portfolio optimization technique. These ideas set the path for a most influential theory in mathematical finance. They will later be utilized in our new classifier aggregation technique.

The *single period* Markowitz algorithm solves the following problem. We consider m assets (e.g. stocks, bonds, etc.) $S_1, \ldots, S_m$. We are given: (i) A predicted expected monetary return r_i for each asset S_i; (ii) A predicted standard deviation σ_i of the return of S_i; and (iii) The $m \times m$ covariance matrix Q with $Q_{ii} = \sigma_i$ and $Q_{ij} = \rho_{ij}\sigma_i\sigma_j$ where ρ_{ij} is the correlation coefficient between the returns of S_i and S_j.

A portfolio is a linear combination of assets. It is given by a vector $\mathbf{w}$ of m weights $\mathbf{w} = (w_1, \ldots, w_m)$ with $\sum_i w_i = 1$. The expected return of a portfolio $\mathbf{w}$ is $\sum_i w_i r_i$. The risk of a portfolio is traditionally measured by its variance $\sigma^2(\mathbf{w})$,

$$\sigma^2(\mathbf{w}) = \sum_{i,j} w_i w_j Q_{ij} = \mathbf{w}^t Q \mathbf{w}.$$

It is assumed that investors are interested in portfolios that yield high returns but are averse to large variance. The exact risk aversion pattern of an investor is modeled via a utility function. Nevertheless, an empirical fact (which is backed up by economic theories) is that the return of an asset typically trades-off its variance; that is, assets with large average return tend to exhibit large variance and vice versa.

The output of the Markowitz algorithm is a set of portfolios with expected return greater than any other with the same or lesser risk, and lesser risk than any other with the same or greater return. This set is called the *efficient frontier*. The efficient frontier is conventionally plotted on a curve with the standard

[1] Thirty-eight years after Markowitz published his paper "Portfolio Selection" [5] he shared a Nobel Prize with Miller and Sharpe for his study that has become a well established theory of portfolio selection.

deviation (risk) on the horizontal axis, and the expected return on the vertical axis. An efficient frontier illustration is given in Figure 1[2]. A useful feature of the single period mean-variance portfolio problem is that it is soluble using quadratic programming.

Using the efficient frontier an investor seeking to invest in an "optimal" portfolio should choose one that lies on the frontier curve. The exact portfolio will be chosen using his/her personal utility function. A particular "off-the-shelf" recommended utility function was proposed by Sharpe and is called the Sharpe Ratio [7]. Sharpe's ratio is a risk-adjusted measure of return that divides a portfolio's return in excess of the riskless return by the portfolio's standard deviation. Specifically, let R_0 be the return of a risk-free asset (i.e. cash or treasury bills) and let $(R(\mathbf{w}), \sigma(\mathbf{w}))$ be the return and risk pair of a portfolio $\mathbf{w}$. Then, the Sharpe ratio of $\mathbf{w}$ is

$$\text{Sharpe}(\mathbf{w}) = \frac{R(\mathbf{w}) - R_0}{\sigma(\mathbf{w})}. \tag{1}$$

3 Bagging and Vogging

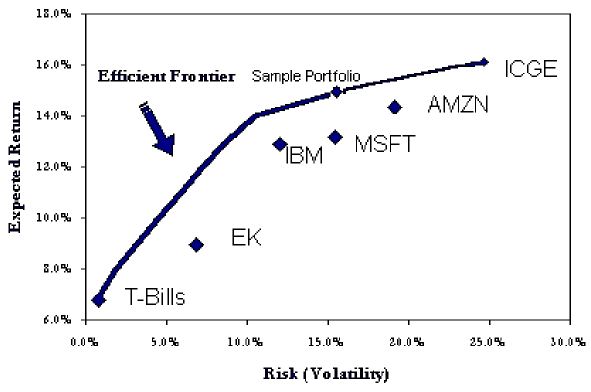

Fig. 1. Efficient Frontier illustration

Let $\mathcal{H}$ be a binary hypothesis class of functions from the input space $\mathcal{X}$ to $\{\pm 1\}$ and let $S = (x_1, y_1), \ldots, (x_n, y_n)$ be a training sample where $x_i \in \mathcal{X}$ and $y_i \in \{\pm 1\}$. *Bagging* works as follows. We generate T bootstrap samples $B_1, \ldots, B_T$ from S. Each bootstrap sample is generated by sampling with replacement n points from S. We train T classifiers $h_j \in \mathcal{H}$, $j = 1, \ldots, T$, such that h_j is trained using the sample B_j. Given a new point $x \in \mathcal{X}$ we predict that its label is sign $\left(\frac{1}{T}\sum_j h_j(x)\right)$. Thus, the aggregated classifier is simply a threshold applied on a uniform average of the base classifiers h_j.

The idea in ***Vogging*** (Variance Optimized Bagging) is to optimize a linear combination of classifiers so as to aggressively reduce variance while attempting to preserve a prescribed accuracy. As in ordinary Bagging we generate T bootstrap samples from the training set and induce T hypotheses from $\mathcal{H}$ using these samples. Here again, let h_j denote the resulting hypotheses. Let $A_j(B_i)$ denote the empirical accuracy achieved by h_j on the sample B_i. Let

[2] A nice Java applet computing the efficient frontier of some familiar assets can be downloaded at http://www.duke.edu/ charvey/applets/EfficientFrontier/frontier.html.

$\bar{A}_j = \frac{1}{T-1}\sum_{i\neq j} A_j(B_i)$ be the average empirical accuracy over all the other bootstrap samples. Since each Bootstrap sample B_j only contains a fraction of the data (on average, approximately 63%), we can view $\bar{A}_j$ as a proxy for a truly unbiased estimation of the error (more sophisticated "out-of-bag" methods can be considered, as discussed in Section 7).

Consider the (column) vectors $\mathcal{A}_j = (A_j(B_1), \ldots, A_j(B_T))$, $j = 1, \ldots, T$, and let $\bar{\mathcal{A}}$ be their average, $\bar{\mathcal{A}} = \frac{1}{T}\sum_{j=1}^{T} \mathcal{A}_j$. Let Q be the empirical covariance matrix of these vectors,

$$Q = \frac{1}{T-1}\sum_{j=1}^{T}(A_j - \bar{A})(A_j - \bar{A})^t \tag{2}$$

Using the empirical accuracies and covariance matrix Q we now employ the Markowitz algorithm to estimate the efficient frontier of combined minimum-variance "portfolios" of base-classifiers and use the classifier with the highest Sharpe ratio (see below). Specifically, we estimate the dynamic range of achievable accuracies using the end points $\min_j \bar{A}_j$ and $\max_j \bar{A}_j$ and take k uniformly spread points, $a_1, \ldots, a_k$, in this interval. Each a_i is an achievable empirical accuracy by some linear combination of classifiers. Using the a_i's we interpolate the efficient frontier as follows. For each $a \in \{a_i\}$ we solve the following quadratic program (QP) with linear constraints:

$$\begin{array}{rc} \text{minimize (over } \mathbf{w}\text{):} & \frac{1}{2}\mathbf{w}^t Q \mathbf{w} \\ \text{subject to:} & (\bar{A}_1, \ldots, \bar{A}_T)^t \mathbf{w} \geq a \\ & \sum_j w_j = 1, \ \mathbf{w} \geq 0. \end{array}$$

That is, by solving QP, we attempt to minimize variance while keeping the accuracy sufficiently large.

Remark 1. The solution of QP with a lower bound accuracy constraint a, if it exists, is a weight vector $\mathbf{w}$ that corresponds to a mean accuracy and variance pair (a', σ^2) and a' may be larger than a.

In the Markowitz-Sharpe framework, in order to compute the weighted combination with the largest Sharpe ratio we need to use the return of a "riskless" asset (see Eq. (1)). The best analogy in our context is the expected accuracy of the trivial classifier that always predicts according to the label of the largest class in the training set. We call this classifier the *baseline* classifier. In Figure 2 we provide pseudo-code of the Vogging learning algorithm. The output of the algorithm is a single classifier based on the weighted combination that achieved the highest Sharpe ratio. We call this classifier the *Sharpe Classifier.* The motivation for using the Sharpe classifier is purely heuristic. While we would like to use a classifier with a small risk, this would make little sense if the variance of the classifier is very large. In order to reach a compromise between the risk and variance, we select a classifier with a small risk, subject to a constraint that its standard deviation is no too large. Eq. (1) provides an approximate implementation of this idea. Note also that a similar type of argument is used in the construction of the classic Fisher discriminant function.

```
Input:
  1. $T$ (number of bagged classifiers)
  2. $k$ (number of efficient frontier points)
  3. $S = (x_1, y_1), \dots, (x_n, y_n)$ (training set)
  4. $\mathcal{H}$ (base classifier hypothesis class)
Training:
  1. Generate $T$ bootstrap samples, $B_1, \dots, B_T$ from $S$
  2. Train $T$ classifiers $h_1, \dots, h_T$ such that $h_j \in \mathcal{H}$ is trained over $B_j$
  3. Evaluate $\bar{A}_j$, for all $j = 1, \dots, T$; evaluate $Q$ (see Eq. (2))
  4. Choose $k$ uniformly spread points $a_1, \dots, a_k$ in $[\min_j \bar{A}_j, \max_j \bar{A}_j]$
  5. Solve $k$ instances of QP (Eq. (3)) with the accuracy constraints $a_1, \dots, a_k$. For
     $i = 1, \dots, k$, let $\mathbf{w}_i$ and $(a_i', \sigma_i)$ be the resulting weight vector and mean-variance
     pair corresponding to $a_i$.
  6. Let $p_0$ be the proportion of the larger class in $S$
Output: "Vogging weight vector" $\mathbf{w}_{i^*}$ with $i^* = \arg\max_i \frac{\bar{a}_i' - p_0}{\sigma_i}$
```

Fig. 2. Pseudo-code for Vogging learning algorithm

4 Experimental Design

Our main goal in these experiments is to analyze and better understand the new Vogging technique and compare its performance to ordinary Bagging.

Many previous studies of Bagging considered as their base-classifiers inductive learning algorithms such as decision trees, neural networks and naive Bayes; see e.g. [1], [8], [9], [10], [8], [2] and [11]. As argued by [1] and [4], Bagging becomes effective when the base-classifier is unstable; intuitively this means that its decision boundaries significantly vary with perturbations of the training set.

We chose to use a Support Vector Machine (SVM) as our base classifier; see [12] and [13]. While SVM's are considered to be rather stable classifiers, even an SVM classifier exhibits instabilities when trained with small samples, especially when polynomial kernels are used. Since our main focus here is on situations where only a small amount of data is available, in all experiments described below we always used 30% of the available labeled samples to train our classifiers while leaving the rest of the data for testing. While Support Vector Machiens have been widely used for many problems, and shown to yeild state-of-the-art results, their behavior for particularly small data sets has not been thoroughly investigated. In ongoing work we are looking at other classifiers (including decision trees and neural networks). It should be emphasized that for small data sets variance is known to be a major problem, and thus we expect that variance reduction techniques should be particularly useful in this case. In fact, several exact calculations support this observation [15, 16], stressing the particular advantage of sub-sampling.

The new algorithm was tested on a number of datasets from the UCI repository [17]. Table 1 provides some essential properties of the datasets used. Note that the baseline (i.e. the proportion of the largest class in the training set) of each dataset is used for computing the Sharpe-ratio (Step 6 in the algorithm pseudo-code). In each experiment we used 10-fold cross-validation. Each fold consisted of a 30%-70% random partition where the 30% portion was used for training. The remaining 70% was used solely for testing, and was in no way accessible to the learning algorithm, for example, while the Ion dataset contains 351 labeled instances, in each of our folds we only used 105 labeled instances for training. Following [1] we generated, in most cases, $T = 50$ bootstrap samples (and 50 base-classifiers) from each training set. Due to the computational intensity, for the larger sets we generated 25 bootstrap samples[3]. In all experiments we used a polynomial kernel SVM with degree 20. The polynomial kernel is particularly convenient to use in our method due to its relative instability compared to other popular kernels such as RBF and linear.

In most of the experiments we report on the performance of the following classifiers: (i) The Vogging classifier; (ii) The Bagging classifier; (iii) The "*full-set*" base-classifier, which is trained over the entire training set.

5 Results

As an illustration of the proposed algorithm we first present one experiment in some detail. In Figure 3 we depict the training results of a single fold of the Vogging algorithm on the Voting dataset (130 training examples). The figure shows the mean-variance points corresponding to the observed accuracy and variance (estimated based on the training) of 50 base-classifiers. On the left part of the figure we see the efficient frontier and the Sharpe classifier (on the frontier). As can be seen, the top composite classifiers on the efficient frontier achieve somewhat smaller training accuracy than the best base-classifiers in the pool, but the composite classifier show noticeable reduction in standard deviation. Unlike financial assets, which usually exhibit a trade-off between return and variance (see illustration in Figure 1), the training

Dataset	Instances (training set size)	Attributes	Baseline
Voting	435 (130)	16	0.61
Diabetes	768 (230)	8	0.65
Ion	351 (105)	34	0.64
Sonar	208 (62)	60	0.53
Breast	683 (204)	10	0.65
WDBC	569 (170)	30	0.62
Credit-G	653 (195)	15	0.54
Tic-Tac-Toe	958 (287)	9	0.65

Table 1. Some essential details of the datasets used. The "Baseline" attribute is the trivial accuracy that can be achieved (proportion of the largest class in the training set)

[3] In [2] 25 bootstrap samples were used in all experiments.

Table 2. 10-fold cross-validated mean/std error performance comparison between Vogging, Bagging,the full-sample classifier and Vogging advantage over Bagging; see also Figure 5

DATASET (TRAINING SET SIZE)	VOGGING	BAGGING	FULL-SAMPLE BASE-CLASSIFIER	VOGGING ADVANTAGE
VOTING (130)	13.11±4.12	23.90±11.31	37.22±12.18	45.15%
DIABETES (230)	33.46±1.46	35.55±1.10	42.24±14.14	5.88%
ION (105)	15.89±2.37	29.51±10.21	32.64±15.31	46.16%
SONAR (62)	38.36±4.42	45.96±7.00	*40.71±5.67	16.53%
BREAST (204)	4.97±1.72	19.12±4.04	23.68±11.61	74.00%
WDBC (170)	22.76±8.56	26.12±6.67	36.77±17.05	12.86%
CREDIT-G (195)	40.41±5.35	46.22±1.00	48.41±8.29	12.57%
TIC-TAC-TOE (287)	32.33±3.37	36.30±7.27	51.77±10.98	10.93%
AVERAGE	25.16±3.92	32.84±6.08	39.18±11.9	28.01%

performance of the base-classifiers do not exhibit this trade-off and the better (high accuracy) classifiers also have smaller variance.[4]

In Figure 4 we see the final 10-fold cross-validation average accuracy of Vogging and Bagging on the Voting dataset. On the top left corner we depict the average accuracy and standard deviation of the Vogging classifier To the right of the Vogging classifier we see the Bagging classifier. The layered cloud of circles that fill the bulk of the figure is the test performance of all the 50×10 base-classifiers that were generated during the entire 10-fold experiment. Evidently, the Vogging classifier is significantly better both in terms of accuracy and variance. We should emphasize that the tiny circles (depicting base-classifiers) do not represent cross-validated performance. Interestingly, a large fraction of the base-classifiers converged to the baseline classifier performance (61%), while another subset reduced to the counter baseline classifier (39%).

It is interesting to examine the components of the aggregated classifiers on the frontier. In Figure 3 we identify the 9 largest components of the Sharpe classifier. These components are numbered 1–9 in the figure (in a decreasing order of their weights in the composite classifier). Although the weights are diversified between more classifiers, these 9 classifiers hold most of the weight. It is evident that lower accuracy base-classifiers are included with large weights (e.g. classifier 2 has a weight of 0.15).

In Figure 5 we see 10-fold cross-validated performance comparison on 8 datasets between Vogging, Bagging and the full-sample classifier. These results and the relative advantage of Vogging over Bagging are numerically presented in Table 2. In all these results the base-classifier is a degree 20 polynomial kernel SVM. In Table 2 the last column summarizes relative error improvement of

[4] We could generate a (training set) behavior more similar to this financial assets' pattern by aggressively over-fitting the base-classifiers.

Vogging over Bagging given by

$$\frac{Err(\text{Bagging}) - Err(\text{Vogging})}{Err(\text{Bagging})}.$$

The asterisk in one entry in the second last column corresponds to a case where Bagging could not improve on the full-sample base-classifier. Note that in all cases Vogging outperformed both the Bagging and full-sample classifiers.

We note the following. First it is striking that Vogging achieves higher accuracy than Bagging using polynomial kernel SVMs. Overall we see an error improvement average (over these datasets) of 28% over Bagging[5]. In most cases the standard deviations exhibited by Vogging was significantly smaller than Bagging (and the other base-classifiers). Overall, we see a 35% average variance reduction improvement over Bagging.

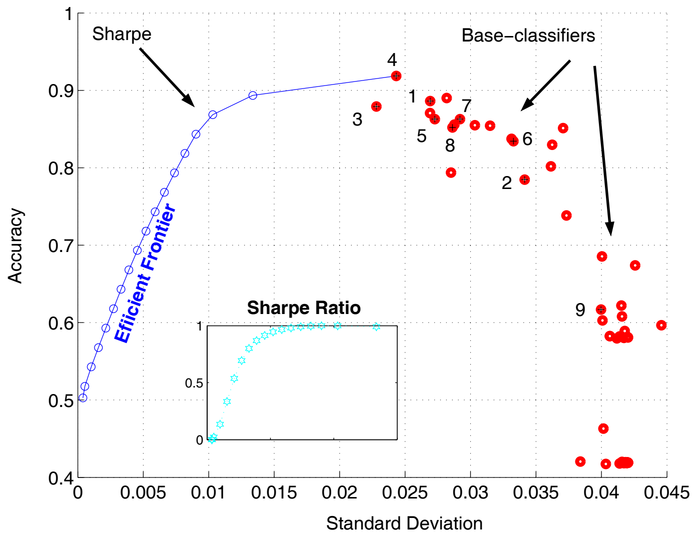

Fig. 3. Voting dataset, estimated training accuracy and standard deviation of 50 base-classifiers (SVM, polynomial kernel of degree 20 with $C = 100$); the resulting (interpolated) efficient frontier is marked; The Sharpe classifier (on the frontier) is pointed by an arrow; the inner graph shows the Sharpe ratios of the various classifiers on the frontier (in the order of their appearance on the frontier)

The absolute errors reported here are not directly comparable to other published results on Bagging performance on the same datasets in [8, 9, 11], which used much larger training set sizes (e.g. most of these studies used 90% of the data for training). In general, our absolute errors are larger than those reported

[5] calculated over polynomial kernels only.

in these studies. In the full version of the paper we will include a comparison with other known algorithms.

6 Related Work

Bagging falls within the sub-domain of "ensemble methods". This is an extensive domain whose coverage is clearly beyond the scope of this paper; see e.g. [18, 19] and references therein. Bagging [1] and Boosting [20] are among the most popular re-sampling ensemble methods that generate and combine a diversity of classifiers using the same learning algorithm for the base-classifiers. Classical boosting algorithms like Adaboost are considered stronger than bagging on noise-free data. They also lie on more solid theoretical grounds in the form of convergence, consistency and generalization results, and have many more extensions and variations than bagging.[6] However, there are strong empirical indications that Bagging is much more robust than boosting in noisy settings; see e.g. [11]. An intuitive explanation for this resilience of bagging suggested by Dietterich is that the bootstrap samples avoid a large fraction of the noise while still generating a diverse pool of classifiers. We note that new regularized and improved boosting algorithms that can resist noise were recently proposed.

There have not been to-date very many theoretical analyses of Bagging. A recent paper [4] proves that bagging with a non-stable base-classifier (where stability is defined in an asymptotic sense similar to statistical consistency) will reduce variance. This paper also analyzes a sub-sampling variant of bagging called sub-agging. Other interesting discussions and analyses of bagging can be found at [21], [2] and [3].

The paper [22][7] proposes a heuristic method for generating weighted average of bagged classifiers. The proposed weighting for a base-classifier is a function of its relative accuracy advantage over the other classifiers where these quantities are estimated over "out-of-bag" training samples. According to this paper, in a comparative empirical evaluation using ID3 decision tree learning as the base-classifier, this weighted bagging technique outperformed ordinary bagging and boosting on the majority of UCI datasets that were examined.

The idea of employing Markowitz portfolio optimization for ensemble construction was proposed by [23] as a method for avoiding standard parameter tuning in neural networks (e.g., based on cross-validation). Specifically, Mani proposed to train a pool of neural nets with a diversity of parameters and then combine them using a Markowitz optimized linear combination. He did not address the issue of choosing a weighted combination from the efficient frontier. As far as we know, this idea was never tested. Along these lines, in the context of regression [24] propose to consider a pool of predictors with a diversity of values to their parameters so as to span a reasonable range. Then they propose to use

[6] See the Boosting web site at http://www.boosting.org/.

[7] The exact date of this unpublished manuscript is unknown to the authors; we estimated it to be 2000.

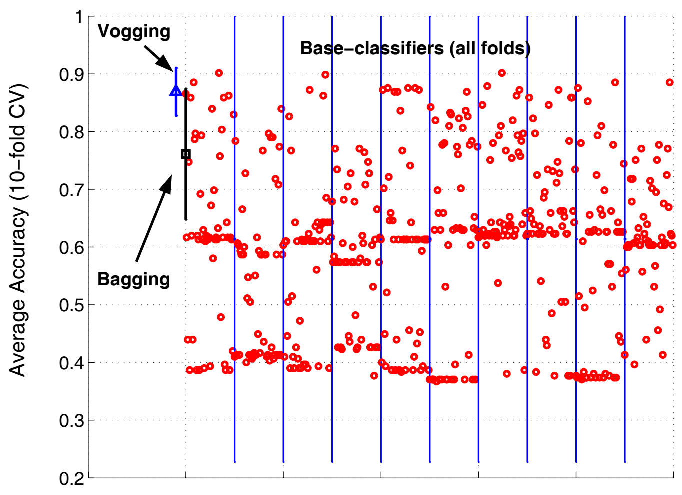

Fig. 4. Voting dataset, 10-fold test accuracy (and standard deviaition represented via error bars) of the Vogging classifier, Bagging and 50×10 base-classifiers. Each of these base-classifier accuracies is not cross-validated and appeared at a single fold. The base-classifiers are grouped by the folds, each vertical strip contains the base-classifiers of a single fold.

an "out-of-bootstrap" sampling technique to estimate least-squares regression weights of members of the pool.

7 Conclusions and Open Directions

In this paper we proposed a novel and natural extension of Bagging that optimizes the weights in linear combinations of classifiers obtained by Bootstrap sampling. The motivation and main ideas of this new weighted bootstrap technique are borrowed from mathematical finance where it is used to optimize financial portfolios. The proposed algorithm attempts to aggressively reduce the variance of the combined estimator, while trying to retain high accuracy.

We presented the results of a number of experiments with UCI repository datasets. Using an SVM as the base-classifier we focused on situations where the training sample size is relatively small. Such cases are of particular interest in practical applications. Our results indicate that the new technique can dramatically improve the (out of sample) test accuracy and variance of the base-classifier and of Bagging. Although these results are striking, due to the moderate scope of our experimental study we view them only as a proof of concept for the proposed method.

We concentrated on small training sample scenarios where the effects of estimation variance are particularly harmful to classification accuracy. It appears that the utilization of Bootstrap samples allowed us to obtain a reliable esti-

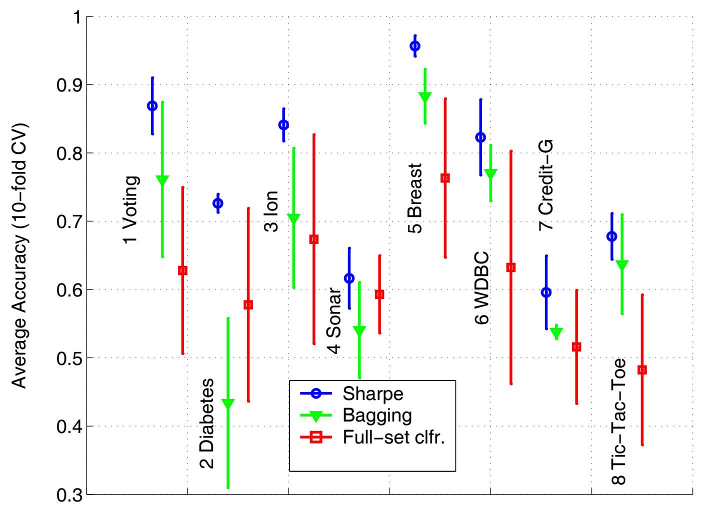

Fig. 5. 10-fold cross-validation test results on 8 datasets using a degree 20 polynomial kernel SVM base-classifier. Each dataset entry consists of 4 mean/std error bars corresponding (from left to right) to Vogging, Bagging and full-set (training set) classifier. Dataset names appear to the left of their error bars. Numerical summary of these results appear in Table 2

mates of the variance (and covariance), a parameter which cannot be reliably estimated from the same set of points used to train the classifiers. More sophisticated estimation techniques can possibly improve the estimation accuracy and the algorithmic efficiency. For instance, techniques similar to those used by [24] and [22] can potentially improve the sampling component of our algorithm.

To the best of our knowledge the above results are the first reported experimental evidence of a successful use of SVM as the base-classifier in Bagging.

Instability of the base-classifier learning algorithm is a major factor in the ability to generate diversity in the form of anti-correlations between the various base-classifiers in the pool, which is the key for variance reduction. Therefore, one can expect that the relative advantage of our technique will increase if used with more unstable base-classifiers such as decision trees and neural networks. We plan to investigate this direction.

References

1. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
2. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.
3. J. Friedman and P. Hall. On bagging and nonlinear estimation, 2000. Preprint. URL: http://www-stat.stanford.edu/ jhf/ftp/bag.ps.

4. P. Buhlmann and B. Yu. Analyzing bagging. *Annals of Statistics*, 2001, in print.
5. H. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.
6. H. Markowitz. *Portfolio Selection: Efficient Diversification of Investments.* New Haven: Yale University Press, 1959.
7. W.F. Sharpe. Adjusting for risk in portfolio performance measurement. *Journal of Portfolio Management*, Winter:29–34, 1975.
8. J. Quinlan. Bagging, boosting and c4.5. In *Proceedings of 13th Conference on AI*, pages 725–730. MIT press, 1996.
9. R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *The Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI/IAAI, 1997.
10. P. Domingos. Knowledge acquisition from examples via multiple models. In *Proc. 14th International Conference on Machine Learning*, pages 98–106. Morgan Kaufmann, 1997.
11. T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
12. B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 2002.
13. V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.
14. T. Joachims. Estimating the generalization performance of an SVM efficiently. In *Proc. 17th International Conf. on Machine Learning*, pages 431–438. Morgan Kaufmann, San Francisco, CA, 2000.
15. R. Meir. Bias, variance and the combination of least-squares estimators. In *Advnces in Neural Information Processing Systems 7*, pages 295–302. Morgan Kaufmann, San Francisco, CA, 1994.
16. A. Krogh and P. Sollich. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811–825, 1997.
17. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL: http://www.ics.uci.edu/∼mlearn/MLRepository.html.
18. T.G. Dietterich. *Ensemble Methods in Machine Learning*, pages 1–15. MIT Press, 2nd edition, 2001.
19. S. Hashem, B. Schmeiser, and Y. Yih. Optimal linear combinations of neural networks: An overview. In *1994 IEEE International Conference on Neural Networks*, 1994.
20. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
21. P. Domingos. Why does bagging work? A bayesian account and its implications. In D. Pregibon in D. Heckerman, H. Mannila and R. Uthurusamy eds, editors, *Proceedings of the third international conference on Knowledge Discovery and Data Mining*, pages 155–158. AAAI Press, 1997.
22. D. Grossman and T. Williams. Machine learning ensembles: An empirical study and novel approach. Unpublished manuscript, 2000. URL: http://www.cs.washington.edu/homes/ grossman/projects/573projects/learning.
23. G. Mani. Lowering variance of decisions by using artificial neural network portfolios. *Neural Computation*, 3(4):483–486, 1991.
24. J. Rao and R. Tibshirani. The out-of-bootstrap method for model averaging and selection. Technical report, Statistics Department, Stanford University, 1997. URL = http://www-stat.stanford.edu/ tibs/ftp/outofbootstrap.ps.