On Feature Selection, Bias-Variance, and Bagging

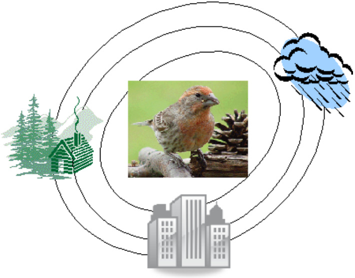
Art Munson[1] Rich Caruana[2]

[1]Department of Computer Science
Cornell University

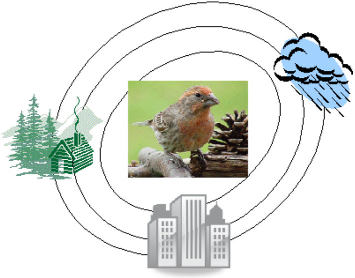
[2]Microsoft Corporation

ECML-PKDD 2009

Task: Model Presence/Absence of Birds



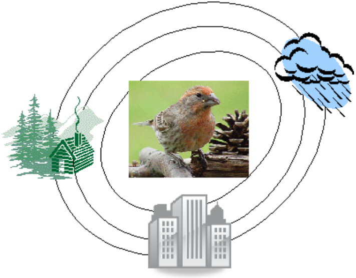
Task: Model Presence/Absence of Birds



Tried:

- SVMs
- boosted decision trees
- bagged decision trees
- neural networks
- . . .

Task: Model Presence/Absence of Birds



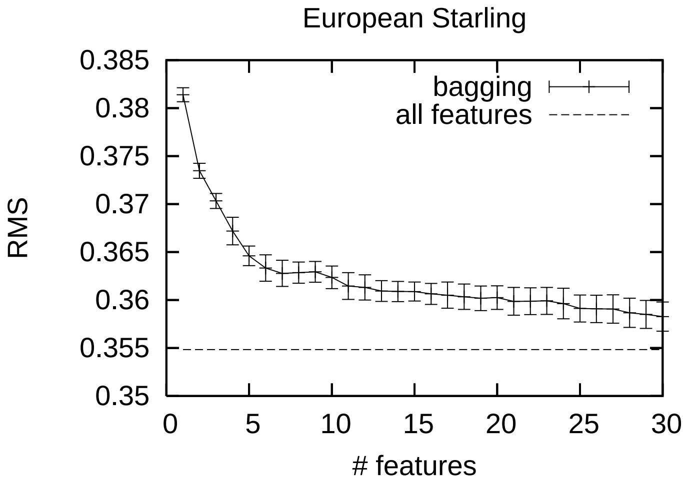
Tried:

- SVMs
- boosted decision trees
- bagged decision trees
- neural networks
- ...

Ultimate goal: understand avian population dynamics

Ran feature selection to find smallest feature set with excellent performance.

Bagging Likes Many Noisy Features (?)



Surprised Reviewers

Reviewer A

[I] also found that the results reported in Figure 2 [were] strange, where the majority [of] results show that classifiers built from selected features are actually inferior to the ones trained from the whole feature [set].

Reviewer B

It is very surprising that the performance of all methods improves (or stays constant) when the number of features is increased.

Purpose of this Study

Does bagging often benefit from many features?

If so, why?

Outline

Review of Bagging

Bagging: simple ensemble learning algorithm [Bre96]:

- draw random sample of training data
- train a model using sample (e.g. decision tree)
- repeat N times (e.g. 25 times)
- bagged predictions: average predictions of N models

Facts about Bagging

- Surprisingly competitive performance & rarely overfits [BK99].
- Main benefit is reducing variance of constituent models [BK99].
- Improves ability to ignore irrelevant features [AP96].

Review of Bias-Variance Decomposition

Error of learning algorithm on example x comes from 3 sources:

- **noise** intrinsic error / uncertainty for x's true label
- **bias** how close, on average, is algorithm to optimal prediction
- **variance** how much does prediction change if change training set

Error decomposes as:

$$\text{error}(x) = \text{noise}(x) + \text{bias}(x) + \text{variance}(x)$$

On real problems, cannot separately measure bias and noise.

Measuring Bias & Variance (Squared Error)

Generate empirical distribution of the algorithm's predictions [BK99]:

- Randomly sample $\frac{1}{2}$ of the training data.
- Train model using sample and make predictions y for test data.
- Repeat R times (e.g. 20 times).
- Compute average prediction y_m for every test example.

Measuring Bias & Variance (Squared Error)

Generate empirical distribution of the algorithm's predictions [BK99]:

- Randomly sample $\frac{1}{2}$ of the training data.
- Train model using sample and make predictions y for test data.
- Repeat R times (e.g. 20 times).
- Compute average prediction y_m for every test example.

For each test example x with true label t:

$$\text{bias}(x) = (t - y_m)^2$$

$$\text{variance}(x) = \frac{1}{R}\sum_{i=1}^{R}(y_m - y_i)^2$$

Average over test cases to get expected bias & variance for algorithm.

Review of Feature Selection

Forward Stepwise Feature Selection

- Start from empty selected set.
- Evaluate benefit of selecting each non-selected feature (train model for each choice).
- Select most beneficial feature.
- Repeat search until stopping criteria.

Review of Feature Selection

Forward Stepwise Feature Selection

- Start from empty selected set.
- Evaluate benefit of selecting each non-selected feature (train model for each choice).
- Select most beneficial feature.
- Repeat search until stopping criteria.

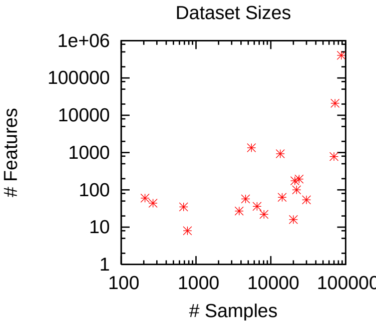
Correlation-based Feature Filtering

- Rank features by *individual* correlation with class label.
- Choose cutoff point (by statistical test or cross-validation).
- Keep features above cutoff point. Discard rest.

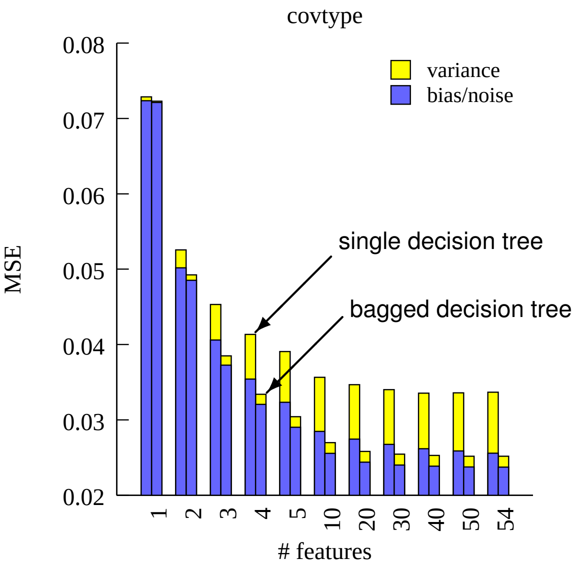
Experiment 1: Bias-Variance of Feature Selection

Summary:

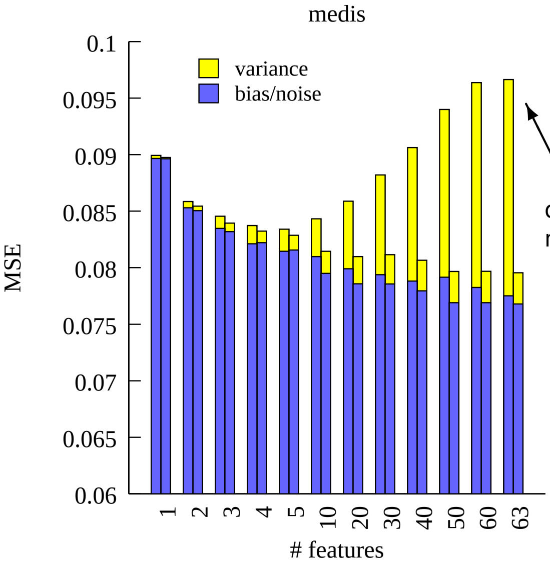
- 19 datasets
- order features using feature selection
- forward stepwise feature selection or correlation feature filtering, depending on dataset size
- estimate bias & variance at multiple feature set sizes
- 5-fold cross-validation



Case 1: No Improvement from Feature Selection



Case 2: FS Improves Non-Bagged Model

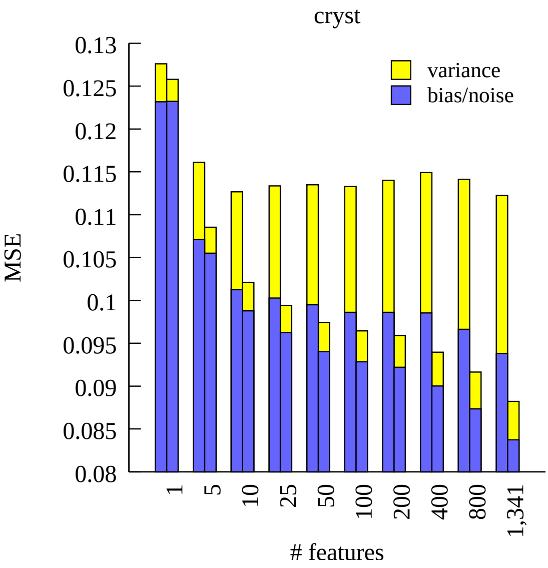


overfits with too many features

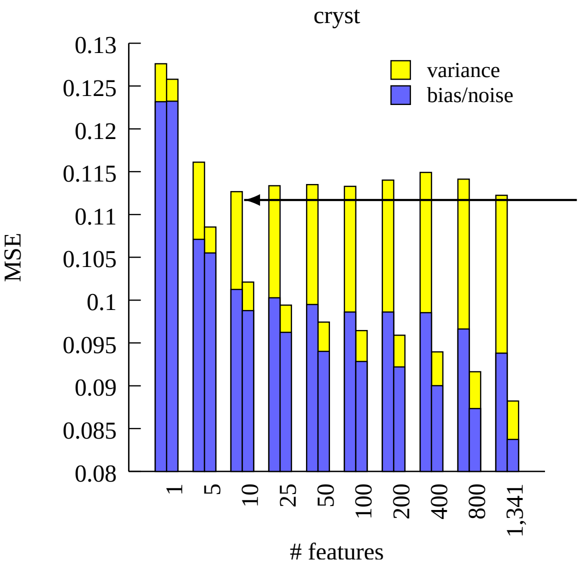
Take Away Points

- More features $\Rightarrow$ lower bias/noise, higher variance.
- Feature selection does not improve bagged model performance (1 exception).
- Best subset size corresponds to best bias/variance tradeoff point.
  - Algorithm dependant
  - Relevant features may be discarded if variance increase outweighs extra information

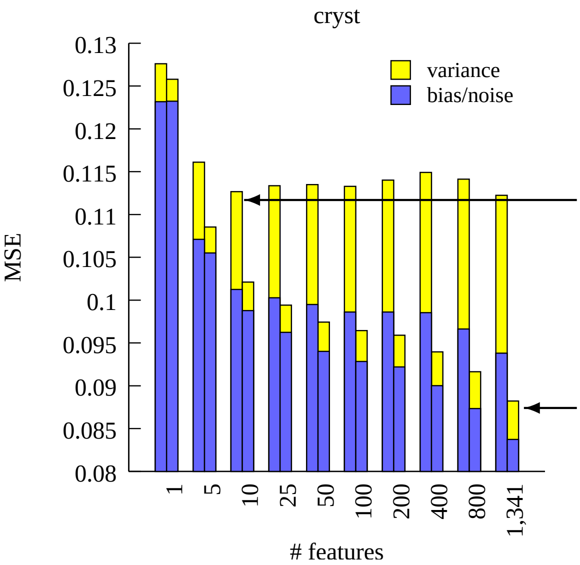
Why Does Bagging Benefit from so Many Features?



Why Does Bagging Benefit from so Many Features?



Why Does Bagging Benefit from so Many Features?



Hypothesis

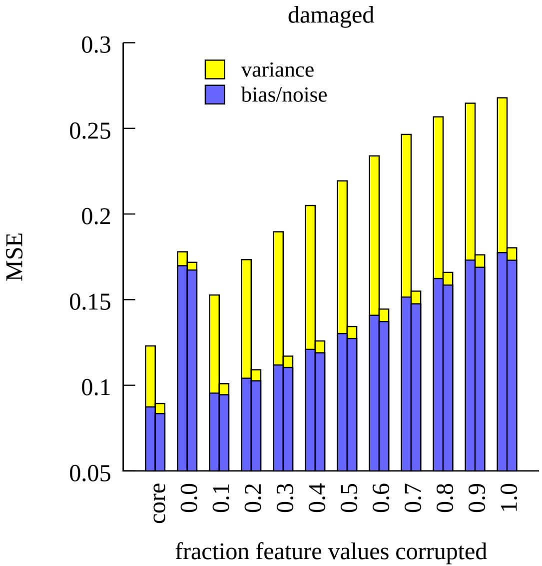
Bagging improves base learner's ability to benefit from weak, noisy features.

Experiment 2: Noisy Informative Features

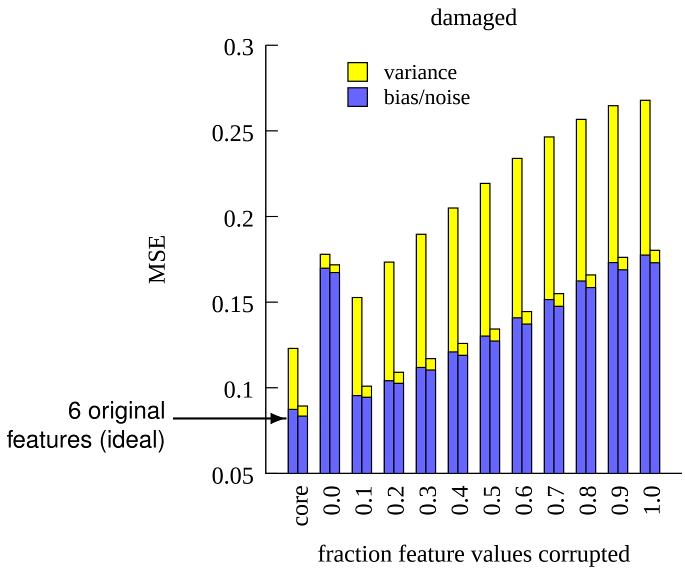
Summary:

- generate synthetic data (6 features)
- duplicate 1/2 of the features 20 times
- corrupt X% of values in duplicated features
- train single and bagged trees with corrupted features and 3 non-duplicated features
- compare to:
  - ideal, unblemished feature set, and
  - no noisy features (3 non-duplicated only)

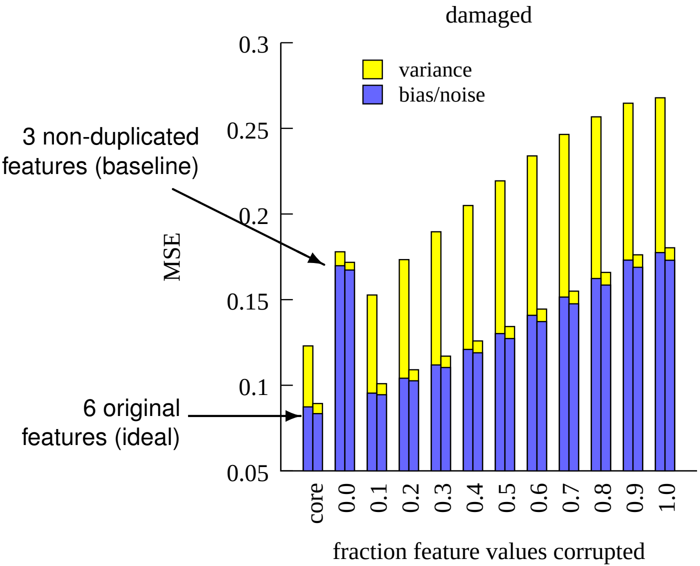
Bagging Extracts More Info from Noisy Features



Bagging Extracts More Info from Noisy Features



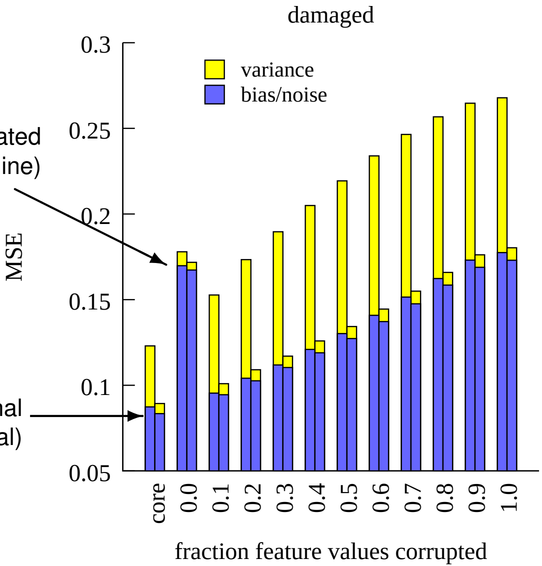
Bagging Extracts More Info from Noisy Features



Bagging Extracts More Info from Noisy Features

3 non-duplicated features (baseline)

6 original features (ideal)



everything else: 3 non-duplicated features + 60 noisy features

Conclusions

After training 9,060,936 decision trees . . .

Experiment 1:

- More features $\Rightarrow$ lower bias/noise, higher variance.
- Feature selection does not improve bagged model performance.
- Best subset size corresponds to best bias/variance tradeoff point.

Experiment 2:

- Bagged trees surprisingly good at extracting useful information from noisy features. Different weak features in different trees.

Bibliography

Kamal M. Ali and Michael J. Pazzani.
Error reduction through learning multiple descriptions.
Machine Learning, 24(3):173–202, 1996.

Eric Bauer and Ron Kohavi.
An empirical comparison of voting classification algorithms: Bagging, boosting, and variants.
Machine Learning, 36(1-2):105–139, 1999.

Leo Breiman.
Bagging predictors.
Machine Learning, 24(2):123–140, 1996.

Exception: Overfitting Pseudo-Identifiers