On Feature Selection, Bias-Variance, and Bagging

M. Arthur Munson[1] and Rich Caruana[2]

[1] Cornell University, Ithaca NY 14850, USA, mmunson@cs.cornell.edu
[2] Microsoft Corporation, rcaruana@microsoft.com

Abstract. We examine the mechanism by which feature selection improves the accuracy of supervised learning. An empirical bias/variance analysis as feature selection progresses indicates that the most accurate feature set corresponds to the best bias-variance trade-off point for the learning algorithm. Often, this is *not* the point separating relevant from irrelevant features, but where increasing variance outweighs the gains from adding more (weakly) relevant features. In other words, feature selection can be viewed as a variance reduction method that trades off the benefits of decreased variance (from the reduction in dimensionality) with the harm of increased bias (from eliminating some of the relevant features). If a variance reduction method like bagging is used, more (weakly) relevant features can be exploited and the most accurate feature set is usually larger. In many cases, the best performance is obtained by using all available features.

1 Introduction

In a collaboration with ecologists, we were faced with the following challenge: learn accurate models for the presence and absence of bird species from noisy observational data collected by volunteers watching bird feeders. Trying many different supervised learning algorithms (SVMs, boosted trees, neural nets, ...), we found that bagged decision trees yielded the best performance for the task. The resulting models were large, complicated, and used almost all of the 200 features available to the learning algorithm. Since the ultimate goal was to gain ecological understanding about avian population dynamics, we ran forward stepwise feature selection to find the smallest feature set yielding excellent performance.

To our surprise, after 30 steps of feature selection performance was still inferior to the performance when using all features and the gap was closing slowly as more features were added (see Figure 1). Unlike most learning algorithms, bagging appeared to perform remarkably well with many noisy features.

In this paper we examine how feature selection improves the accuracy of supervised learning through the lens of bias/variance analysis. We run feature selection for nineteen data sets and compare the bias-variance decompositions of single and bagged decision trees for many different feature subset sizes. The results show that the most accurate feature sets correspond to the best bias-variance trade-off point, and this depends on the learning algorithm. Particularly with high variance algorithms such as decision trees, this is usually *not* the

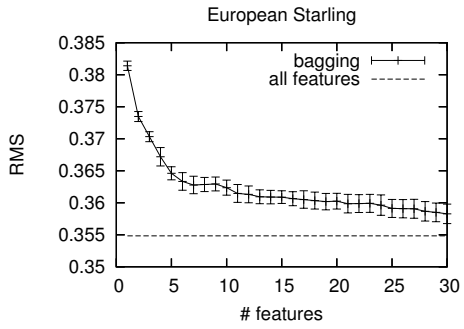

Fig. 1. Bagging performance with forward stepwise feature selection. The *all features* line shows performance of bagging with all 200 features.

separating point between relevant and irrelevant features. With too many variables, the increase in variance outweighs the potential gains of adding (weakly) relevant features. When bagging is used, however, the increases in variance are small, which makes the reduction in bias beneficial for many more features. In many cases, the best bagging performance is obtained by using all available features.

While it is known that ensemble methods improve the base learner's ability to ignore irrelevant features [1, 2], little is known about their effects on weak/noisy features. To explore this, we generate synthetic data and randomly damage varying percentages of the feature values. The results show that bagging dramatically improves the ability of decision trees to profitably use noisy features.

2 Background

This section reviews feature selection and bagging, and situates the current paper in the context of prior work.

2.1 Feature Selection

Four reasons are traditionally given to motivate feature selection [3]: better predictive performance; computational efficiency from working with fewer inputs; cost savings from having to measure fewer features; and simpler, more intelligible models. Different types of feature selection exist to satisfy varying balances of these competing goals under a variety of data regimes. This work focuses on forward stepwise feature selection (FSFS) [4] and correlation-based feature filtering (CFF). FSFS is preferred when getting the best performance from the smallest feature set possible is important — as long as it is computationally feasible. For large data sets with hundreds or thousands of features, simple filter methods like CFF are affordable and often surprisingly competitive. In the NIPS 2003 Feature Selection Challenge, "[s]everal high ranking participants obtain[ed]

good results using only filters, even simple correlation coefficients" ([5], p. 6). The main drawback to univariate filters like CFF is that they estimate the value of a feature in isolation, ignoring a) possible interactions with other features, and b) redundant information contained in features ranked higher (already selected).

Starting from an empty selected set, FSFS measures the benefit of adding each individual feature to the selected set. The benefit is measured by training a model using only the selected features (including the feature under consideration). The most beneficial (or least harmful) feature is added to the selected set, and the process is repeated for all remaining unselected features. The search stops after a fixed number of steps, once performance has stopped improving, or after all features have been selected. If feasible, the learning algorithm used in wrapper-based feature selection usually is the same algorithm to be used with the reduced feature set.

It is important for the search process to measure performance using data withheld during training to ensure good performance estimates. Additionally, the search process itself can potentially overfit this withheld data, so a third data set should be used to get an unbiased estimate of the selected subsets' performance [6]. The FSFS experiments below use a validation set to decide which feature to add, and a test set to measure the final performance.

CFF ranks the set of features by their *individual* correlation with the class label. Our experiments with large data sets use the magnitude of Pearson's correlation coefficient as the ranking criterion. The absolute correlation of feature $x_{.j}$ with the label y is:

$$r_j = \frac{|\sum_i (x_{ij} - \overline{x}_{.j})(y_i - \overline{y})|}{\sqrt{\sum_i (x_{ij} - \overline{x}_{.j})^2 \sum_i (y_i - \overline{y})^2}}$$

where i indexes over examples and $\overline{x}_{.j}$ and $\overline{y}$ are the respective means of $x_{.j}$ and y.[3] Features above a *cutoff* point are retained, while the others are discarded. Common strategies for selecting cutoff points include statistical tests of significance and cross-validated model performance at different ranks. We are interested in the performance at varying rank-levels, so we do not need to choose a cutoff.

Some researchers have looked at bias-variance estimates in the context of feature selection, but typically only for the final feature set selected (e.g. [7]). Van der Putten and van Someran [8] use bias-variance analysis to understand the wide performance spread of contestants in the 2000 CoIL challenge. They compare the bias-variance decompositions of a single subset (the top 7 features) against the original feature set, and find that feature selection is important for their problem (the decrease in variance outweighs the increase in bias).

[3] The high dimensional data sets we use are all binary classification problems with binary and/or continuous features, so Pearson's correlation coefficient is reasonable. Spearman's rank correlation would be a reasonable alternative for non-binary problems or nominal-valued features.

2.2 Bagging

Bagging [9] is a meta-learning algorithm that repeatedly creates sub-samples of the training data and trains a model (e.g. decision tree) on each sample. To make predictions, the bagged model averages the predictions of the constituent models. Bagging frequently improves the performance of a learning algorithm, and rarely hurts it. Bauer and Kohavi [10] showed empirically that the main benefit from bagging is a reduction in the variance of the underlying models. Bagging works best when models have good performance and make uncorrelated mistakes [11].

Several pieces of work exist that address features and bagging. We mention them here to avoid confusion and clarify the differences. (These techniques are not used in the experiments below.) First, *feature bagging* generates diverse simple models by training individual models with random samples of the features instead of (or in addition to) random samples of training examples [12, 13], and is particularly useful for building ensembles with simple learners that are inherently stable [2]. In *ensemble feature selection* [14] multiple good feature sets are sought such that a) a good simple model can be built from each set, and b) the simple models are maximally diverse from each other. Finally, *feature selection using ensembles* [15] uses statistics derived from tree ensembles to rank features. More generally, ensembles have been used in feature selection to find more stable feature subsets [16, 17].

3 Methodology

3.1 Learning Algorithms

To handle the wide range of data sizes, we used two different decision tree packages. In all cases bagging used 25 trees per ensemble, and training samples were drawn with replacement.

For data sets with small to medium dimensionality (< 200 features), we used minimum message length (MML) decision trees implemented in Buntine's IND package [18]. IND's MML trees use a Bayesian splitting rule with Wallace's MML tree prior [19] and use a small amount of pre-pruning to limit node expansions unwarranted by the tree's posterior probability. Predictions are smoothed by getting a prediction from the leaf and each of its ancestors on the path to the root; these fine- to coarse-grained predictions are combined in a weighted average. See Buntine [20] for full details.

We selected MML trees because the Bayesian smoothing makes them relatively low variance, so in our experience the individual trees perform well and seem to be resilient to spurious and noisy features. Thus, they are less likely to require feature selection to achieve good performance (vs. a less sophisticated decision tree like ID3), making them a strong baseline method. At the same time, they are not aggressively pruned and are large trees, making them good candidates for bagging.[4]

[4] Experiments with other very different tree methods such as C4.5 yield similar results.

For the high-dimensionality data sets, we used FEST[5], a decision tree package optimized for sparse data. To prevent overfitting, we tuned the maximum tree depth parameter in FEST to maximize performance of a single tree, using *all* features, on the validation fold of each data set. We tried depths of 1 through 10, and then the powers of 2 from 16 through 1024. The best performing depth was used for both single and bagged tree models.

A single FEST tree makes predictions from negative to positive infinity. We calibrated the predictions by fitting a sigmoid to convert them to probabilities [21]. Validation data was used to fit the calibrating sigmoid.

3.2 Performance Metrics

Model performance was measured using zero-one loss and squared error. Note that the models described above predict a probability distribution for an example, indicating the likelihood of the example belonging to each class. When the model needs to pick a single class (i.e. for zero-one loss), the class with the largest probability is chosen. A loss of zero represents perfect prediction for these measures.

Zero-one loss is the percentage of predictions that do not predict the correct class. It equals 1 − accuracy, and is often simply called the error rate for a classification model.

Mean squared error (MSE) is the average squared difference between the true prediction and the model's prediction. Let x denote an example, and let $p(x_k)$ and $q(x_k)$ be the true and predicted probability, respectively, that x is class k. Then:

$$\text{MSE} \equiv \frac{1}{nK} \sum_x \sum_k (p(x_k) - q(x_k))^2$$

where K is the number of classes for the task.[6]

Zero-one loss frequently has high variance, so MSE was used as the performance metric during FSFS when deciding which feature to add.

3.3 Data Sets

We used 19 classification tasks in our experiments: American Goldfinch presence/absence at bird feeders (AMEGFI), Lark Bunting presence/absence in the plains east of the Rocky Mountains (BUNTING), forest cover-type (COVTYPE), Pima Indians Diabetes (PIMA), letter recognition (LETTERS), mushroom identification (MUSHROOM), land classification from satellite images (SATIMAGE, Statlog

[5] http://www.cs.cornell.edu/~nk/fest/

[6] Normalizing by K is not strictly necessary, but places MSE on the same scale regardless of the number of classes.

dataset), sonar classification (SONAR), soybean disease classification (SOYBEAN), spam detection (SPAMBASE and SPAMTREC[7]), cardiac abnormalities (SPECTF), hyper-thyroid conditions (THYROID), two medical prediction tasks (MEDIS and MG5), protein crystallography diffraction pattern analysis (CRYST[8]), hand-written digit recognition (DIGITS[9]), real vs. simulated auto racing and aviation text categorization (REAL-SIM[10]), and finding translation initiation sites (TIS[11]).

Table 1. Summary of datasets.

Data Set	# Samples	# Features	# Classes	Max Depth
amegfi	23948	195	2	
bunting	20998	175	2	
covtype †‡	30,000	54	7	
letters †	20,000	16	26	
medis	14199	63	2	
mg5	22157	100	2	
mushroom †	8124	22	2	
pima †	768	8	2	
satimage †	6535	36	6	
sonar †	208	60	2	
soybean †	683	35	19	
spambase †	4601	57	2	
spectf †	266	44	2	
thyroid †	3772	27	5	
cryst	5,498	1341	2	4
digits	70,000	779	2	16–1024
real-sim	72,309	20,958	2	256–1024
spamtrec	87,688	405,915	2	256–1024
tis	13,375	927	2	5–6

†: Available from UCI Machine Learning Repository [22].
‡: First 30,000 examples from full data set.

Table 1 summarizes the data sets; high-dimensional data sets are listed below the line along with the maximum tree depth(s) chosen during parameter tuning. Data sets were chosen to cover a range of sizes (number of examples) and dimensionalities (number of features). We used the first 30,000 points from the COVTYPE data set to make the experiments more affordable.

[7] Created from TREC 2005 Spam Public Corpora. Nikos Karampatziakis, personal communication.
[8] http://ajbcentral.com/CrySis/dataset.html, unscaled version
[9] MNIST data set converted to binary classification (class 0 = digits 5 or below; class 1 = rest). Original available from http://yann.lecun.com/exdb/mnist/
[10] http://csie.ntu.edu.tw/~cjlin/libsvmtools/datasets
[11] http://datam.i2r.a-star.edu.sg/datasets/krbd/ (Kent Ridge Biomedical Data Repository)

Each data set was divided into five folds. For FSFS, three folds were used for training, one for validation (to pick which feature to add), and one for testing final performance. For CFF, feature ranks were computed from the three training folds. The description for SATIMAGE warns against using cross-validation; for that data set we used the given train/test split instead of cross-validation and pulled 435 examples from the train set to use as a validation set (about 10% of training).

3.4 Bias-Variance Decomposition

The bias-variance decomposition (BVD) of loss is a useful tool for understanding the performance characteristics of a learning algorithm. The squared error for a single example x can be decomposed into the sum of noise, bias, and variance [23], all non-negative. The noise is the intrinsic error / uncertainty for x's correct prediction, regardless of learning algorithm. Bias measures how closely the learning algorithm's average prediction (considering all possible training sets of a fixed size) matches the optimal prediction (the Bayes rate prediction). Finally, the variance of an algorithm is how much the algorithm's prediction fluctuates over different possible training sets of a given size.

We adopt the notation and definitions from Domingos [24] to formally express these quantities. Let $L(t, y)$ denote the squared loss of the prediction y for test example x which has the true value t. Further let $E_D[]$ be the expectation over the distribution of possible data sets of a fixed size; similarly, $E_t[]$ is the expectation over the distribution of possible true values for x (in a stochastic domain), and $E_{D,t}[]$ is over the joint distribution of D and t. Then expected squared loss for x can be decomposed as:

$$E_{D,t}[L(t,y)] = N(x) + B(x) + V(x),$$

$$N(x) = E_t[L(t, y_*)]$$
$$B(x) = L(y_*, y_m)$$
$$V(x) = E_D[L(y_m, y)]$$

where y is the prediction from a model trained on data drawn from D, y_* is the *optimal prediction* that minimizes $E_t[L(t, y_*)]$, and y_m is the *main prediction* that minimizes $E_D[L(y, y_m)]$. For squared loss y_m is the mean prediction of the algorithm across possible training data sets. The expected bias and variance are computed by averaging over multiple test examples.

To estimate bias and variance on real data sets, we follow the same basic sampling procedure used by Bauer and Kohavi [10], since Bouckaert [25] shows that bootstrap sampling results in less reliable bias-variance estimates. The train and validation sets are pooled to create D. Twenty samples of size $|D|/2$ are drawn from D without replacement. Each sample is used to train a model that makes predictions on the test set. This empirical distribution of the algorithm's predictions is used to compute expected bias and variance. To improve the estimates of bias and variance, we repeat this process for each fold and average the estimates.

In practice, we cannot know y_* for real data so we follow previous authors [10, 24] in using $y_* = t$. As a result, the bias and noise cannot be separated and are combined in one term for our estimates.

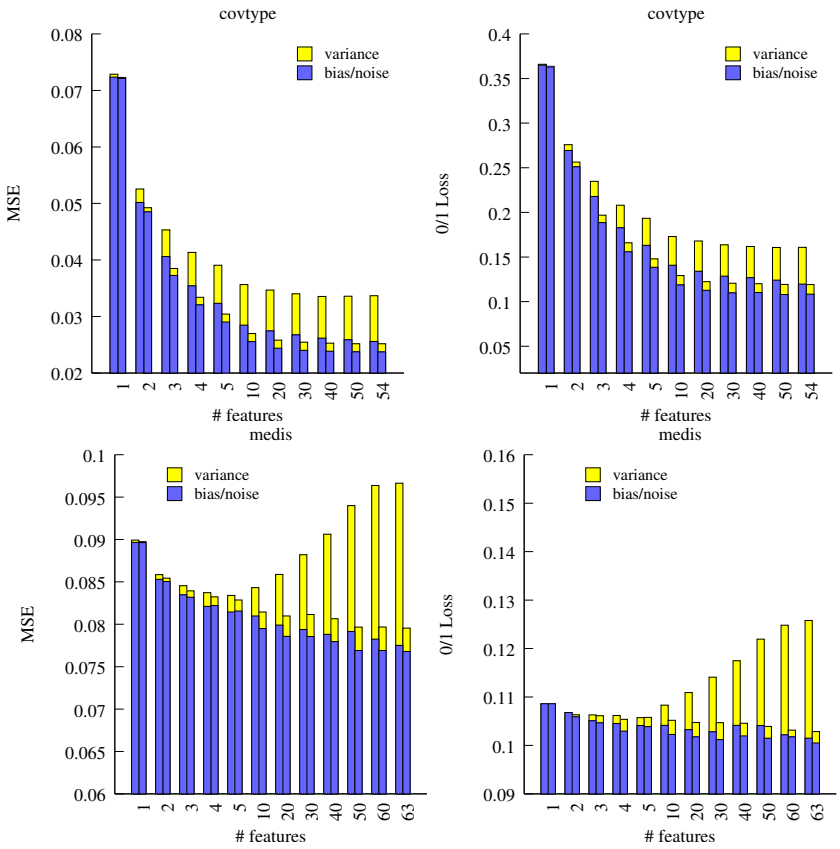

Fig. 2. Bias-variance decomposition of squared error and zero-one error for typical data sets. Left bar in pair: *single tree*; right bar: *bagging*. To better show interesting parts of graphs, the y-axes do not start at 0.

There are multiple proposals for the bias-variance decomposition of zero-one loss [26, 24]. In the results below we focus on the decomposition for squared error because feature selection hill climbing used MSE. We did, however, compute the bias and variance of zero-one loss; the results were qualitatively identical to those obtained using the squared error decomposition.

4 Bias-Variance of Feature Selection

We estimated the bias-variance decomposition for all the data sets in Table 1 at multiple feature set sizes, for both single and bagged decision trees. Feature subset orderings were found using forward stepwise feature selection (FSFS, top of table) and correlation coefficients (CFF, bottom of table). FSFS evaluated performance using single trees or bagged trees, to match the algorithm used in

the final comparison. After establishing subset orderings (and tuning the maximum tree depth for the high dimensional data sets), the training and validation sets were pooled as described in Sect. 3.4.[12] Bias-variance estimates were made at several points along the subset ordering sequence. The entire experiment was repeated across 5-folds and the 5 estimates averaged.[13]

The results cluster into two categories. Figure 2 shows representative results for two of the data sets. The *total* height of each bar is the error for the number of features on the x-axis. The pair of bars for each number of features correspond to using a single tree (left in pair) and using bagged trees (right in pair). Each bar is subdivided into portions that are due to a) the variance of the algorithm, and b) the bias of the algorithm. The bias portion also contains any noise inherent in the domain. For comparison's sake, results are shown for both mean squared error (left column) and zero-one loss (right column). For the moment we focus on patterns in the total error. Detailed observations about bias and variance are below.

Feature selection does not improve the performance of single or bagged trees on data sets in category one. Consider the graphs for COVTYPE (top row of Figure 2). Both bagging and the single tree perform as well (or better) using all features (right side of graph) than when using a subset (interior of graph). The graphs in Figure 3 show qualitatively similar results: feature selection does not improve the accuracy of single or bagged trees. (The results for zero-one loss are qualitatively the same as for squared error, and are omitted for most of the data sets.)

The second category, however, contains data sets on which feature selection improves single tree accuracy but does not improve bagging's accuracy. Looking at MEDIS (bottom row of Figure 2), the single tree achieves the minimum loss between five and ten features. Bagging, on the other hand, first reaches its minimum loss around 50 features, at which point the loss flattens out and stays roughly constant. The graphs in Figure 4 (*except* BUNTING — see discussion below) contain similar results. While single trees eventually lose performance as more features are added, bagging maintains or improves its performance with more features.

It is worth noting that for data sets in both categories (CRYST, LETTERS, MEDIS, PIMA, SATIMAGE, SONAR, SPAMBASE, SPECTF, TIS), bagging performance continues to improve as more features are added after the performance of single trees has plateaued (category 1) or peaked (category 2). In other words, bagging performance flattens further to the right in the graphs. Bagging seems to be capable of extracting information from noisy features as well as ignoring irrelevant ones. Section 5 explores this issue further.

[12] When validation data was needed to calibrate predictions, we set aside 10% of the training sample drawn from the pooled data. Thus, the calibration data varied with each training sample.

[13] The PIMA, SONAR, SPECTF, and THYROID data sets exhibited substantial variance in the results, so we repeated the 5-fold cross-validation five times using different seeds to divide the data into folds.

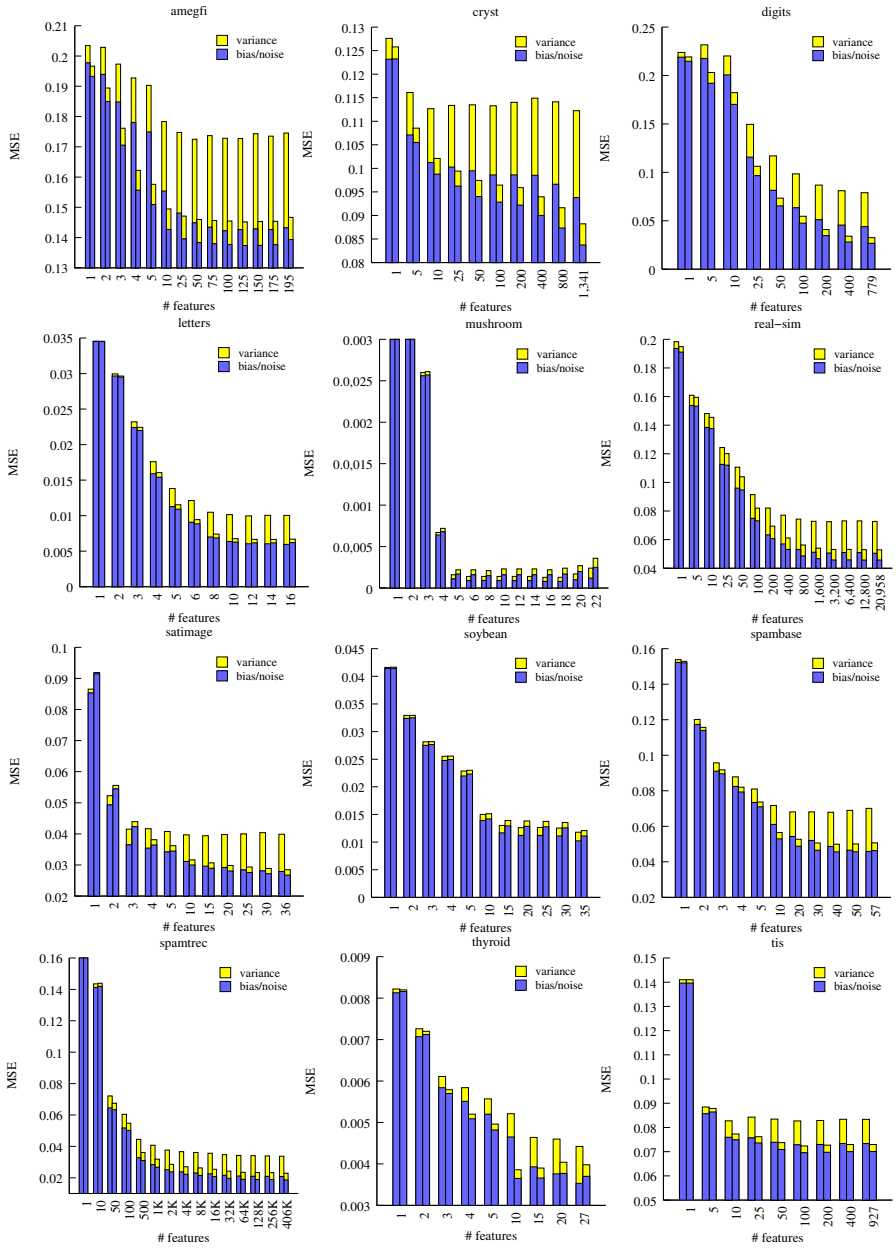

Fig. 3. Bias-variance decomposition of squared error for feature selection on data sets where feature selection does not improve performance (category 1). Left bar in pair: *single tree*; right bar: *bagging*.

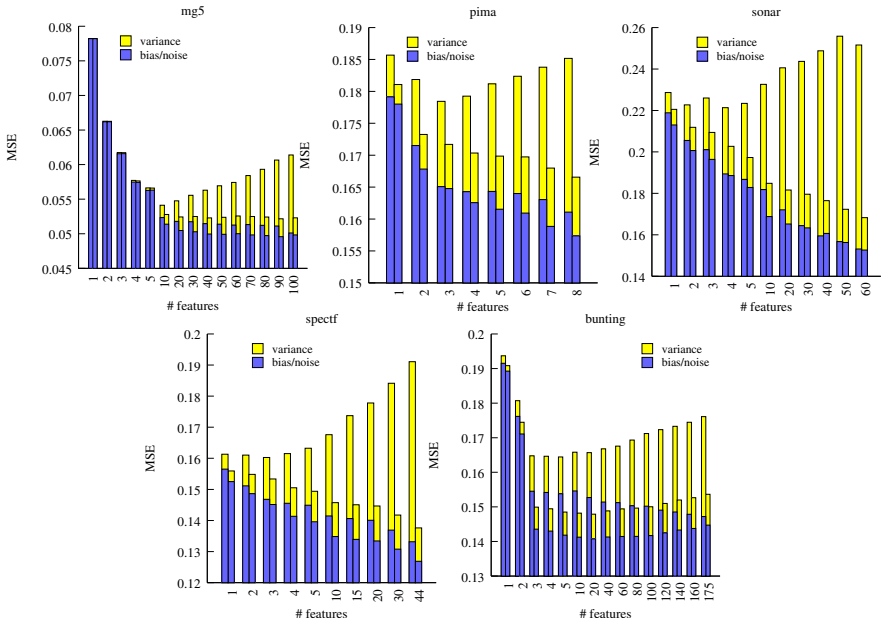

Fig. 4. Bias-variance decomposition of squared error for feature selection on data sets where feature selection helps single trees (category 2). Left bar in pair: *single tree*; right bar: *bagging*.

For all the data sets, bias decreases as more features are added. This makes intuitive sense since extra features can be thought of as extra degrees of freedom. The decrease is largest for the first few features; after that, the bias levels off as the algorithms become sufficiently flexible. Although the bias error is very similar for single trees and bagged trees, bagging does sometimes reduce bias slightly. This corroborates findings in other studies [10].

Counter to bias, variance increases with the number of features. However, this effect is much stronger for single trees than for bagged trees. Whereas the variance for bagging quickly asymptotes to a small amount, the variance for single trees grows quickly and may not asymptote. This is bagging's primary advantage.

In data sets where the performance of single trees levels off (e.g. COVTYPE), the algorithm's bias and variance asymptote so that adding more features does not hurt. Usually bagging's variance stabilizes earlier and to a lower amount, which allows it to reach lower error and benefit from additional bias decreases as more features are added.

In data sets where the single tree performance gets worse with too many features (e.g. MEDIS), the variance increases outstrip the initial benefits of reduced bias. This rarely happens to bagging because its variance typically asymptotes to a small amount of error.

Figures 3 and 4 contain three anomalies, one large and two small. The most important anomaly is the graph for BUNTING, which does not fit into either category described above. On this data set, both single trees and bagging hit peak performance between 5 and 20 features, after which their performance degrades. Thinking that perhaps this domain was just extremely noisy and that more averaging would eliminate bagging's overfitting, we re-ran feature selection on one fold using 100 trees instead of 25. With 100 trees, bagging's performance was slightly better at all points along the x-axis (compared to bagging with 25 trees), but still overfit past 20 features and to the same degree. Further investigation revealed that this data set contains several features that can be combined to create semi-unique identifiers for individual examples in the training set. All the trees in the ensemble effectively memorize these identifiers and then do poorly on the validation and test data. This can be seen in the bias-variance decomposition. Although the variance has asymptoted, the bias for bagging stops decreasing and begins increasing. With the extra features, the trees in the ensemble more consistently construct unique identifiers for training examples and lose diversity in their incorrect predictions.[14]

The other two anomalies are that single decision trees perform (slightly) better than the bagged tree ensemble for the MUSHROOM and SOYBEAN data sets. For MUSHROOM, the single tree is extremely confident in the class probabilities it assigns, and always picks the right class (zero-one loss is 0%). Bagging also always picks the right class, but the randomization from sub-sampling the training data plus averaging results in *slightly* less confident class probabilities (probability mass is pushed away from the extremes). This small bias away from extreme values has a small effect on squared error. SOYBEAN has a different problem. This small data set has 19 classes. Cross-validation and bagging sampling reduces the number of cases for some classes in the training samples so that probability estimates become less reliable and MML pre-pruning prevents leaf expansions, yielding trees that are too small.

Throughout this section (and most of the paper), noise and bias have been conflated since we do not have a way to separate them on real data. We hypothesize that the large decreases in bias—coinciding with adding the first few features—is partly due to decreases in noise. Intuitively, the Bayes optimal error rate, given only a single feature as an information source, may be quite bad (effectively high noise). As more information becomes available (more features), the Bayes rate should improve as uncertainty decreases.

[14] A more detailed explanation follows. The task in BUNTING is to predict the presence or absence of a Lark Bunting. Data are collected at multiple sites; in particular, repeat observations are made at sites over time. Identifying the site is incredibly useful for predicting presence or absence, but is not ecologically interesting. Thus, the five data folds were partitioned by site (i.e. all examples from a site appear in a single fold). Most features are tied to location (e.g. habitat), so the decision trees can easily learn to map inputs to sites in the training set using only a few features. Trees that do this make bad predictions on the validation and test folds. If the folds are created by assigning examples to folds instead of sites (spreading sites across train/valid/test), bagging does not overfit while a single tree does.

To summarize this section, these graphs show that bagging is resilient to noisy features. *Feature selection usually is unnecessary to get good performance from bagged models.* Further, picking the best subset size (using cross-validation, for example) *is not* equivalent to choosing the informative features and discarding irrelevant features. Rather, a discarded feature may be weakly informative (or correlated with a feature already selected) but cause too much variance when selected for the extra information to improve accuracy. The fact that discarded features are sometimes informative was previously noted and exploited to improve model accuracy by using discarded features as extra model outputs during training [27].

5 Noisy Informative Features

In the experiments above, bagging's performance continues to improve after the single tree's performance peaks or plateaus. This suggests that ensemble methods are not only resilient to irrelevant features [1], but also better able to take advantage of features containing useful but noisy information.

We generated synthetic data to study whether bagging improves the base learner's ability to use weak features. A binary classification problem was derived from the equation:

$$v = X_1 + X_2 X_3 + X_4^2 + \text{sign}(X_5 + X_6)$$

The class label is 1 when $v \geq 0$, and 0 otherwise. Each X_i is a univariate Gaussian variable with 0 mean and unit variance. The sign(z) function returns 1 if $z > 0$ and -1 otherwise. This function was chosen to be challenging for decision tree learning algorithms.

We generated 5,000 examples using the above function, randomly corrupted some of the inputs to generate weak features, split the data set into 5 folds, and ran a bias-variance analysis using the procedure outlined in Sect. 3.4. A feature was corrupted by permuting a fraction of its values, chosen randomly among the examples. For example, at the 0.1 corruption level, 10% of the values in corrupted features are shuffled. This was repeated 20 times, creating 20 noisy versions of each corrupted feature. Half of the X_i features were corrupted, independently of each other, while the other X_i were left intact. Single decision trees and bagged decision trees were trained using the intact features and the noisy duplicates, but not the original versions of the corrupted features. To avoid experimental bias, this process was repeated for all $\binom{6}{3}$ combinations of choosing 3 features to corrupt, and the results averaged.

Figure 5 shows the results for different corruption levels. The far left column, *core*, is the error obtained when training using only the unblemished 6 original features, and shows the best performance obtainable on this data set for these algorithms (i.e. when the ideal feature set is used). The 0.0 column shows the performance obtained using only the 3 intact features, without any corrupted features. Performances that beat this baseline indicate an algorithm is learning

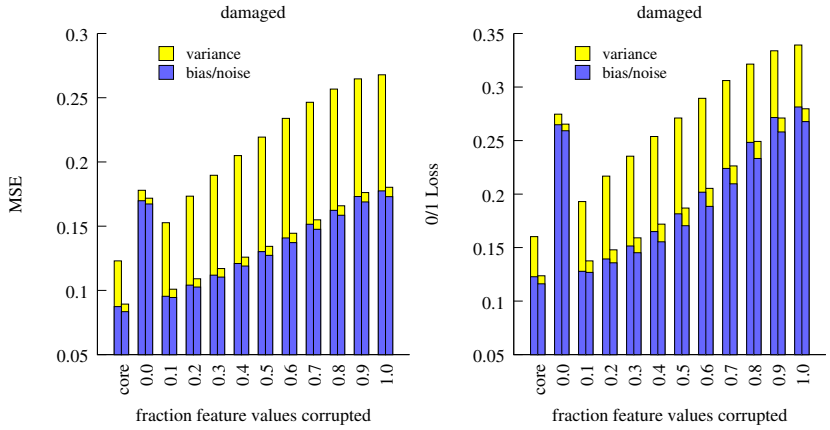

Fig. 5. Bias-variance decompositions for DAMAGED data sets with corrupted feature values. Left bar in pair: *single tree*; right bar: *bagging*. Note that the y-axes do not start at 0.

something useful from noisy features. Finally, the far right column (1.0) shows the performance when the corrupted features are pure noise (irrelevant features).

We make the following observations. First, at low corruption levels both single and bagged trees learn something useful from the noisy features. For bagged trees, performance is close to that of using the ideal feature set. Second, noisy features increase the bias (because noise is lumped in with bias in our empirical decomposition) of both single and bagged trees (vs. core), and increase the variance of single trees. Third, the main effect of increasing the corruption level is to increase the bias/noise component. Finally, the extra variance in the single trees means that the benefits of noisy features are quickly lost as the corruption level increases. At least for this synthetic task, the problem is more pronounced for squared error. In contrast, the bagged trees are remarkably resilient to damaging the feature values, and are able to extract useful information when as much as 80% of the values are corrupted.

6 Conclusions

Our experiments show that feature selection finds the feature set that represents the best trade-off between the bias of having too few features and the variance of having too many features. Because of this, most feature selection algorithms are not reliable methods for determining which features are relevant and irrelevant to a given problem: the threshold for feature inclusion/exclusion depends on the learning algorithm. Ultimately this limits the utility of feature selection for discovering which factors are important and unimportant in problems such as the avian analysis that originally motivated this work.

A by-product of our analysis is the discovery that when feature selection is too expensive to be feasible or effective, bagging provides a viable alternative to

protect from the overfitting that can occur when models are trained with too many features.[15] The bagged models always benefit from using at least as many features as the individual unbagged models. In fact, when models will be bagged, any amount of feature selection often is detrimental, and it is better to train the base models using all available features. One interpretation of our results is that feature selection is best viewed as a model regularization method instead of as a means of distinguishing relevant from irrelevant inputs.

Acknowledgments We thank the anonymous reviewers for helpful comments on paper drafts. This work was supported by NSF Award 0612031.

References

1. Ali, K.M., Pazzani, M.J.: Error reduction through learning multiple descriptions. Machine Learning **24**(3) (1996) 173–202
2. Bay, S.D.: Combining nearest neighbor classifiers through multiple feature subsets. In: ICML '98: Proceedings of the 15th International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1998) 37–45
3. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
4. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97**(1-2) (1997) 273–324
5. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the NIPS 2003 feature selection challenge. In: Advances in Neural Information Processing Systems 17. MIT Press, Cambridge, MA (2005) 545–552
6. Reunanen, J.: Overfitting in making comparisons between variable selection methods. Journal of Machine Learning Research **3** (2003) 1371–1382
7. Loughrey, J., Cunningham, P.: Using early-stopping to avoid overfitting in wrapper-based feature selection employing stochastic search. Technical Report TCD-CS-2005-37, Trinity College Dublin, Department of Computer Science (May 2005)
8. van der Putten, P., van Someren, M.: A bias-variance analysis of a real world learning problem: The CoIL challenge 2000. Machine Learning **57**(1-2) (2004) 177–195
9. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
10. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning **36**(1-2) (1999) 105–139
11. Dietterich, T.G.: Ensemble methods in machine learning. First International Workshop on Multiple Classifier Systems (2000) 1–15
12. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 832–844

[15] Of course, feature rankers like CFF are less expensive than training a single bagged tree ensemble. Even with a ranker, however, there is the problem of choosing a cutoff threshold for which features to include — which typically requires training models for multiple candidate threshold levels. Thus, carefully choosing a threshold could easily cause a feature ranker to be more computationally expensive than training a single ensemble.

13. Bryll, R., Gutierrez-Osuna, R., Quek, F.: Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition **36**(6) (2003) 1291–1302
14. Opitz, D.W.: Feature selection for ensembles. In: AAAI 1999: Proceedings of the 16th National Conference on Artificial Intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (1999) 379–384
15. Tuv, E., Borisov, A., Torkkola, K.: Feature selection using ensemble based ranking against artificial contrasts. In: International Joint Conference on Neural Networks. (2006) 2181–2186
16. Saeys, Y., Abeel, T., Peer, Y.: Robust feature selection using ensemble feature selection techniques. In: ECML PKDD '08: Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases, Berlin, Heidelberg, Springer-Verlag (2008) 313–325
17. Tuv, E.: Ensemble learning. In Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A., eds.: Feature Extraction: Foundations, and Applications. Volume 207 of Studies in Fuzziness and Soft Computing. Springer (2006) 187–204
18. Buntine, W., Caruana, R.: Introduction to IND and recursive partitioning. Technical Report FIA-91-28, NASA Ames Research Center (10 1991)
19. Wallace, C.S., Patrick, J.D.: Coding decision trees. Machine Learning **11**(1) (1993) 7–22
20. Buntine, W.: Learning classification trees. Statistics and Computing **2**(2) (1992) 63–73
21. Platt, J.C.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In Smola, A.J., Bartlett, P.J., Schoelköpf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press (2000) 61–74
22. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
23. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation **4**(1) (1992) 1–58
24. Domingos, P.: A unified bias-variance decomposition and its applications. In: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann (2000) 231–238
25. Bouckaert, R.R.: Practical bias variance decomposition. In: AI 2008: Advances in Artificial Intelligence. Lecture Notes in Computer Science, Springer (2008) 247–257
26. Kohavi, R., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufman (1996)
27. Caruana, R., de Sa, V.R.: Benefitting from the variables that variable selection discards. Journal of Machine Learning Research **3** (2003) 1245–1264