Computational Data Analysis

Ensemble methods

Bagging, Boosting and Random Forest

GET THE BIASES AND VARIANCES RIGTH FOR THE EXAM

Line Clemmensen and Lars Arvastson

April 4, 2018

Today's Lecture

- Recap
- Bootstrapping (recap)
- Theory behind and use of
  - Bagging
  - Boosting
  - Random forest
- Out-of-bag estimates (OOB)

Last Week

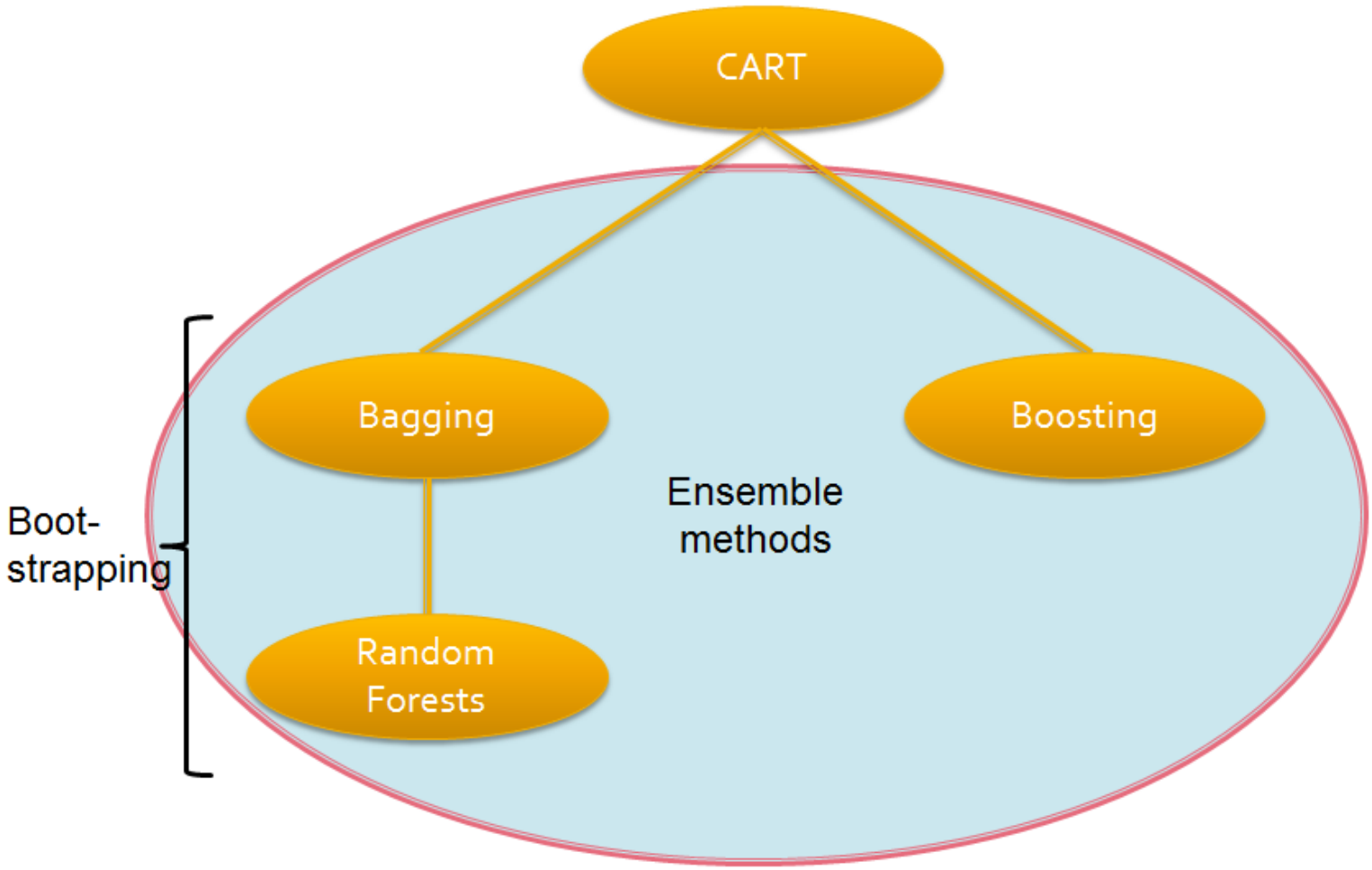
- Classification And Regression Trees - CART
  - Partition the feature space into rectangles and fit simple model (constant) in each one.

  Advantage:
  - Interpretability, tree defines a set of rules which are easy to follow.
  - Handles missing data.
  - Can take both continuous and categorical variables as input.

  Drawbacks:
  - Deep trees have high variance and low bias.
  - Small trees have low variance and high bias.
  - New data might completely change the shape of the tree.

Exercise 1c,
number of combinations is $2^{k-1} - 1$

Bagging vs Boosting vs Random forest



Bootstrapping reviewed

Start with available data,

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \\ 5 & 10 \end{bmatrix}$$

sample observations with **replacement**

$$X_1' = \begin{bmatrix} 3 & 6 \\ 3 & 6 \\ 5 & 10 \\ 2 & 4 \\ 3 & 6 \end{bmatrix} \qquad X_2' = \begin{bmatrix} 4 & 8 \\ 2 & 4 \\ 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix} \qquad X_3' = \begin{bmatrix} 1 & 2 \\ 5 & 10 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \end{bmatrix}$$

Idea: Empirical distribution mimics the distribution X is drawn from.

Bagging

- Many models built on bootstrap replicates of data
- Output from all models aggregated into one output

Bagging

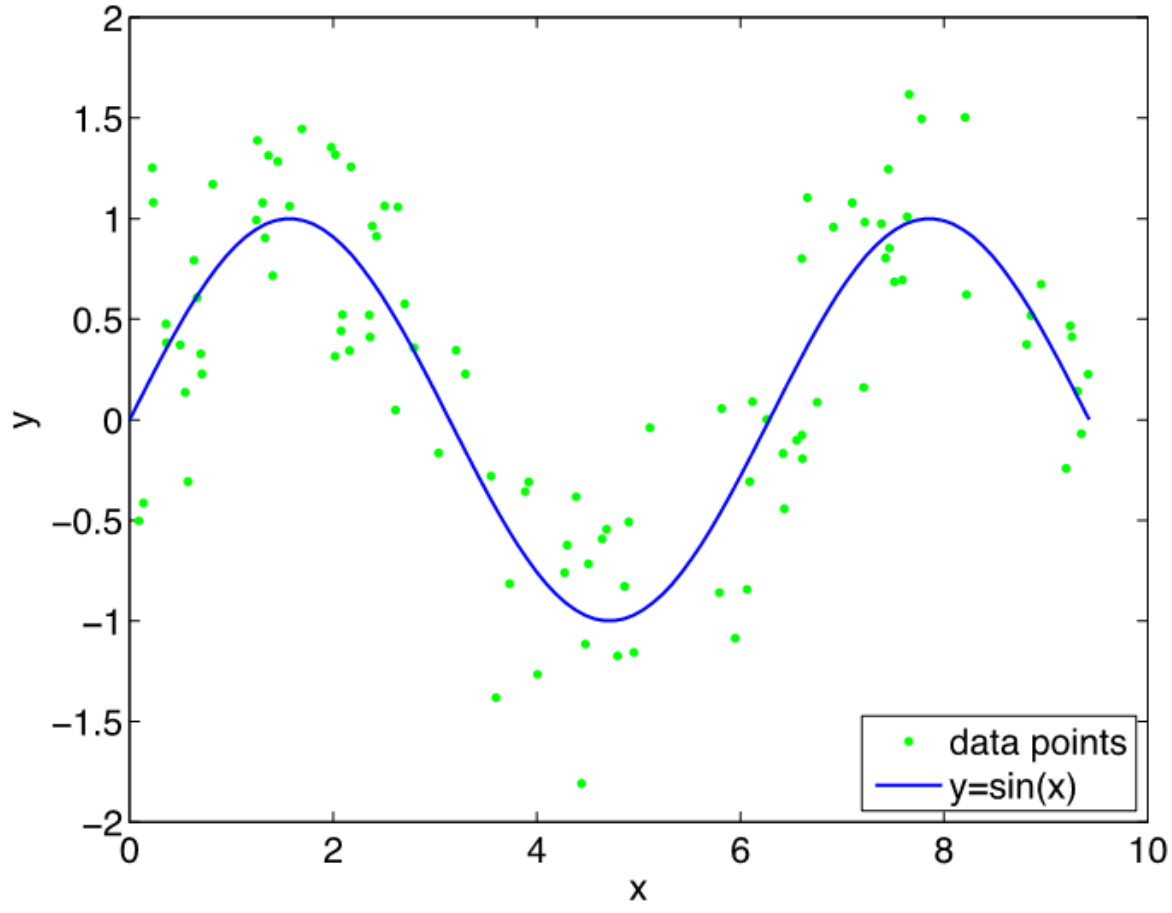
- Use the bootstrap to improve predictions (not as earlier to tune parameters or assess prediction errors).
- Bagging averages predictions over a collection of bootstrap samples.
- Average many noisy but approximately unbiased models and thereby reduce the variance

Bagging algorithm

1. Make B bootstrap samples of size N.
2. For $b = 1$ to B repeat step (a)
   a. Fit a model using the bth sample and make the prediction $\hat{y}_b$.
3. The bagging estimate is then given by the average of the ensemble of B predictors,

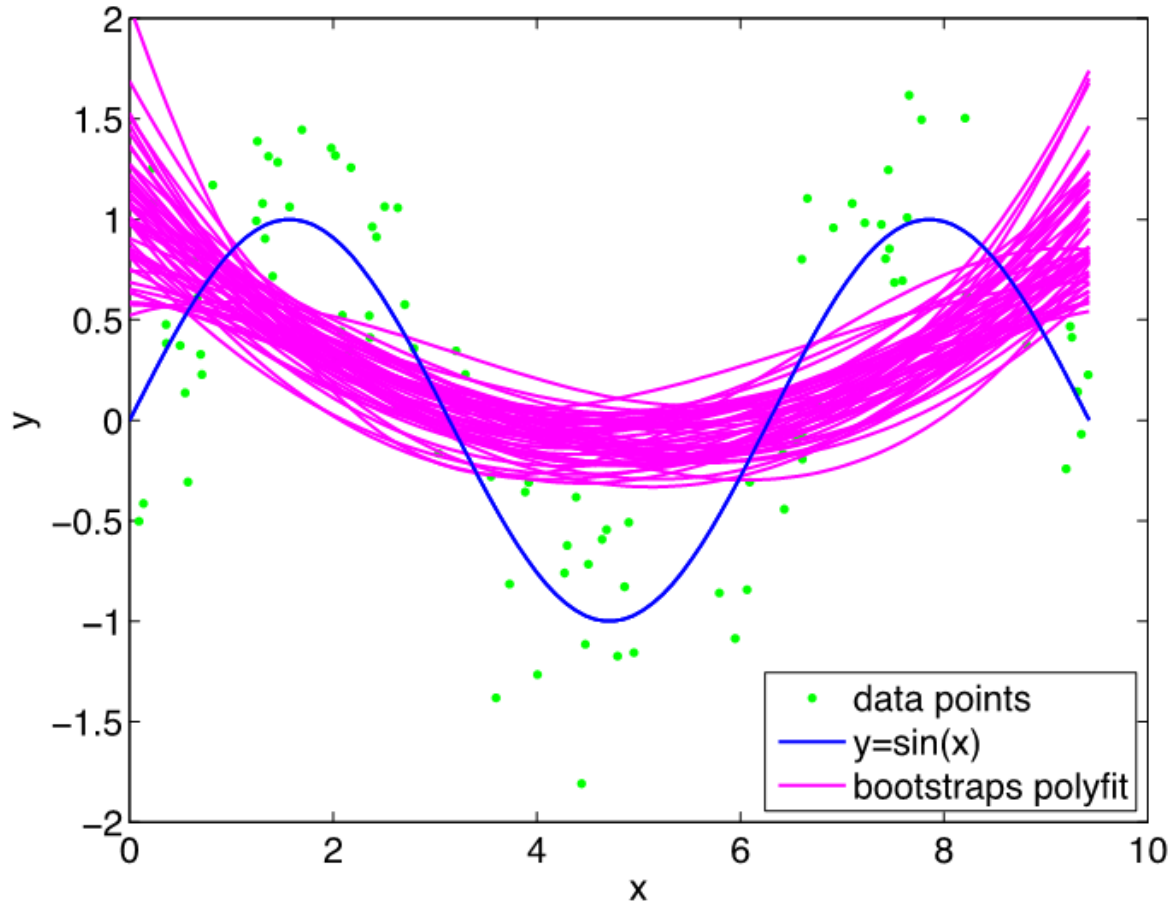
$$\hat{y}_{\text{bagging}} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b$$

Bagging example - sin(x)



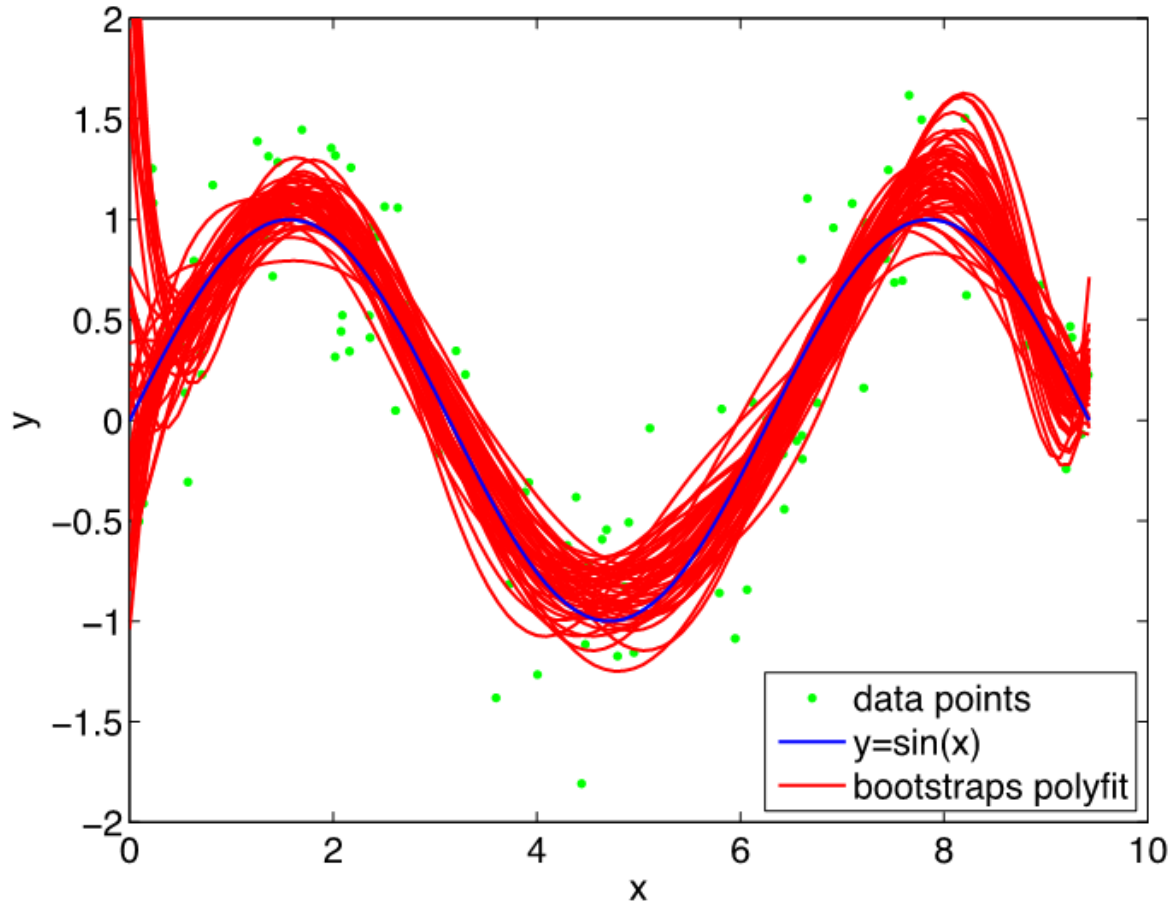
100 data points with white noise added from $y = \sin(x)$.

Bagging example - sin(x)



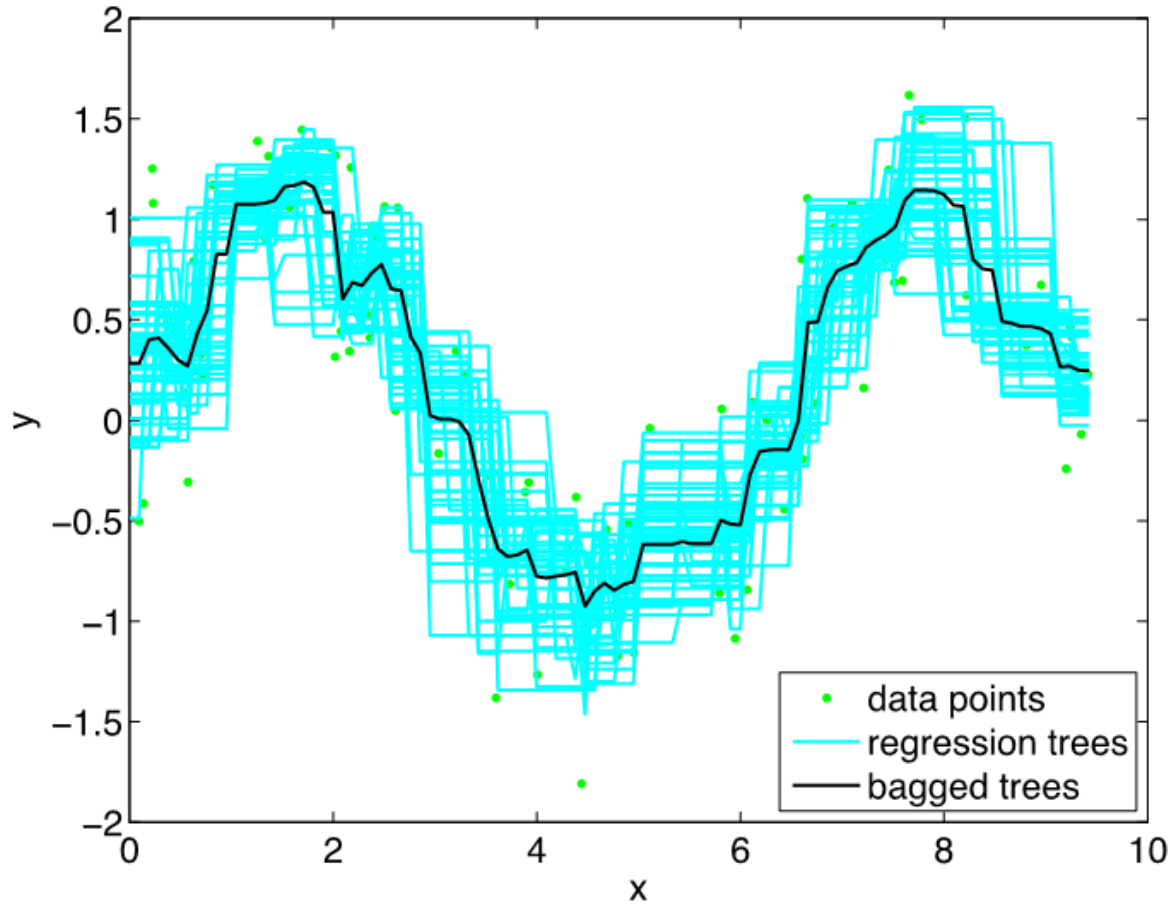
Fits of a 3rd degree polynomial to 50 bootstraps of data.

Bagging example - sin(x)



Fits of a 9rd degree polynomial to 50 bootstraps of data.

Bagging example - sin(x)



Fits of regression trees to 50 bootstraps of data and the bagged tree.

Buzz exercise

What types of model would you choose for bagging?

Bagging - bias

The bias of bagged trees is the same as that of the individual trees, as the trees are identically distributed,

$$
\begin{aligned}
E(\hat{y} - y) &= E\left[\frac{1}{B}\sum_{b=1}^{B}(\hat{y}_b - y)\right] \\
&= \frac{1}{B}\sum_{b=1}^{B} E(\hat{y}_b - y) \\
&= E(\hat{y}_b - y) \quad b = 1, ..., B
\end{aligned}
$$

Bagging - variance

The variance of the average of bagged estimates is (ρ is the correlation between pairs of trees)

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

The second term goes to zero as B increases. We see that the size of the correlation between pairs of bagged trees, ρ, is what limits the variance reduction.

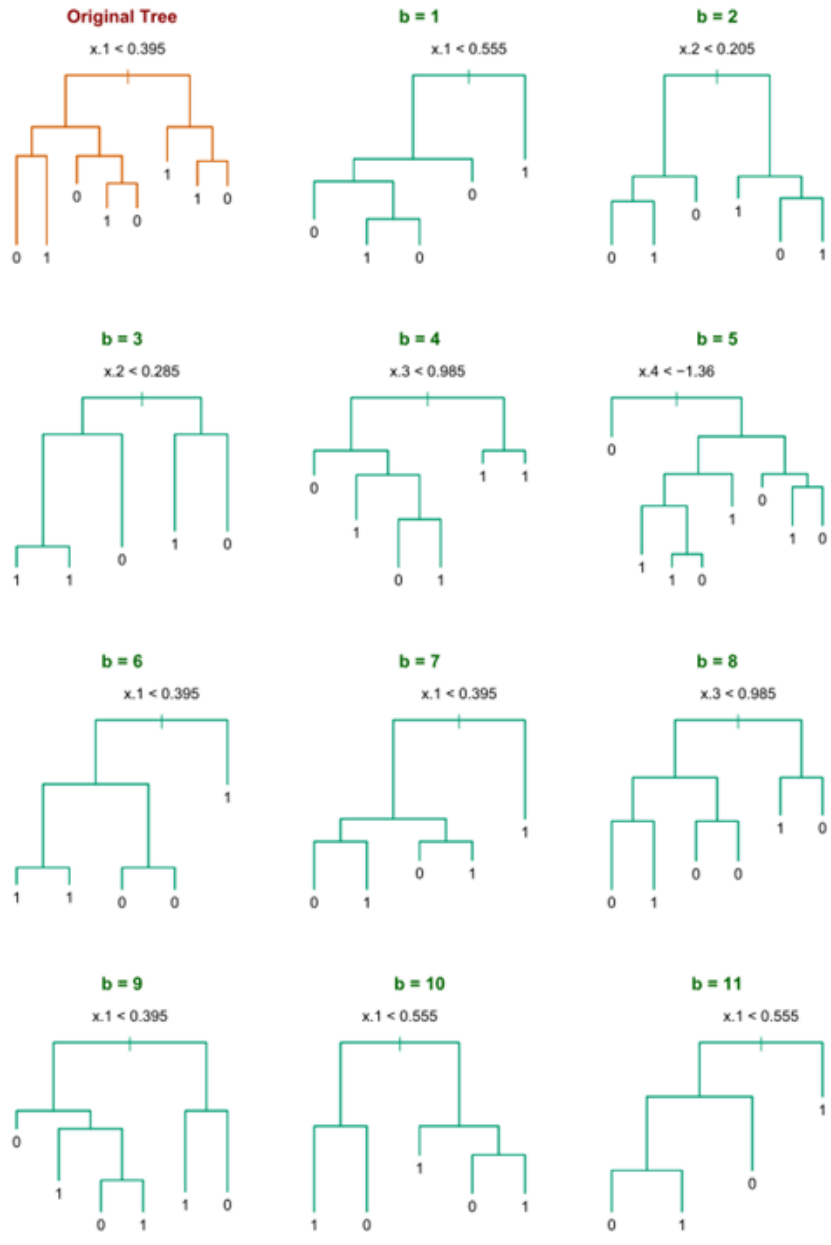
We will explore this further in the exercises!

Bagging

- Particularly good for high-variance, low-biased methods - such as trees.
- **Regression**
  - Regression trees are fitted to bootstrap samples of the training data. The result is the average over all of the trees.
- **Classification**
  - A committee of trees each cast a vote for the class - and the majority vote is used as the prediction.

Bagging trees

Only average the predictions.
Not the structure



Bagging in Matlab

- Simply use bootstrapping and then average the predictors or
- use the `TreeBagger` function for bagging and random forests of trees or
- use the `fitensemble` method for boosting and bagging with learners KNN, discriminant analysis and trees.

Boosting

Focus on the mistakes there have been made..

- Average many trees, each grown to reweighted versions of the training data.
- Weighting *decorrelates* the tree, by focusing on regions missed by past trees.

Boosting

Weak classifiers → focus on small tress.
We want model with less variance but you can introduce some bias.

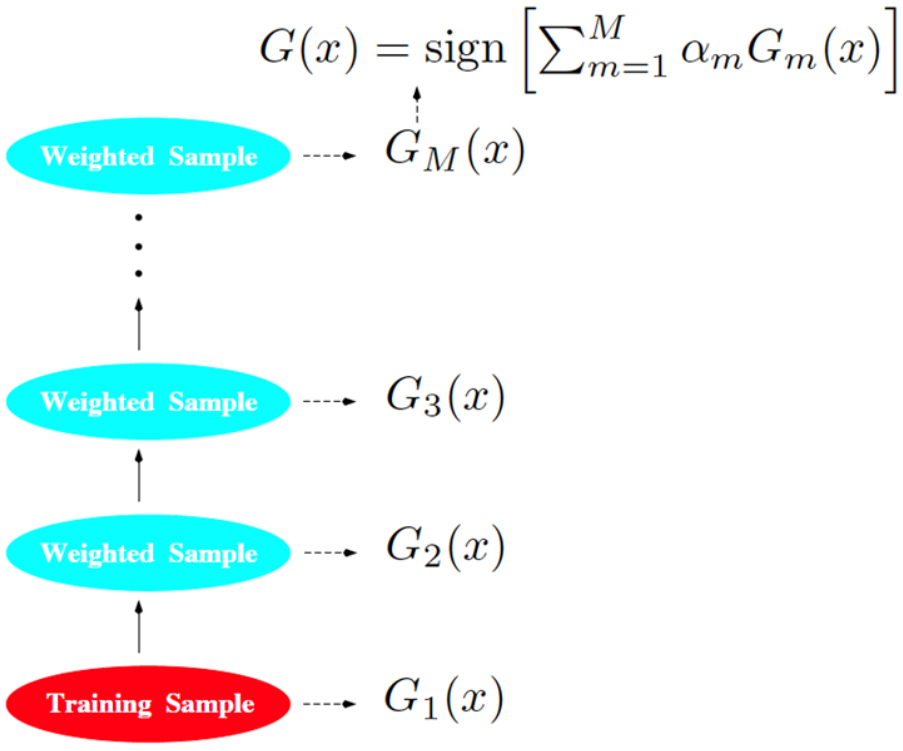
- A set of weak classifiers (small trees) are grown in an adaptive manner to remove bias (hence they are not independent).
- Weights are put on the observations
  - weights are updated to emphasize missclassifications.
- The final estimator is a weighted average of all the trees.
- **Boosting shrinks bias.**
  - Therefore, combine small trees (stubs) with high bias.
  - Many stubs together form a (very) good model with Boosting.

AdaBoost.M1

- The most popular choice of boosting algorithm
- For a two class problem (binary)
- Freund and Schapire 1997 Hard to prove why it works.

Caveat: The method is empirically motivated. Showing any optimal properties is difficult.

AdaBoost.M1



Schematic of AdaBoost. Classifiers are trained on weighted versions of the data set, and then combined to produce a final prediction.
Note: $G_m(x) \in \{-1, 1\}$

AdaBoost.M1 algorithm

1. Initialize the observation weights $w_i = 1/N, \;\; i = 1, 2, ..., N$.

2. For $m = 1$ to M repeat steps (a) - (d)

   a Fit a classifier $G_m(x)$ to the training data using the weights w_i.

   b Compute weighted error of the newest tree

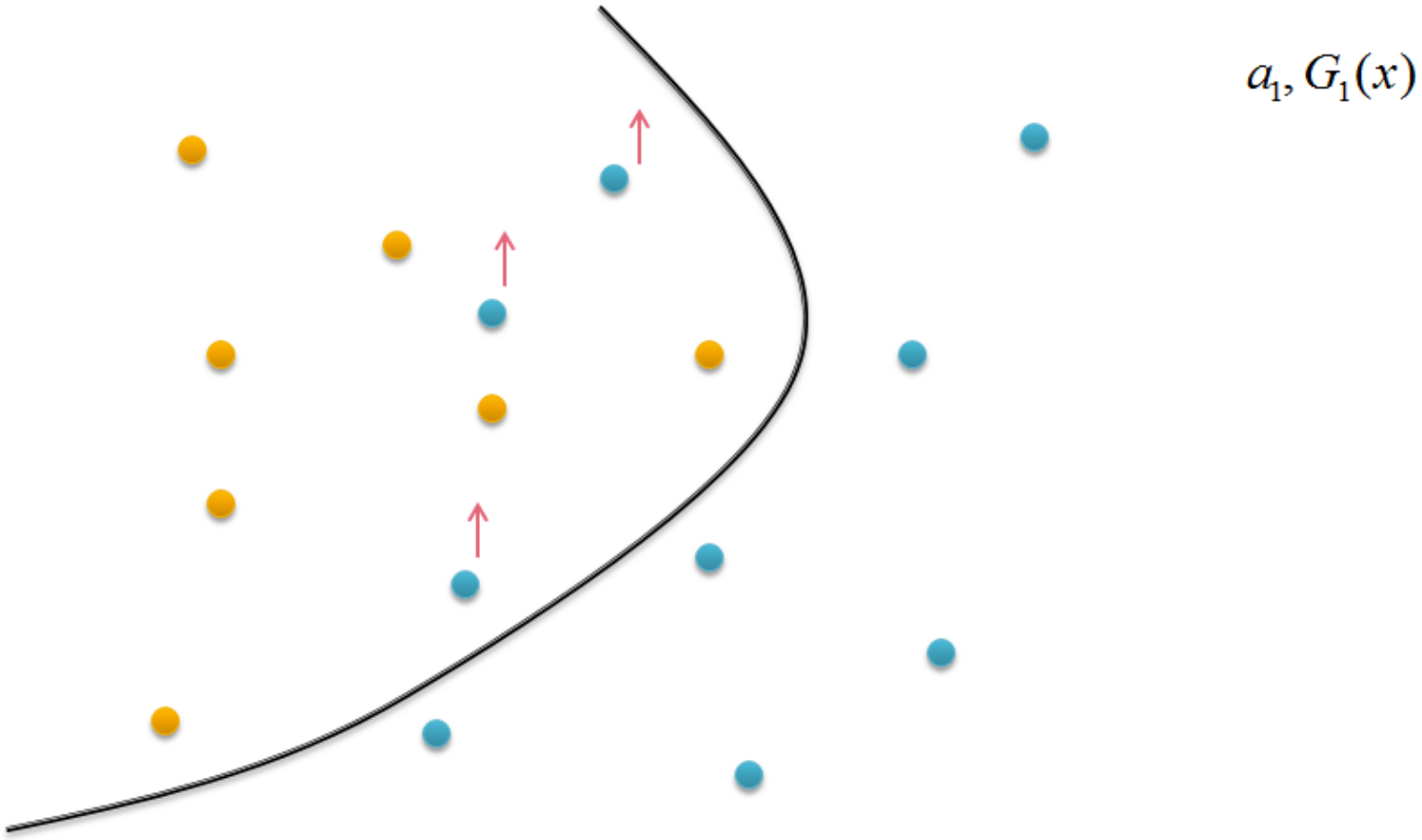
$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

   c Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$.

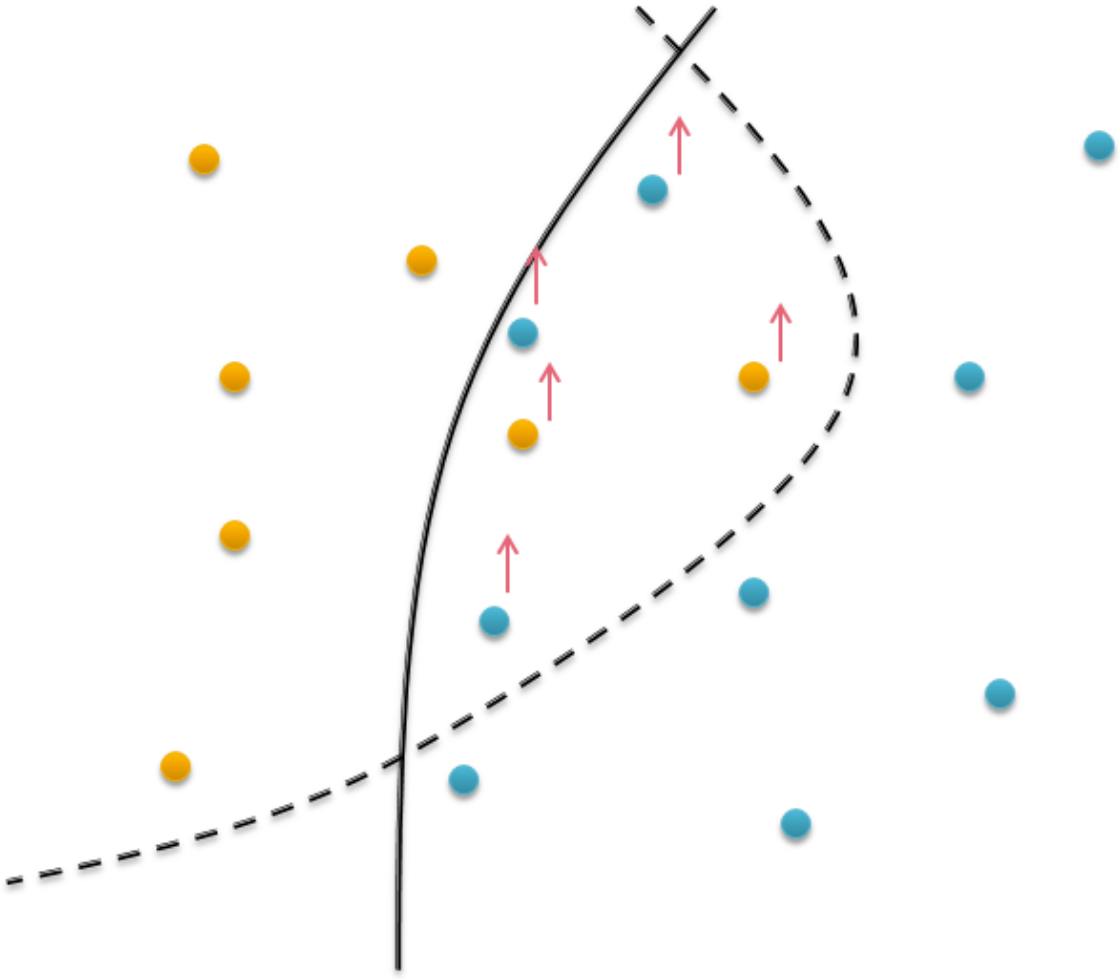
   d Update weights for $i = 1, ..., N$:
   $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(x_i))]$
   and renormalize to w_i to sum to 1.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Boosting example



Boosting example



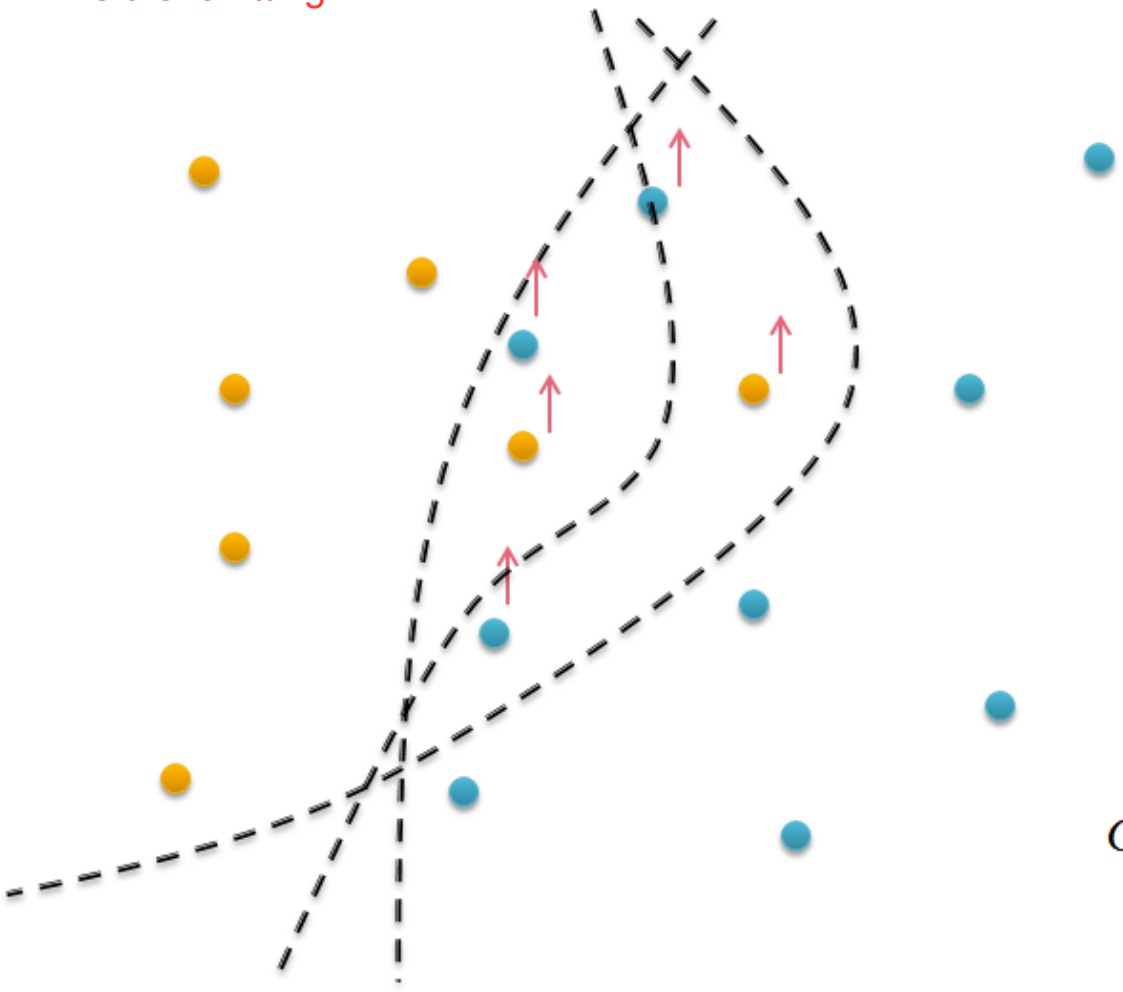
$a_1, G_1(x)$

$a_2, G_2(x)$

Boosting example

Adding weigth.. repeating the misclassified samples..

Avoid overfitting..



$a_1, G_1(x)$

$a_2, G_2(x)$

$a_3, G_3(x)$

$$G(x) = sign\left[\sum_{m=1}^{M} a_m G_m(x)\right]$$

"What you put in - is what you get out."

Boosting

always have an IID data → know when to stop.

- Was intended as a committee method like Bagging.
- Is a committee of *weak learners* where each cast a vote for the final prediction. However, these evolve over time and the members cast a weighted vote.
- Boosting dominates Bagging on most problems, and therefore became the preferred choice.

Caveat: Boosting is not known for overfitting but it can overfit!

Shrinkage in Boosting

- Shrinkage, like ridge, can be added by controlling the **learning rate**.
- Meaning the contribution of each tree is scaled by a factor $0 < \nu < 1$

Example - sin(x)

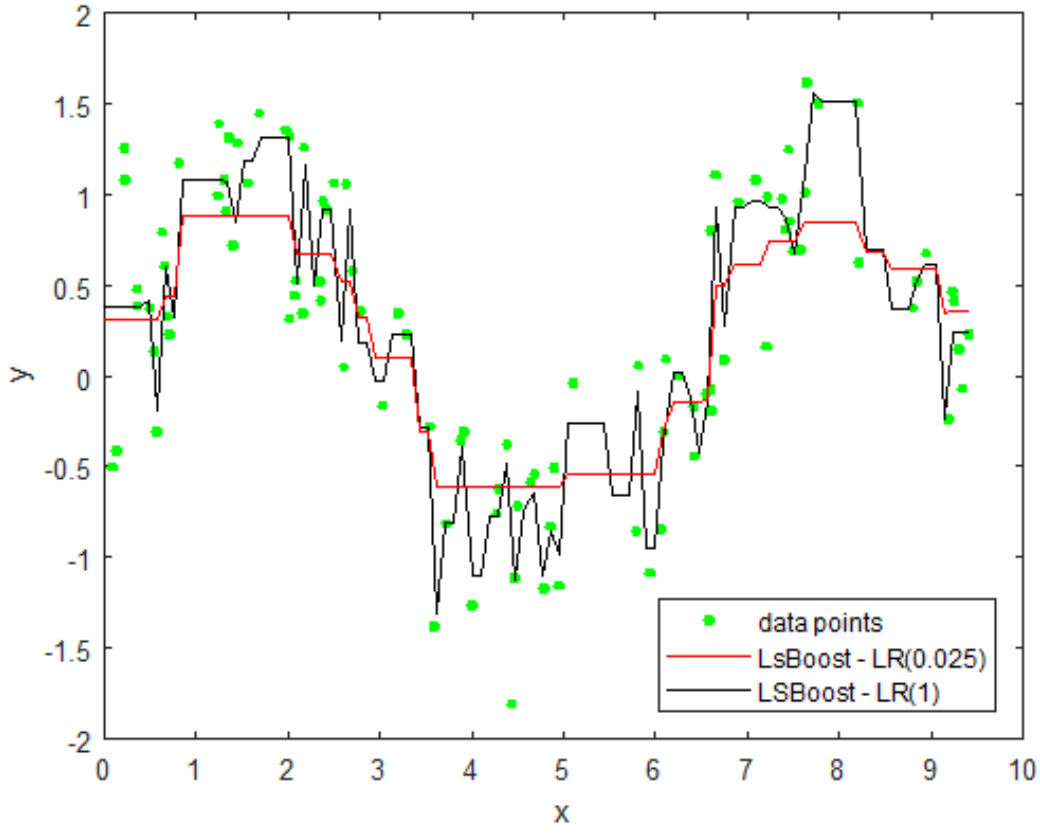
```
% Boosting - Regression

% Setup the individual trees
t1 = RegressionTree.template('MaxNumSplits',3);
t2 = RegressionTree.template('MaxNumSplits',3);

% fit boosted trees
ens = fitensemble(x,y,'LSBoost',100,t2);
ybo = predict(ens,xt');
h3  = plot(xt,ybo,'k');

% fit using shrinkage
ens = fitensemble(x,y,'LSBoost',100,t1,'LearnRate',.025);
ybo = predict(ens,xt');
h2  = plot(xt,ybo,'r');
```

Example - sin(x)



Example - classification of zip data

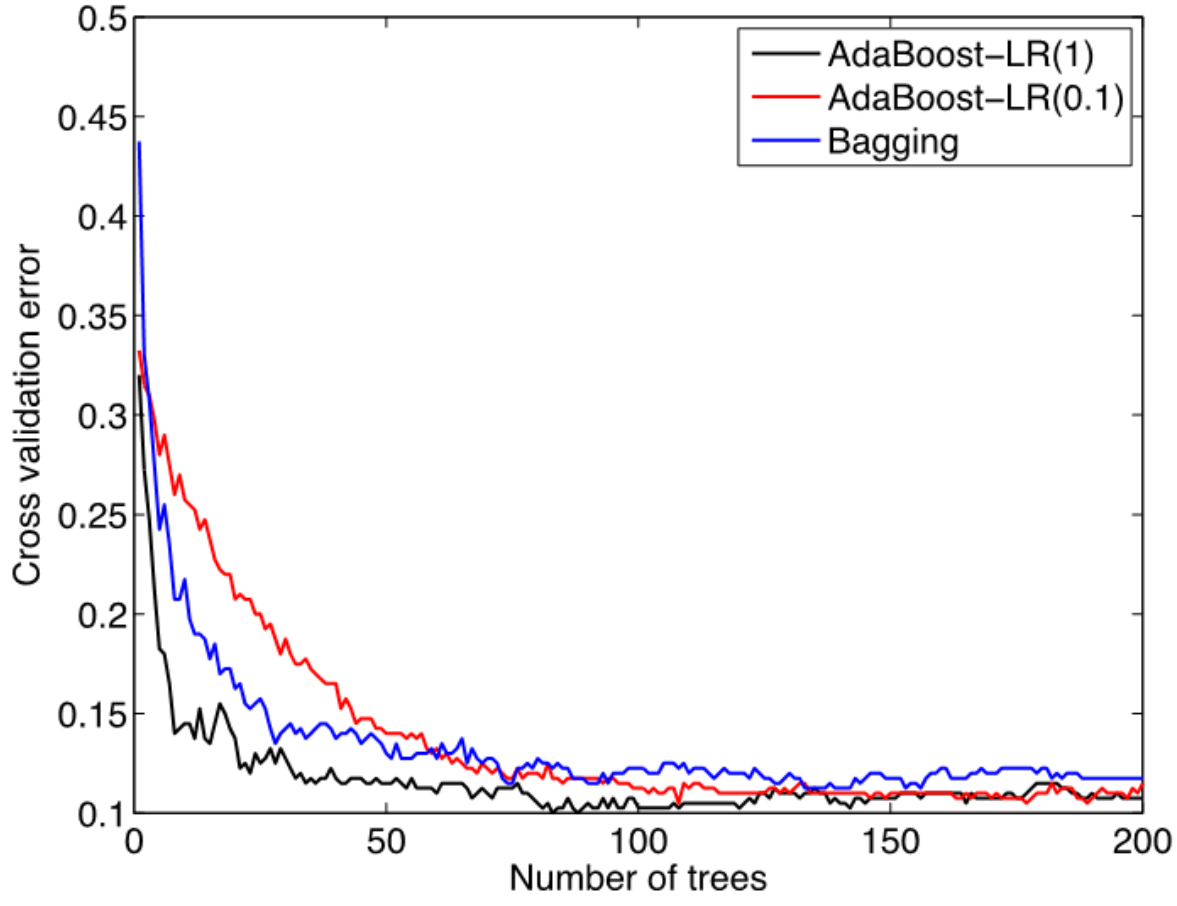
```
t = ClassificationTree.template('MinLeaf',5,'Prune','on');
% set up the individual tree model

% Fit Boosted tree without shrinkage
ens1 = fitensemble(X,y,'AdaBoostM2',200,t,'crossval',...
                   'on','LearnRate',1);
plot(1:200,kfoldLoss(ens1,'mode','cumulative'),'k');

% Fit Boosted tree with shrinkage
ens2 = fitensemble(X,y,'AdaBoostM2',200,t,...
                   'crossval','on','LearnRate',.1);
plot(1:200,kfoldLoss(ens2,'mode','cumulative'),'r');

% Fit Bagged model
ens3 = fitensemble(X,y,'Bag',200,t3,'type',...
                   'classification','crossval','on');
plot(1:200,kfoldLoss(ens3,'mode','cumulative'),'b');
```

Example - classification of zip data



Cross validated error vs. number of trees

Random forests

- Refinement of bagged trees.
- Random forest tries to improve on bagging by *decorrelating* the trees, and reduce the variance.

From Bagging and Boosting to Random Forest

- Is a substantial modification of bagging.
- Builds a large collection of *de-correlated* trees, and then averages them.
- Random forest perform similar to boosting.
- Random forest and boosting are both simple to tune and train.
- RF are popular and many implementations exist.
- The trees are independent and the code can be parallelized.

Random Forest algorithm

1. Define number of trees, B. Typically a few hundred (overfitting is not a problem)
2. For $b = 1$ to B repeat steps (a) - (b)
   a. Take a random sample of size N with replacement from the data (bootstrapping).
   b. Repeat until minimum node size, n_{min}, is reached (do not prune):
      - Take a random sample without replacement of the predictors (of size $m < p$)
      - Construct the first CART partition of the data (pick the best split among the m variables).
3. Output: B trees

Key ingredient

Why pick a random subset of variables in each split?

Hint: Bagging variance again

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Using a Random Forest for prediction

Classification

- Drop data down each tree in the forest
- Classify according to majority vote of B trees

Regression

- Drop data down each tree in the forest
- Prediction is

$$\hat{y} = \frac{1}{B} \sum_{i=1}^{B} T_b(x)$$

where $T_b(x)$ is the estimate of x for each bth random tree.

Random forest model selection

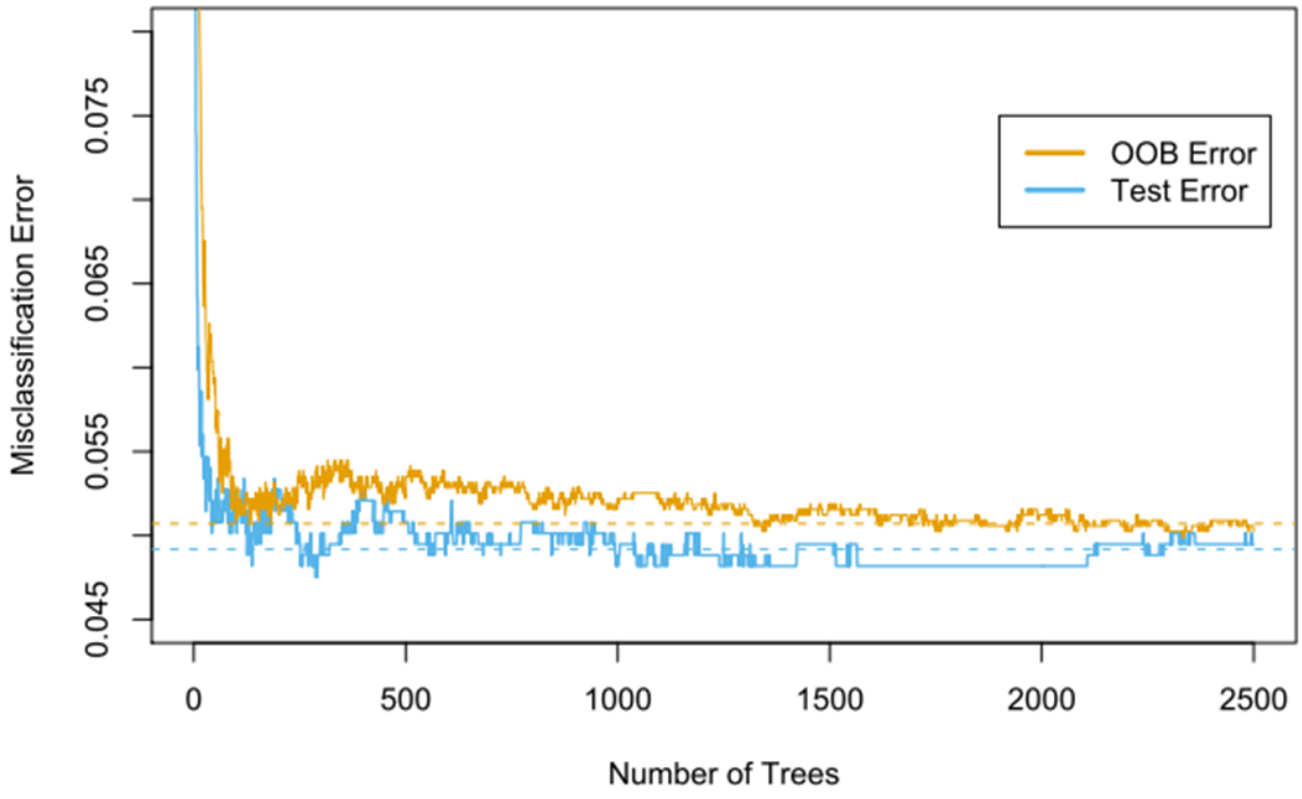
Out-of-bag estimates

- Samples not included in each bootstrap sample are called out-of-bag (OOB) samples.
- These can fairly be used for assessing individual tree performance!
- Using OOB samples for their corresponding trees, we obtain a missclassification rate.
- Results are similar to cross validation.

Random forest model selection

- OOB samples provide a way of constructing trees in a single sequence.
- As the forest is growing:
  - Assess prediction error using OOB samples.
  - Stop when error no longer decreases.

Random forest model selection



Comparison: OOB vs. test set error

Some heuristics for choosing parameters

- Classification
  ```
  m = floor(sqrt(p));
  ```
- Regression
  ```
  m = floor(p/3);
  ```
- However, we should tune them as they depend on the problem; using e.g. OOB error estimates.

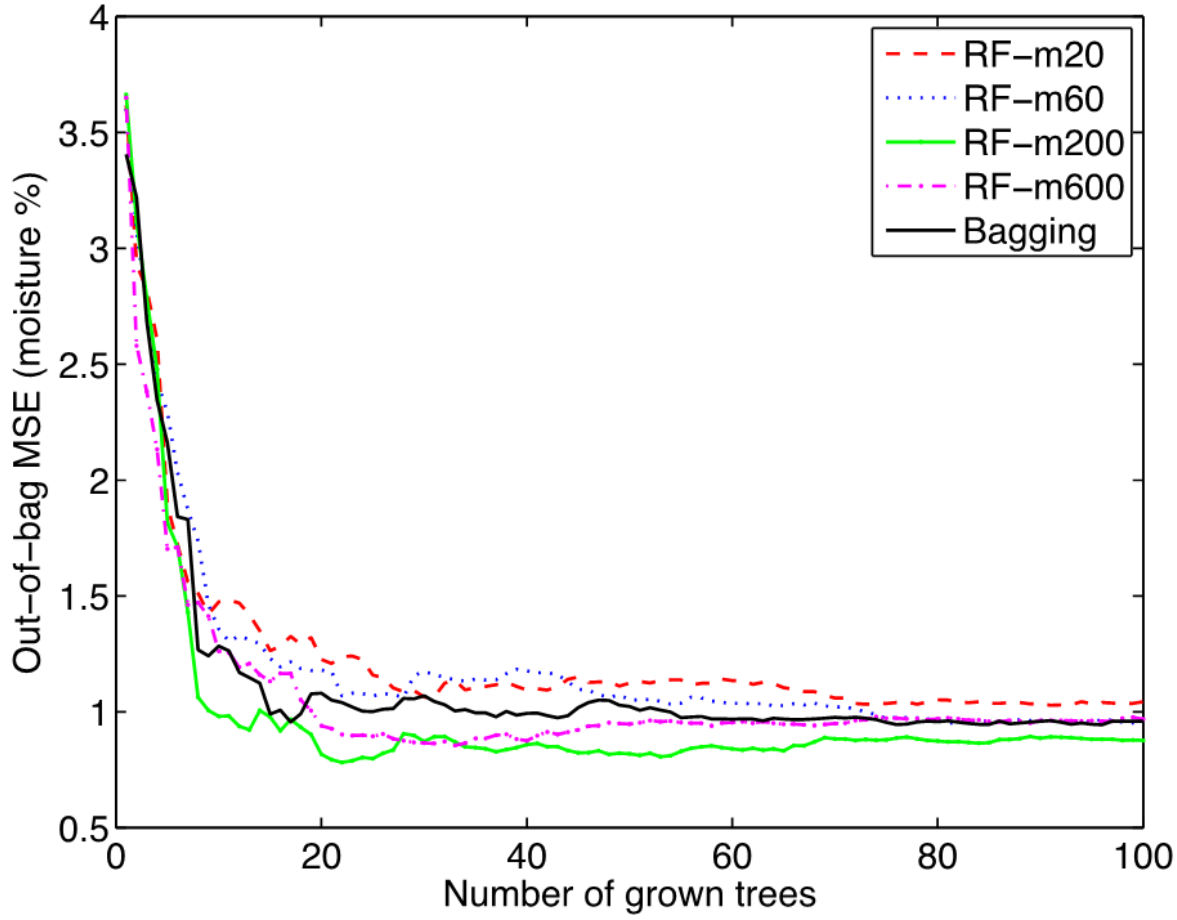
Example of Bagging and Random Forests for sand data set

```
B=100; % Number of trees to grow

% Random Forest regression
Brf20 = TreeBagger(B,X,Y,'oobpred','on','Method',...
                   'regression','NVarToSample',20,...
                   'MinLeaf',5);

% Bagging regression
Bbag = TreeBagger(B,X,Y,'oobpred','on','Method',...
                  'regression','NVarToSample',...
                  'all','MinLeaf',5);
```

Example of Bagging and Random Forests for sand data set



More on comparisons

- $n << p$
- Variable importance
- Proximity plots

A few notes on the model

Bias and variance:

Bagging and Random Forests have the same bias as the bias of an individual tree. This means that the gain obtained in prediction is due to variance reduction.

Boosting lowers bias as well as variance. Hence, we can use small trees.

Connection to ridge

- In ridge regression we see a similar bias-variance trade-off which indicates that bagging and random forests are suitable for $n << p$ problems.
- The ensamble averaging in RF reduces the contribution of any one variable much like the shrinkage in ridge regression.
- This is particular when m, the number of candidate variables in each split, is small.

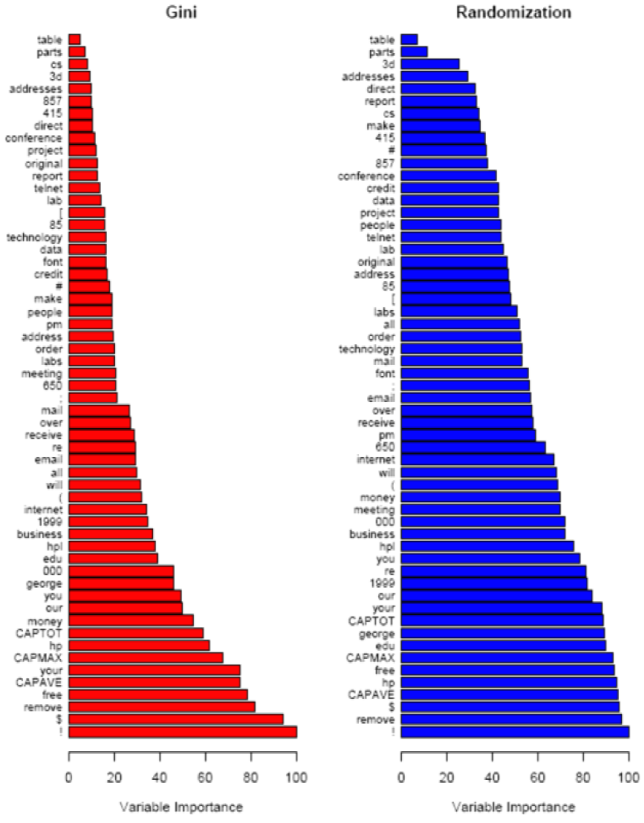
$p > n$

- Random forests work also with more variables than observations
- ...but sometimes has problems
  - When?

Random forests and variable importance

- Two measures (the same can be done for boosting)
  - The Gini index
  - An OOB estimate
- **Gini:** The improvement in the split-criterion at each split is accumulated over all the trees for each variable.
- **OOB:** Measures prediction strength by first dropping the OOB sample down the tree, then permuting the values for the jth variable and computing the prediction accuracy again. An average of the difference in accuracy for all the trees gives the measure of importance for variable j.

Random forests and variable importance



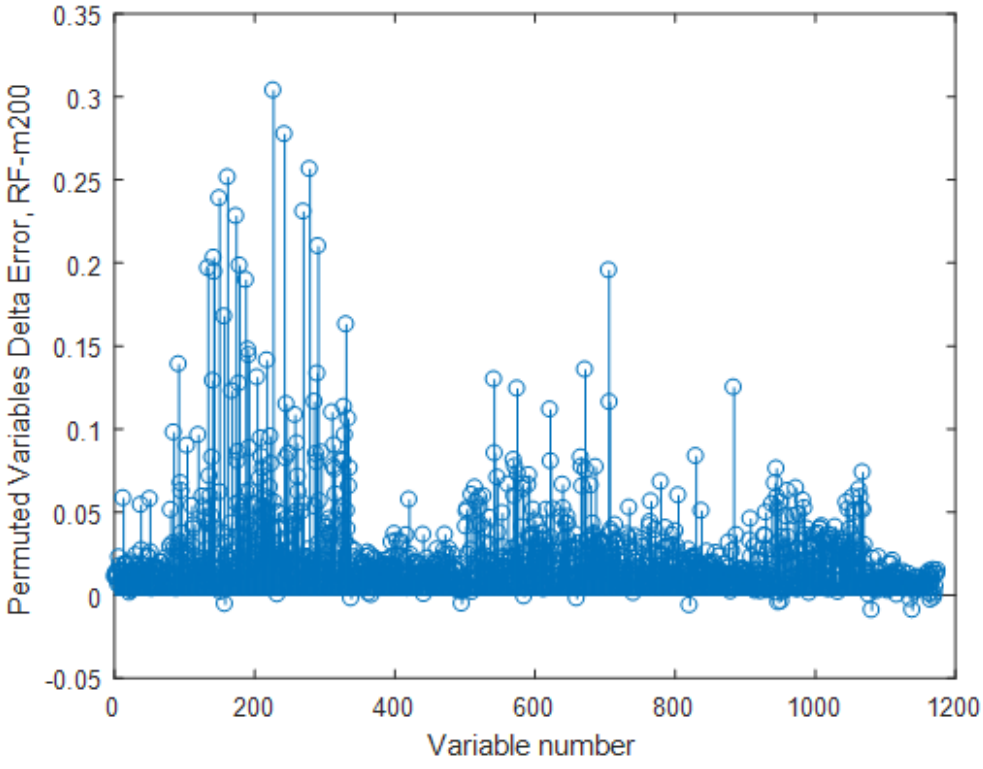
Variable importance for a random forest classification.

The left plot is based on the Gini index whereas the right plot is based on OOB randomization.

The OOB approach tends to spread the importance more uniformly. The rankings are usually similar.

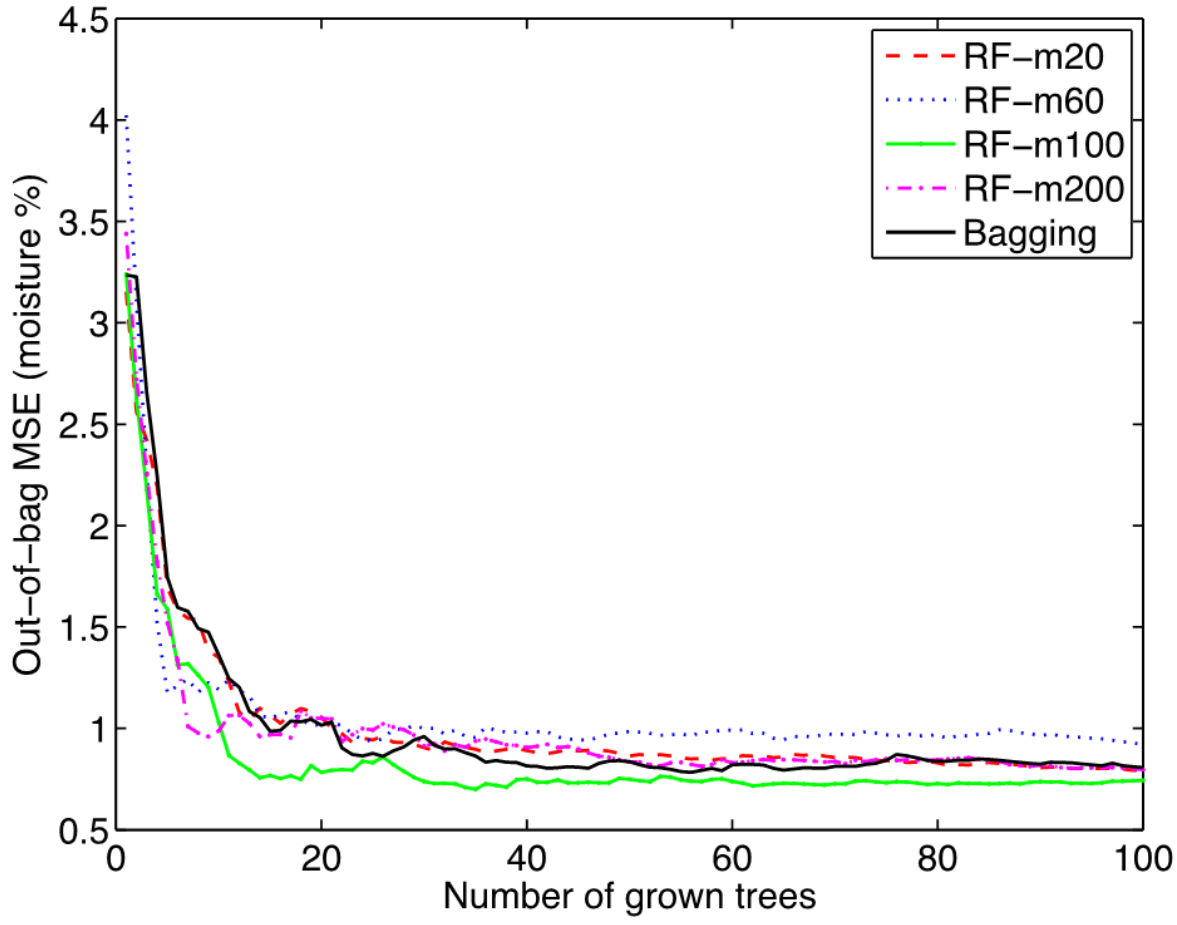
OOB variable importance for the sand data set

```
Brf200 = TreeBagger(100000,X,Y,'OOBVarImp','on',...
                    'Method','regression',...
                    'NVarToSample',200,'MinLeaf',5);
stem(Brf200.OOBPermutedVarDeltaError);
```



It takes many trees to get good precision in importance, here 100 000 trees.

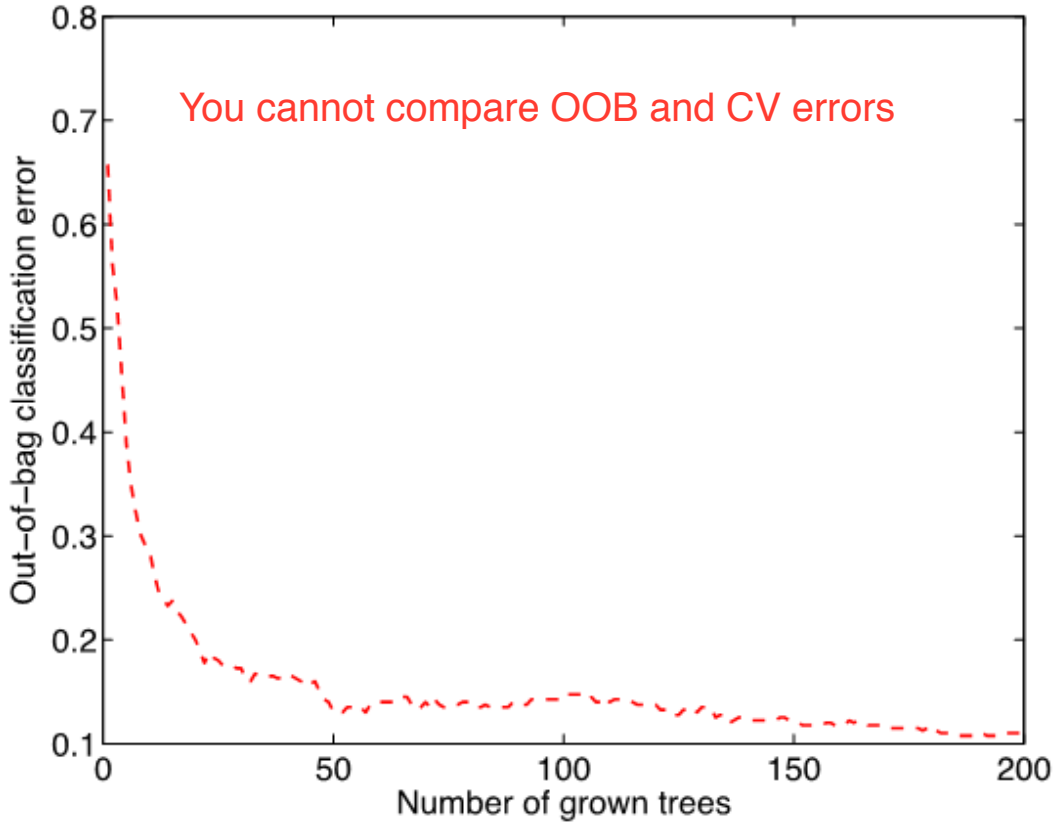
Feature selection using variable importance for the sand data set



Out-of-bag MSE (moisture %) vs. number of grown trees. Based on variables with Delta Error > 0.025 (previous slide).

Example of Random Forest for zip data classification

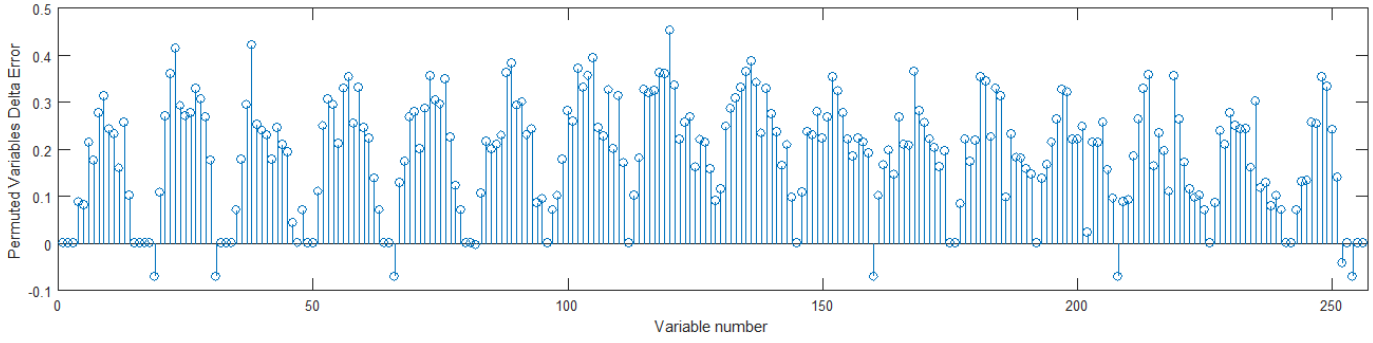
```
rf = TreeBagger(200,X,y,'OOBVarImp','on',...
                'Method','classification',...
                'NVarToSample',16,'MinLeaf',5);
```

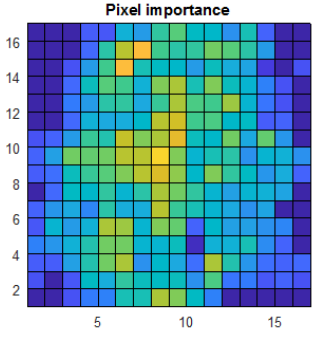


Pitfall: Direct comparison between CV-error and OOB-error.

Out-of-bag classification error vs. Number of grown trees.

Example of Random Forest for zip data classification





Permuted Variable Delta Error vs. Variable number.

Proximity matrix and plots

- An $n \times n$ proximity matrix is created by increasing the proximity value between two OOB samples by one when they end up in the same terminal node.
  - Hence, a large value in (i, j) reflects that observation i and j are similar according to the classifiers.
- The proximity matrix can be represented in two dimensions using multidimensional scaling. The idea behind this is to find a low-dimensional representation which preserves the pairwise distances.
- In general the proximity plots look like stars where points far from the decision boundary are the extremities, and the points close to the boundary are near the center.

Proximity plots and decision boundaries

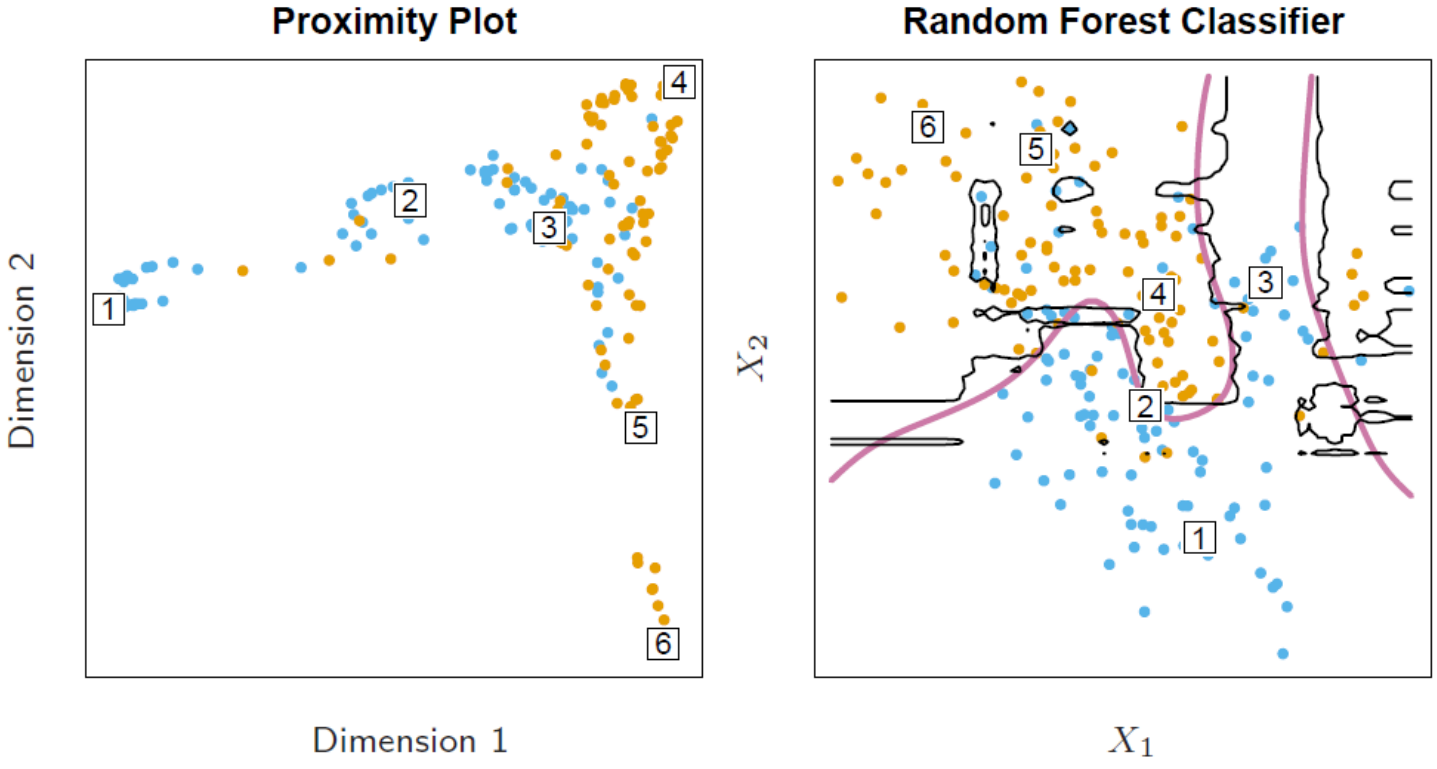
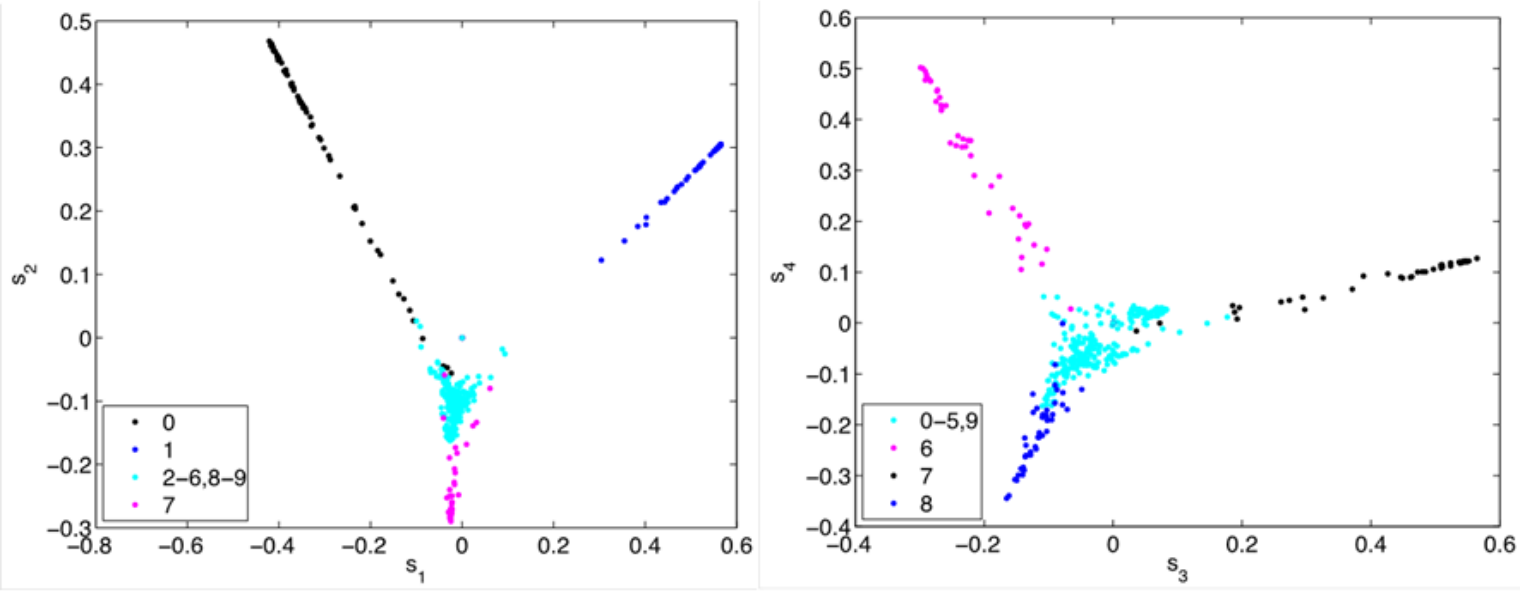


FIGURE 15.6. *(Left): Proximity plot for a random forest classifier grown to the mixture data. (Right): Decision boundary and training data for random forest on mixture data. Six points have been identified in each plot.*

Proximity plots and decision boundaries

```
[S,E] = mdsProx(fillProximities(rf));
plot(S(:,1),S(:,2));
plot(S(:,3),S(:,4));
```



Identify one of the observations which are in the middle - look at the digit. Why is this observation harder to classify?

Questions