Ensemble methods

Line Clemmensen

DTU

02582 Computational Data Analysis, 2019

Today's Lecture

- Recap CART and Bagging
- Theory behind and use of
  - Random forest
  - Boosting

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 $2^{k-1} - 1$

Last Week

- Bagging
  - Draw B bootstrap samples of data and build a model on each bootstrap sample.
    - ⋆ Idea: Empirical distribution mimics the distribution X is drawn from.
  - Average predictions or take majority vote.

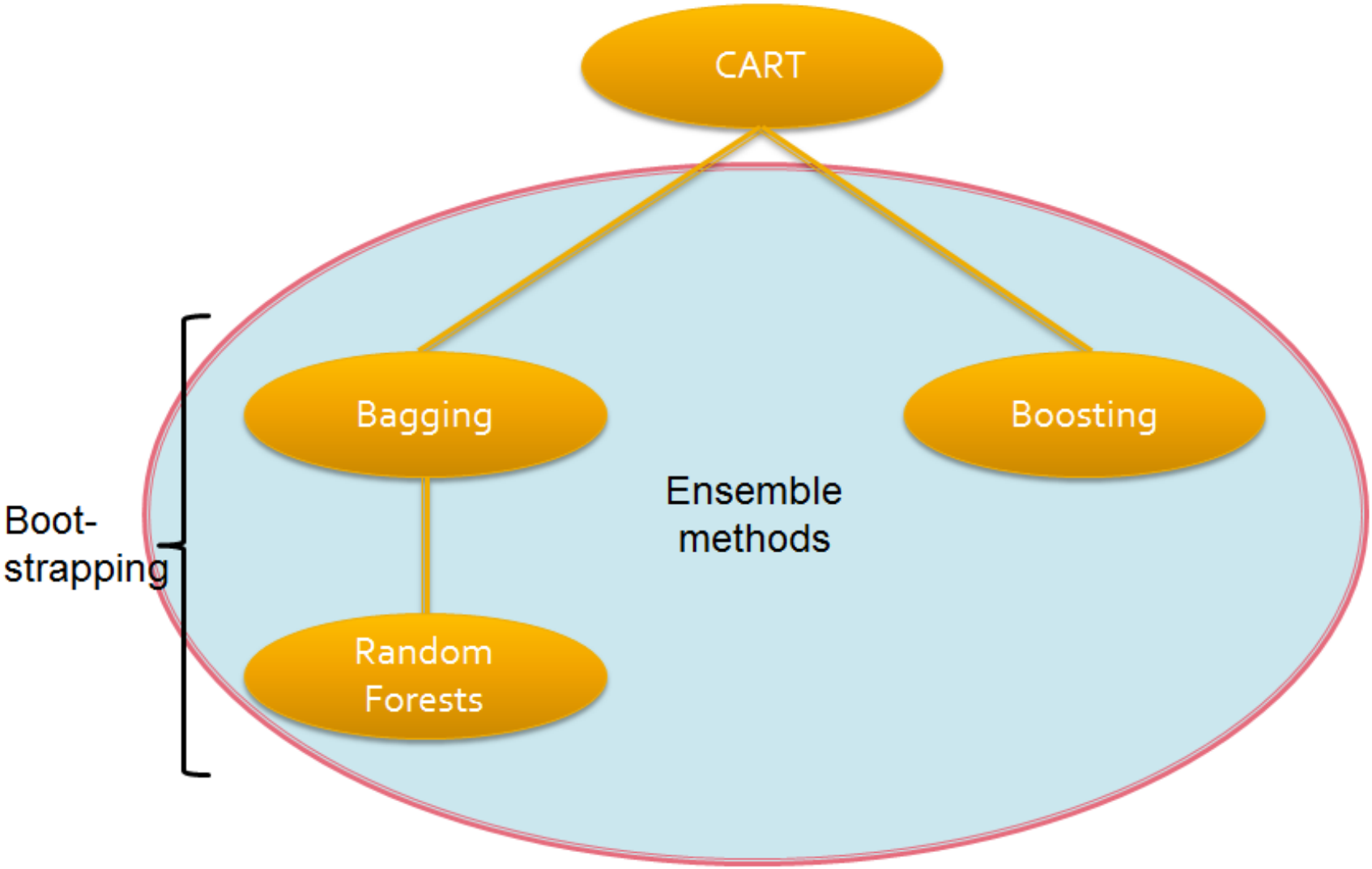
Advantage:

- Handles missing data.
- Can take both continuous and categorical variables as input.
- Averages out noise (lowers variance).

Drawbacks:

- Does nothing to the bias.
- Variance limited by the correlation between trees.

Bagging vs Boosting vs Random forest



Bagging - bias (for reference)

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 (for reference)

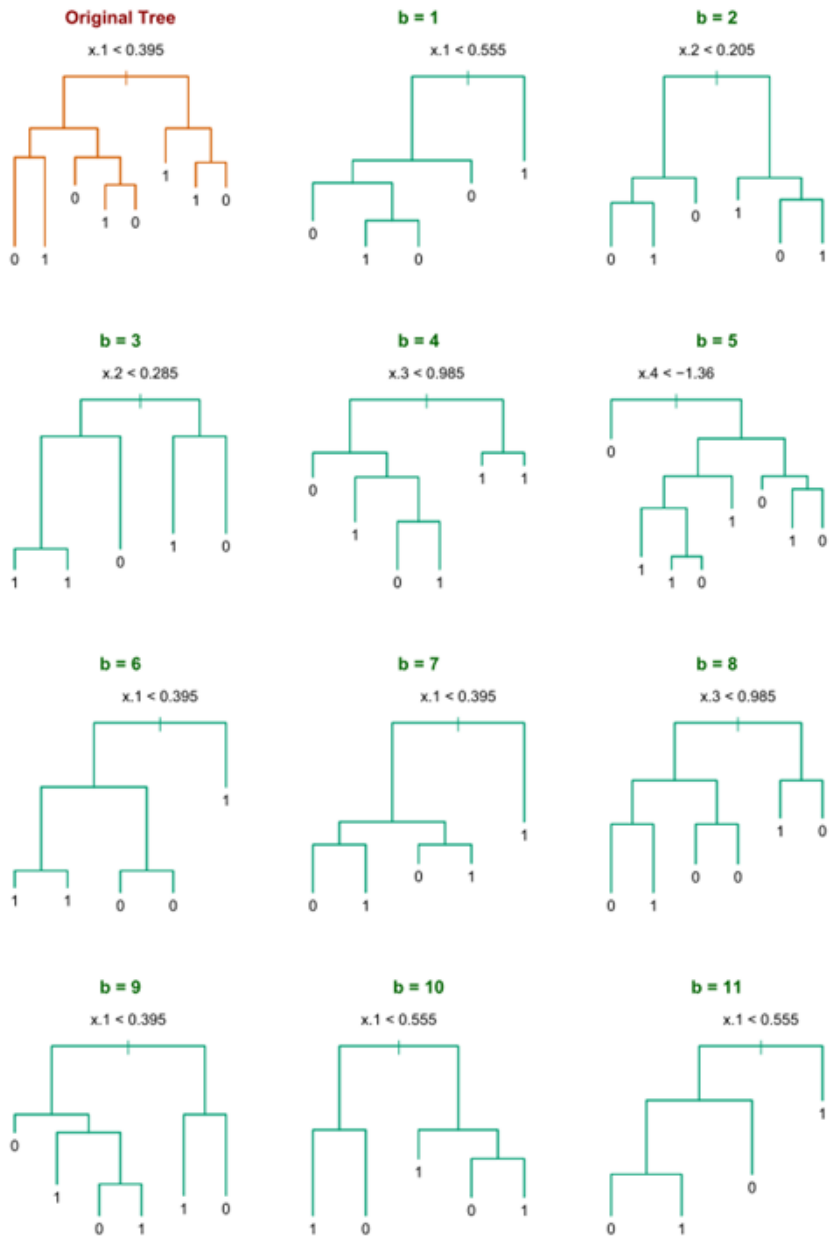
The variance of the average of bagged estimates is (ρ is the correlation between pairs of trees)

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

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 trees



Random forests

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

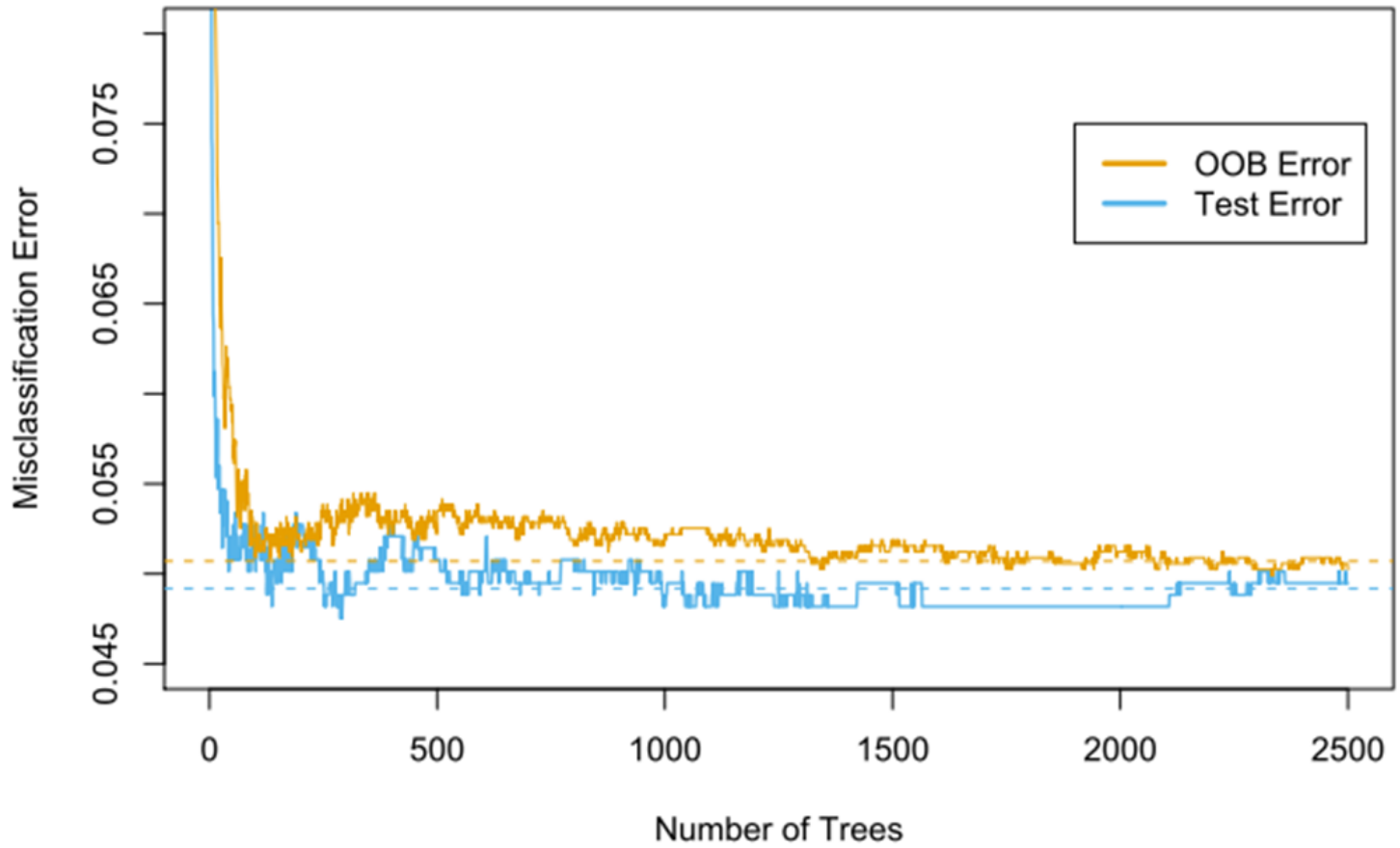
Out-of-bag estimates

- 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

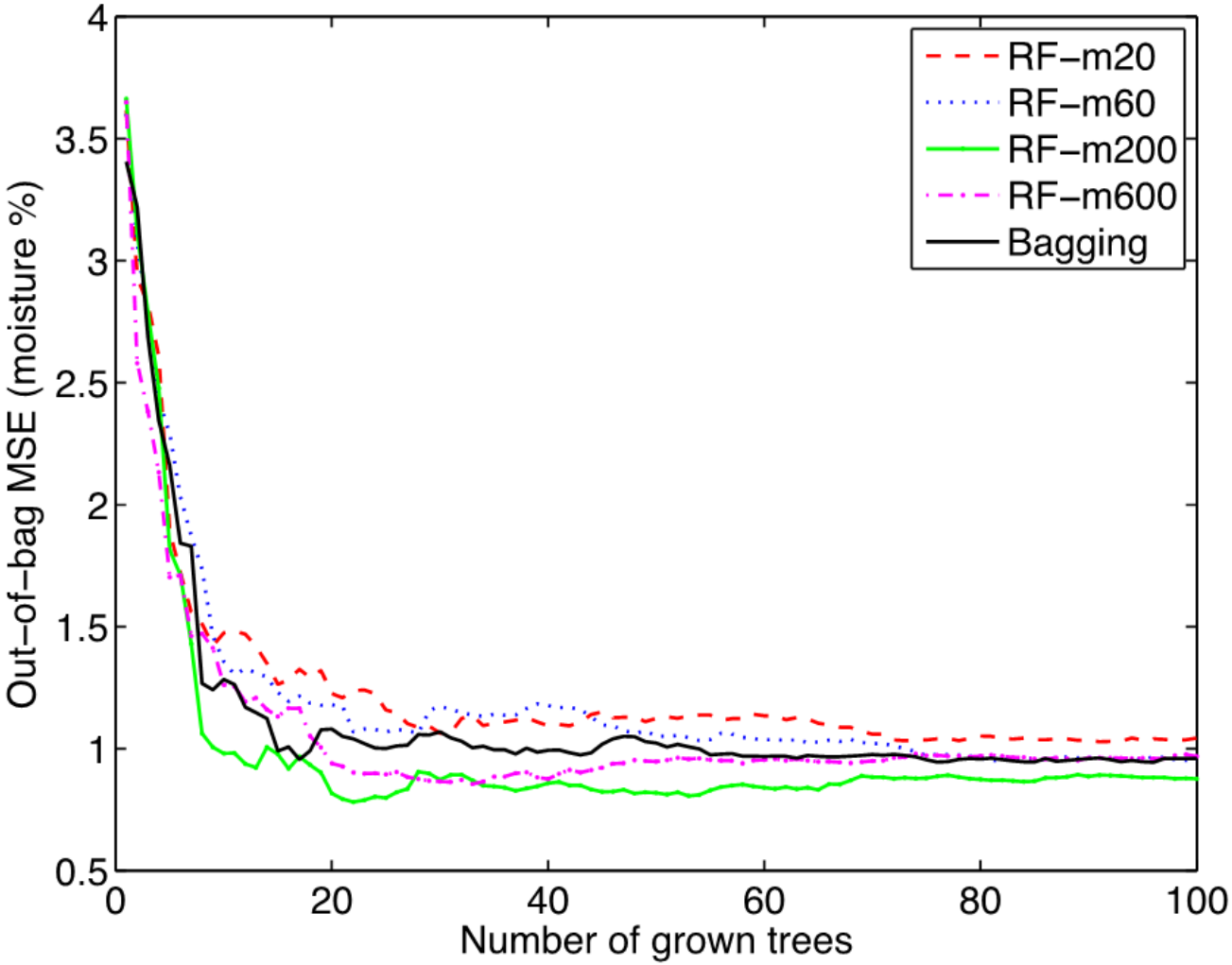
  ```
  m = floor(sqrt(p));
  ```

- 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, with varying m



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.

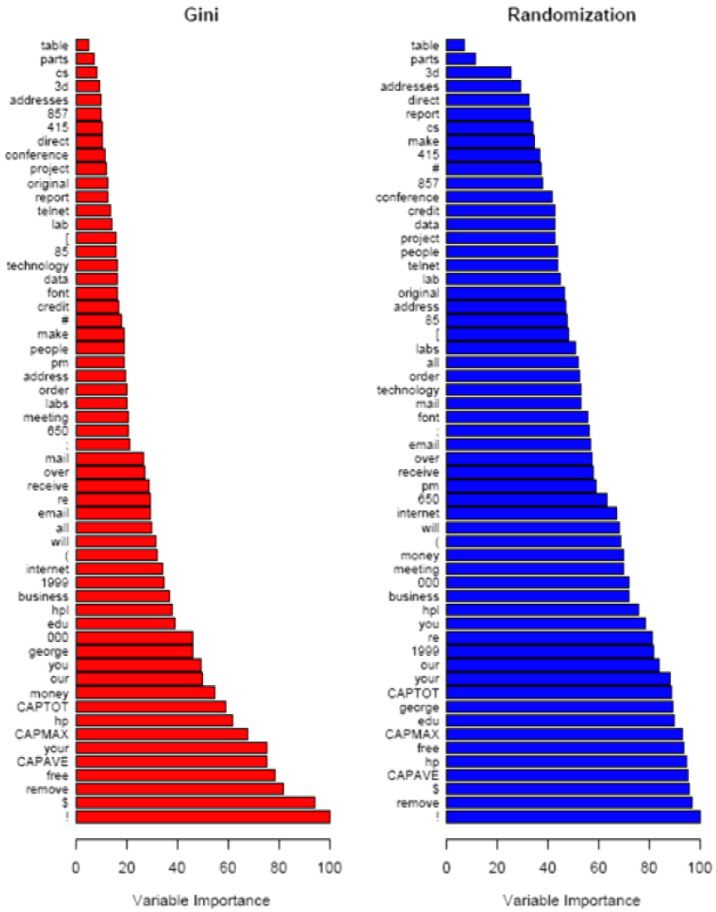
$p > n$

- 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



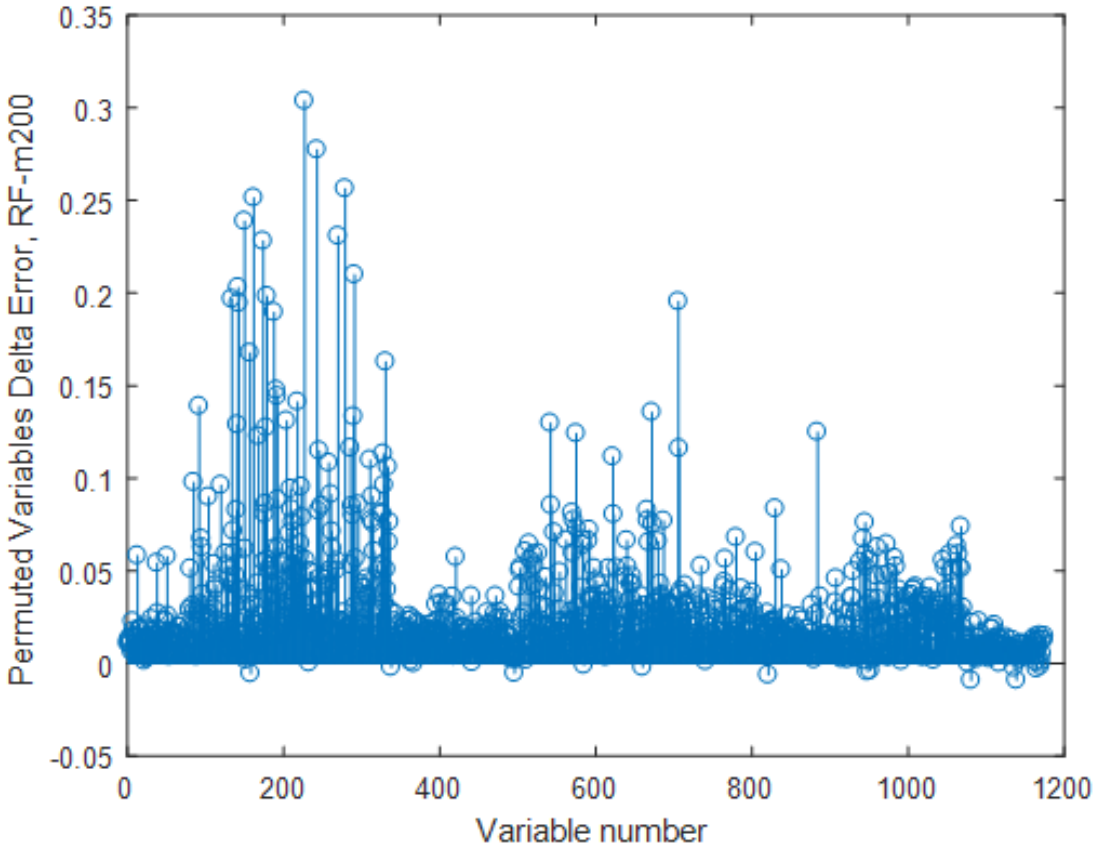
Variable importance for a random forest classification.

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.

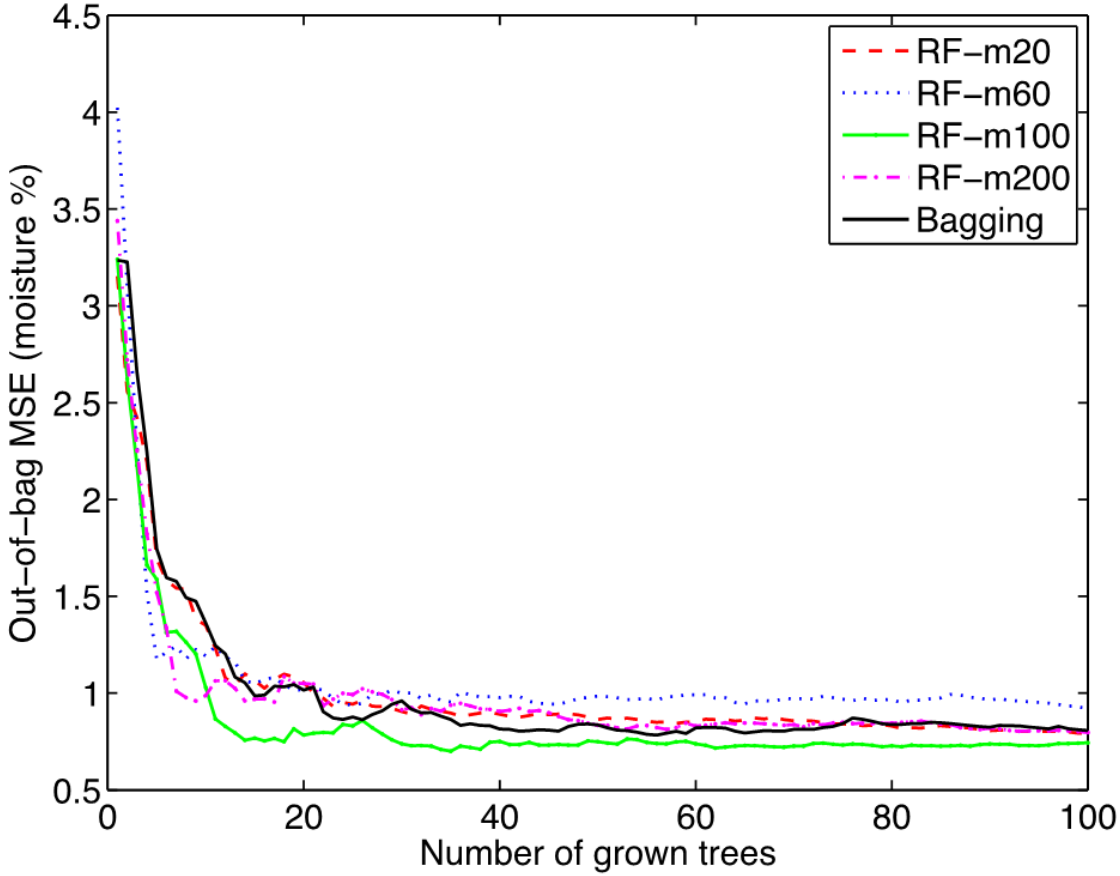
OOB variable importance for the sand data set

```
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.

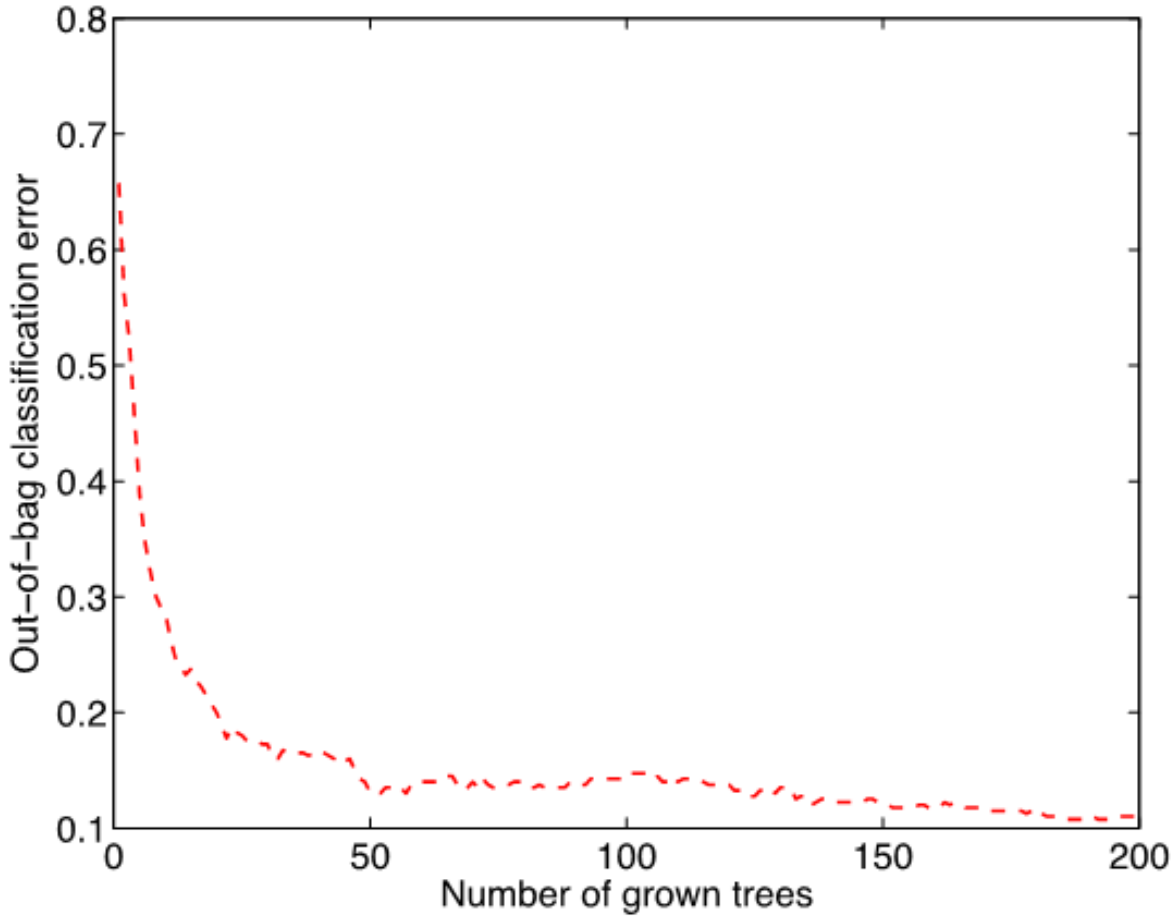
Feature selection using variable importance for the sand data set



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

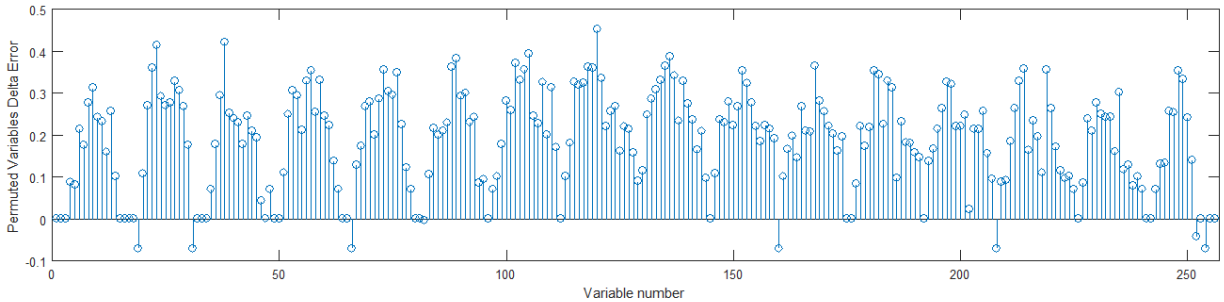
Example of Random Forest for zip data classification

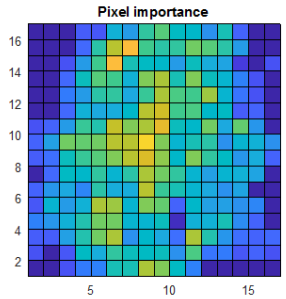
```
rf = TreeBagger(200,X,y,'OOBVarImp','on',...
                'Method','classification',...
                'NVarToSample',16,'MinLeaf',5);
```



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

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

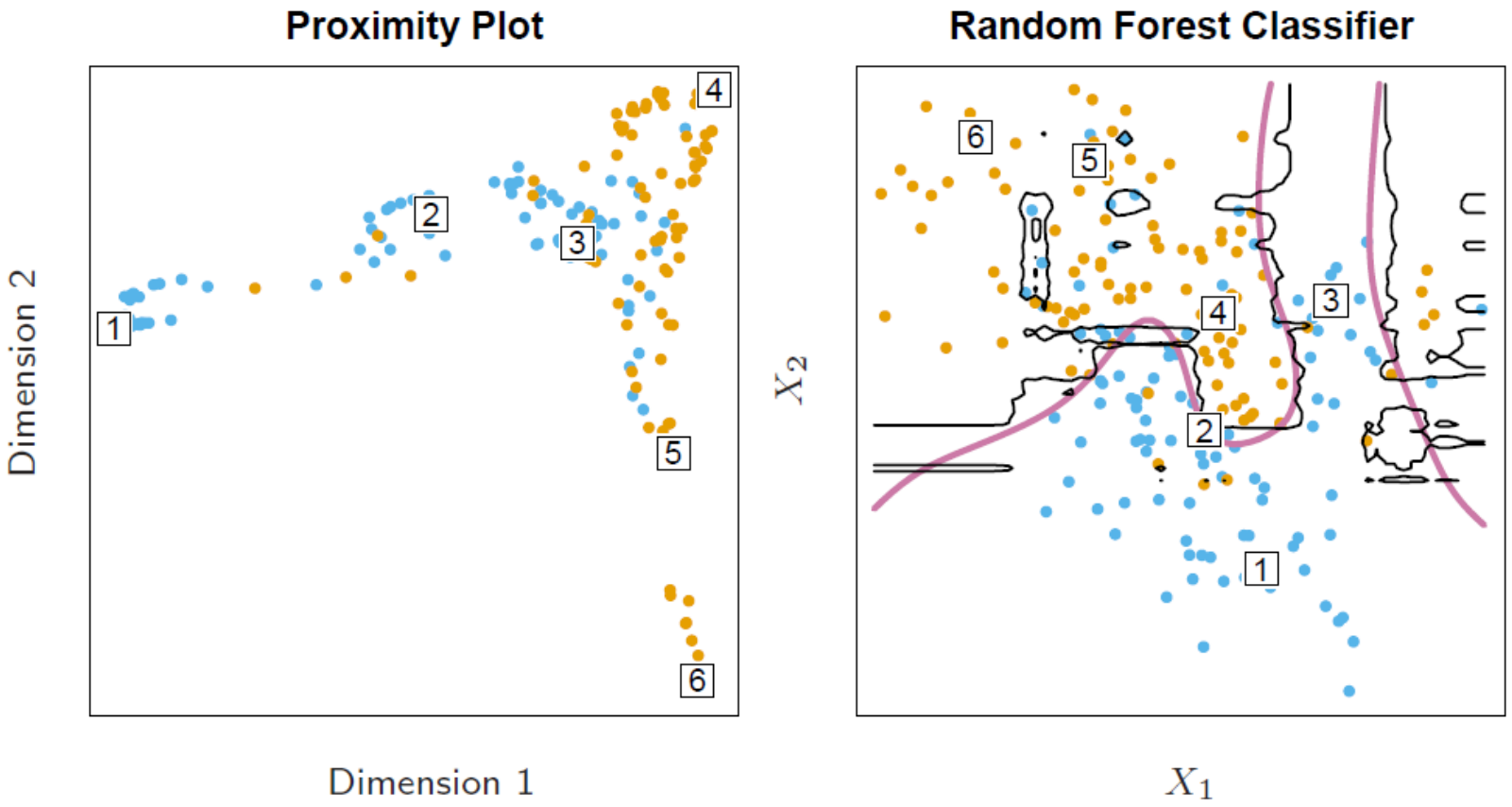
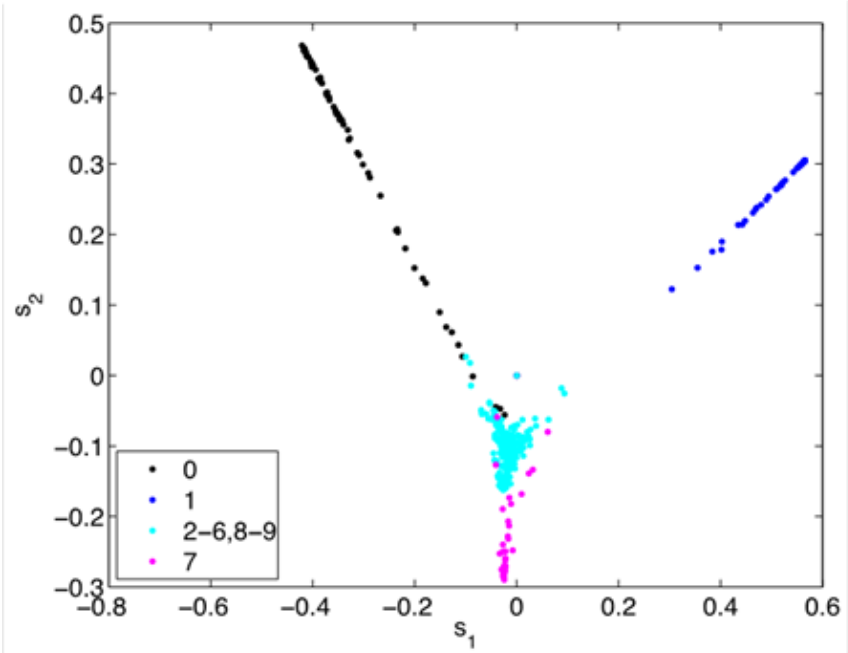
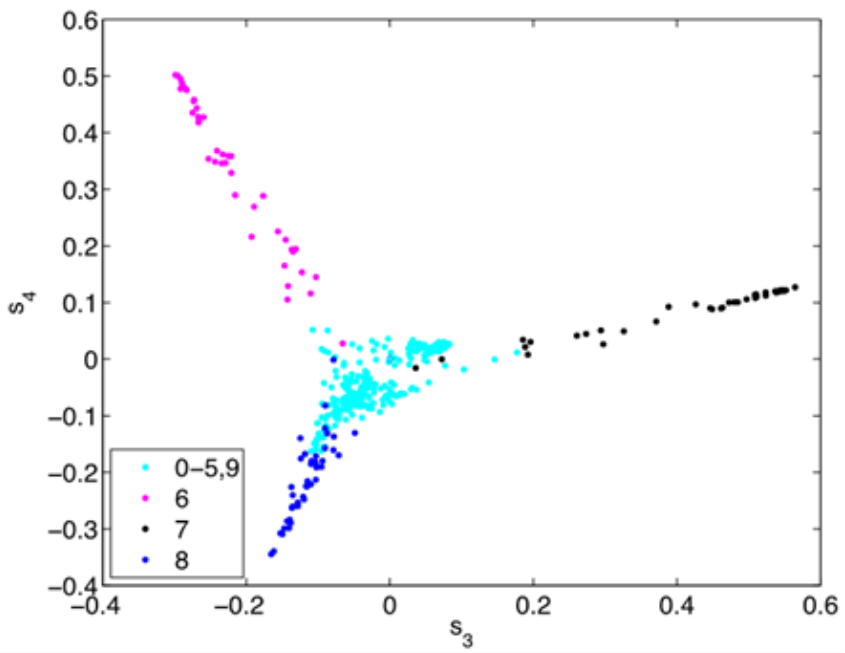


FIGURE 15.6. *(Left): Proximity plot for a random forest classifier grown to the mixture data. (Right): Decision boundary and training data for random forest on mixture data. Six points have been identified in each plot.*

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?

Boosting

- 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

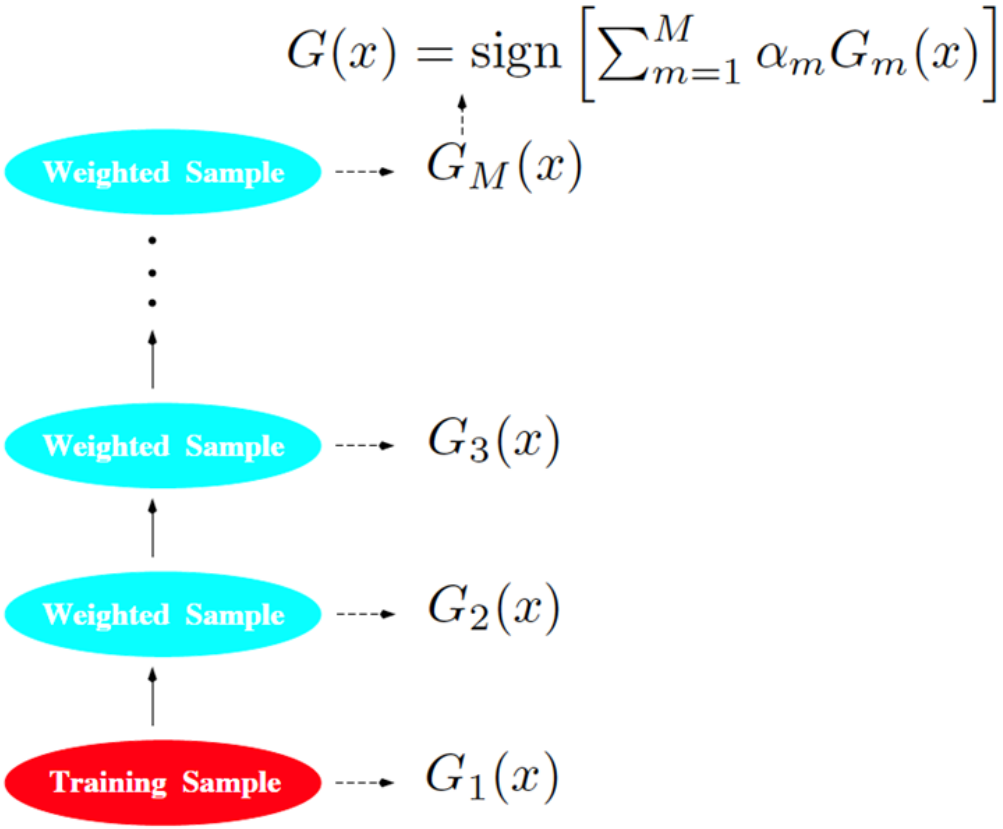
- A set of weak classifiers (small trees) are grown in an adaptive manner to remove bias (hence they are not independent).
- Weights are put on the observations
  - weights are updated to emphasize missclassifications.
- The final estimator is a weighted average of all the trees.
- **Boosting shrinks 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

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: $G_m(x) \in \{-1, 1\}$

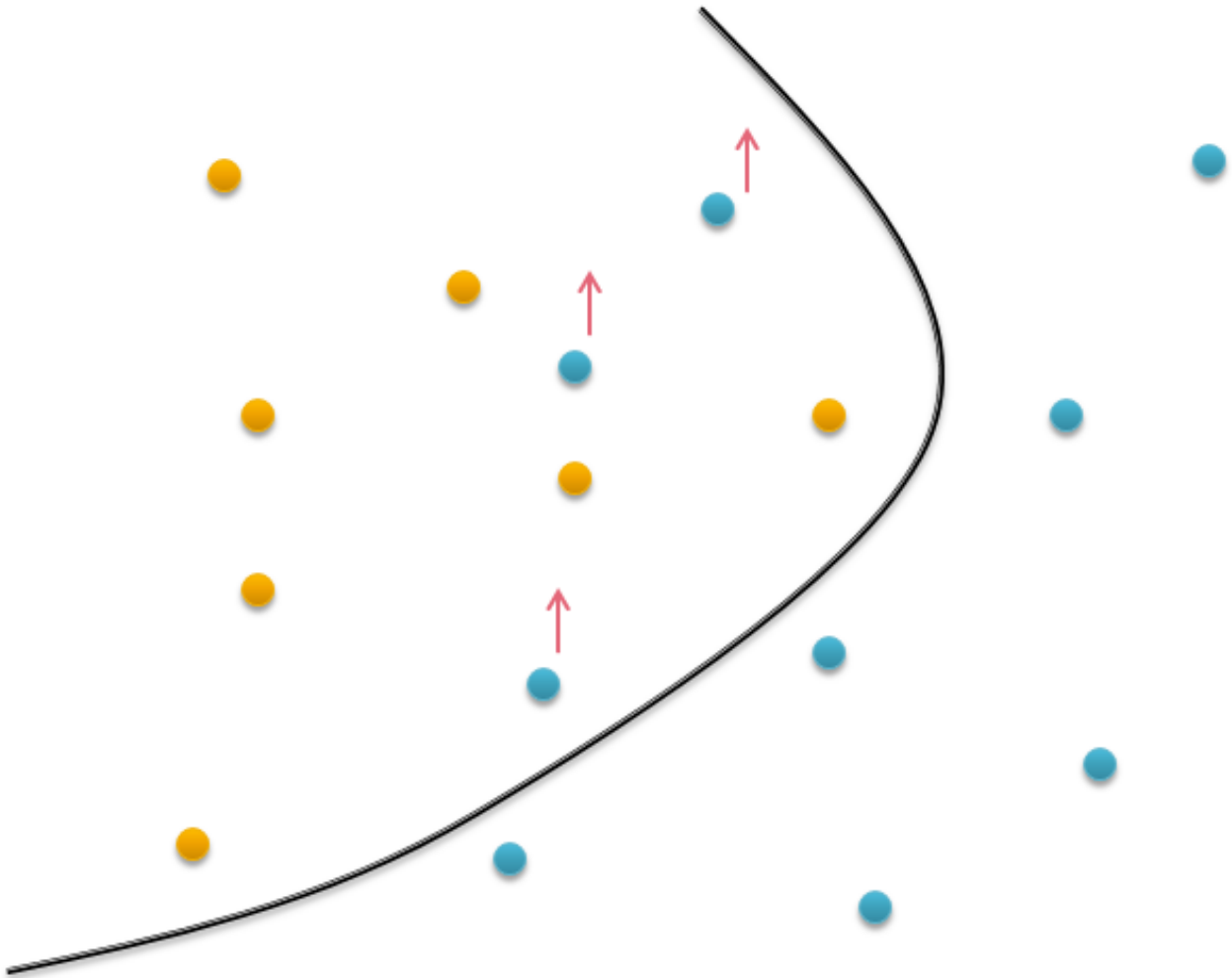
AdaBoost.M1 algorithm

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

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

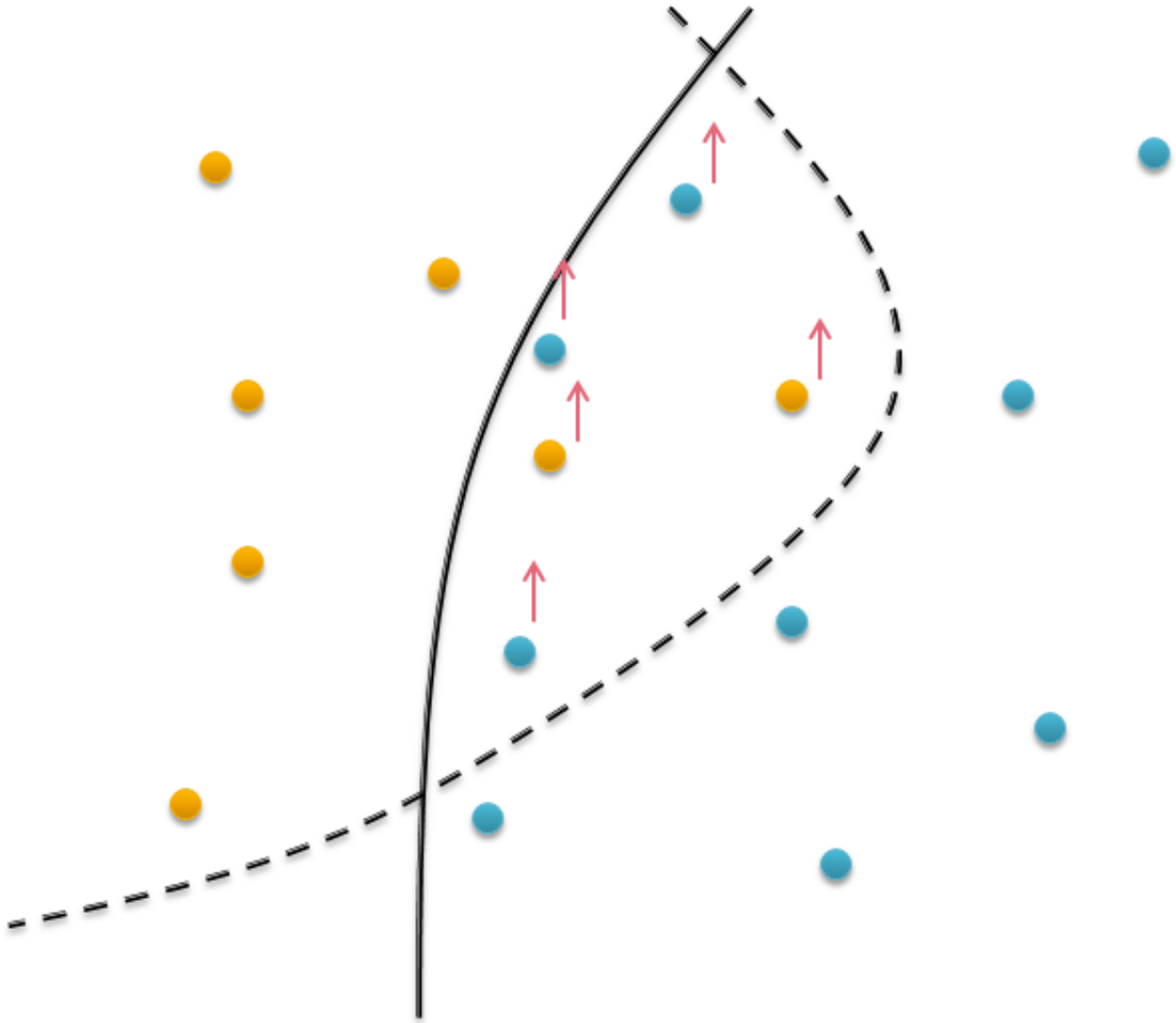
   c. Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$.
   d. Update weights for $i = 1, ..., N$:
   $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(x_i))]$
   and renormalize to w_i to sum to 1.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Boosting example



$a_1, G_1(x)$

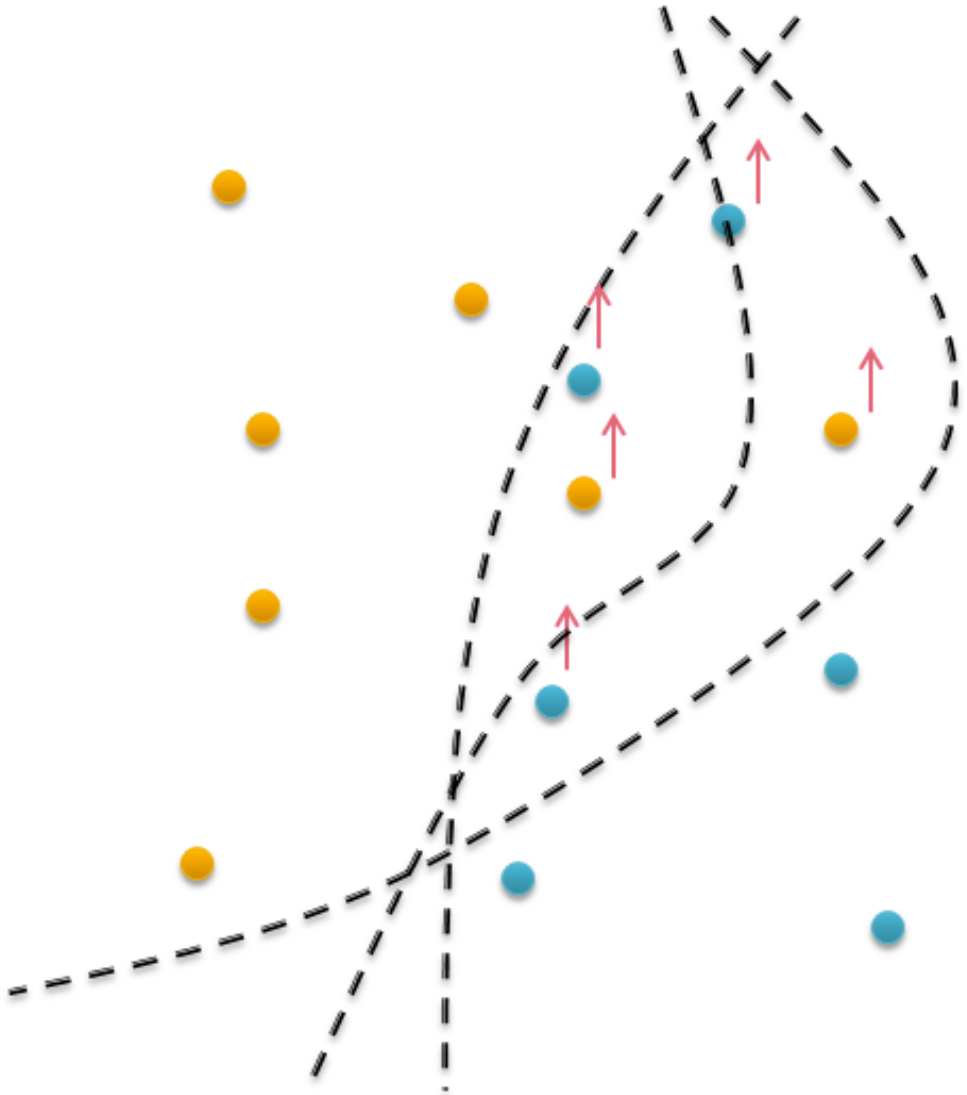
Boosting example



$a_1, G_1(x)$

$a_2, G_2(x)$

Boosting example



$a_1, G_1(x)$

$a_2, G_2(x)$

$a_3, G_3(x)$

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

Boosting

- 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)

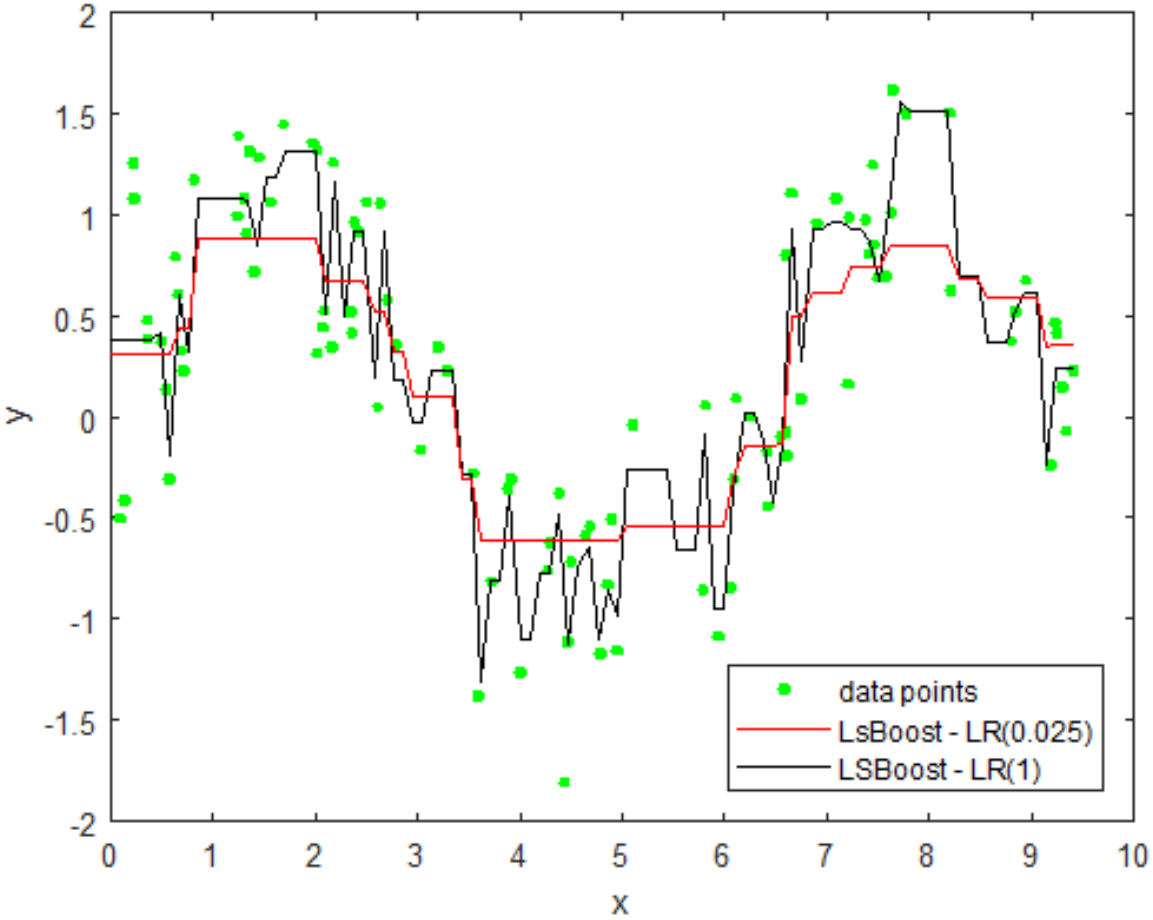
```
% Boosting - Regression

% 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

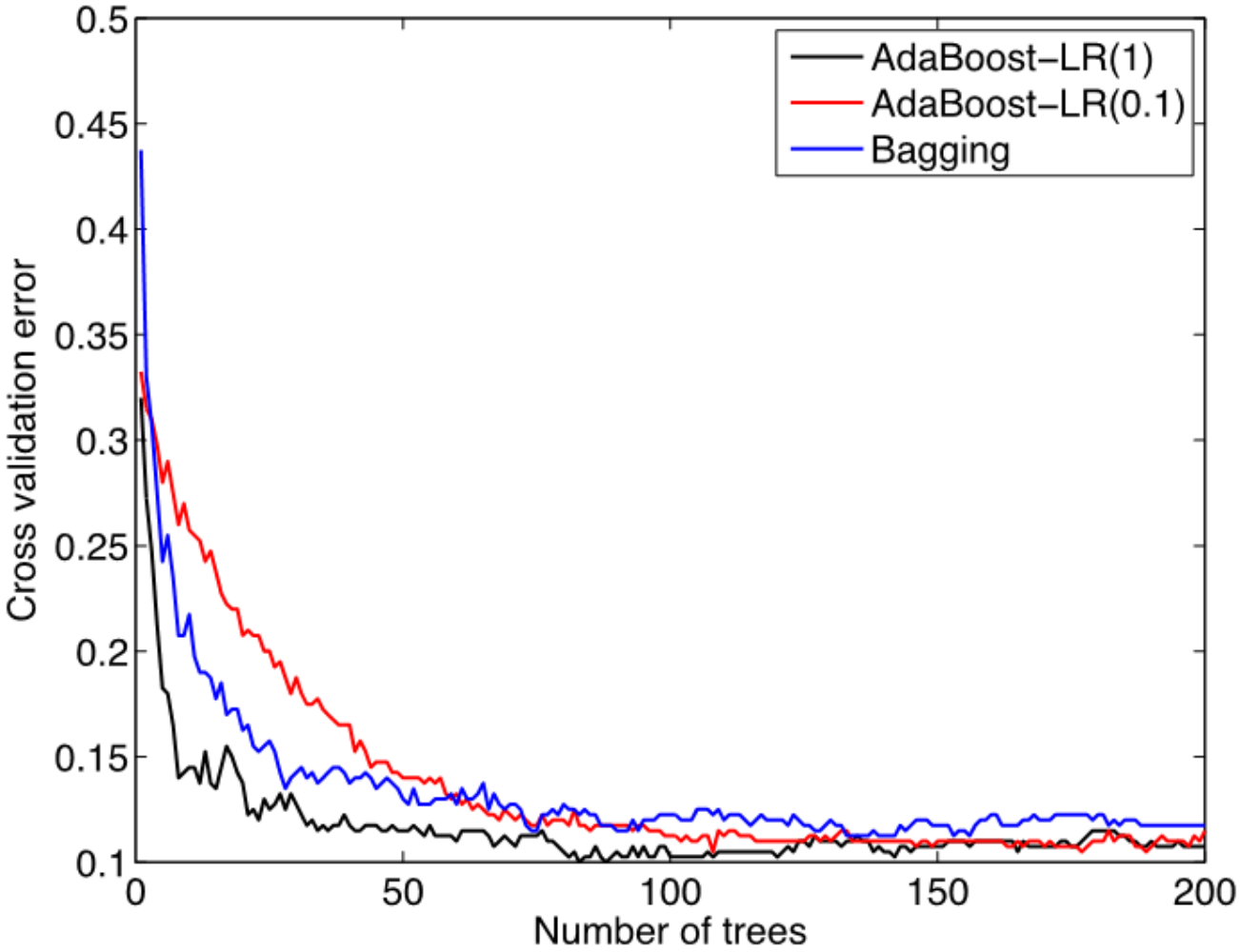
```
t = ClassificationTree.template('MinLeaf',5,'Prune','on');
% set up the individual tree model

% 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

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.

Boosting - summing up

- Based on *weak learners*
  - High bias, low variance
  - Classify more than 50% correct
  - E.g. shallow classification trees (a stump is a single split) or rules of thumb
- Combine these through an optimally weighted majority vote to a *strong classifier*

Boosting - moving beyond

- Additive Models
- Forward Stagewise Additive Models
- Relation to logistic regression
- A few properties
- Loss functions
- Gradient boosting

Boosting and Additive Models

Consider the boosting classification equation (10.1 in ESL):

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

We can see the individual classifiers, $G_m(x)$, as basis functions, i.e. we have the basis expansion

$$F(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m). \tag{2}$$

where input $b(x, \gamma_m)$ is a tree and γ_m parameterizes the splits.

Boosting and Additive Models

- Traditionally the parameters γ_m and β_m would be fit *jointly* using e.g. least squares or maximum likelihood.
- For boosting, we optimize the parameters in a *stagewise* fashion. Meaning, we fit the next tree, but all others are held fixed.
- This slows down the rate of overfitting, as there is no adjustment of the past. Works as *regularization*.

Forward Stagewise Additive Modelling - regression

1. Start with $F_0(x) = 0$ and residual $r = y$, $m = 0$
2. Repeat the following many times (M times):
   a. $m = m + 1$
   b. Fit a CART regression tree to r resulting in $g(x)$
   c. Set $f_m(x) = \epsilon g(x)$. *Shrink* it with shrinkage-parameter ϵ
   d. Set $F_m(x) = F_{m-1}(x) + f_m(x)$, $r = r - f_m(x)$

Adaboost and Stagewise Modelling

- ESL 10.4: AdaBoost is equivalent to stagewise modelling with an exponential loss function, i.e. AdaBoost builds an *additive logistic regression model*:

$$F(x) = \log \frac{P(y=1|x)}{P(y=-1|x)} = \sum_{m=1}^{M} \alpha_m f_m(x) \tag{3}$$

  by use of stagewise fitting, and with the loss function

$$L(y, F(x)) = \exp(-yF(x)) \tag{4}$$

- Instead of fitting trees to residuals, the exponential loss function leads to fitting trees to weighted versions of the original data.

Proof of relation to logistic regression

Lemma[1] $E[\exp(-yF(x))]$ is minimized at

$$F(x) = \frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)} \tag{5}$$

Proof It is sufficient to minimize the criterion conditional on x:

$$\begin{aligned}
E[e^{-yF(x)}|x] &= P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)} \\
0 = \frac{dE[e^{-yF(x)}|x)]}{dF(x)} &= -P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)} \\
P(y=1|x)e^{-F(x)} &= P(y=-1|x)e^{F(x)} \\
\frac{P(y=1|x)}{P(y=-1|x)} &= e^{2F(x)} \Rightarrow F(x) = \frac{1}{2}\ln\frac{P(y=1|x)}{P(y=-1|x)}
\end{aligned}$$

A factor 2 in difference to logistic regression.

[1] Friedman, Hastie and Tibshirani, The Annals of Statistics, 2000, Vol. 28, No. 2, 337-407

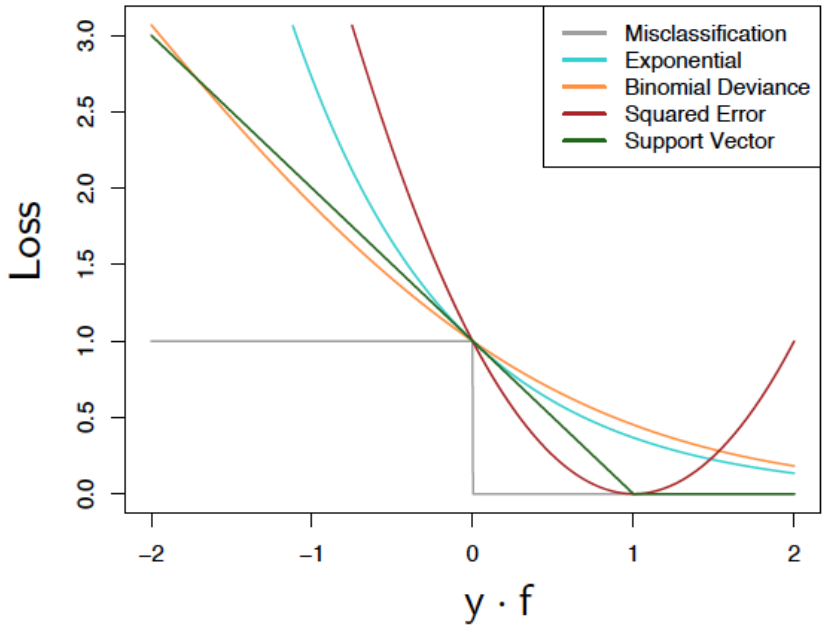
Probabilities from AdaBoost

Using the Lemma we can solve for $P(Y = 1|x)$

$$
\begin{aligned}
e^{2F(x)} &= \frac{P(y=1|x)}{P(y=-1|x)} \\
e^{2F(x)} &= \frac{P(y=1|x)}{1 - P(y=1|x)} \\
e^{2F(x)} - e^{2F(x)} P(y=1|x) &= P(y=1|x) \\
e^{2F(x)} &= P(y=1|x)(1 + e^{2F(x)}) \\
P(y=1|x) &= \frac{e^{2F(x)}}{1 + e^{2F(x)}}
\end{aligned}
$$

Loss function choice

- Exponential loss gives *computational* advantages (basically because it leads to the simple reweighing scheme in AdaBoost).
- Exponential and Bionomial Deviance Losses are *similar* for observations with postive margins.
- Exponentail loss *penalizes heavily* for observations with negative margins compared to Binomial Deviance Loss (exponential vs linear).



Boosting and Noise

- Boosting is not robust to settings where the Bayes error rate is not close to zero (*noisy* settings)
- Boosting is sensitive to *wrongly labeled* observations
- This is in parts due to the exponential emphasis on misclassifications

Boosting Algorithms - General Loss Function

1. Start with $F_0(x) = 0$ and $m = 0$
2. Repeat the following many times (M times):
   a. $m = m + 1$
   b. Compute $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta b(x_i; \gamma))$
   c. Set $F_m(x) = F_{m-1}(x) + \epsilon\beta_m b(x; \gamma_m)$
3. ϵ is the *shrinkage* factor.

Gradient Boosting - Friedman, 2001

- Different *loss functions* can be used
  - Poisson model, Cox model, logistic regression, binomial, etc.
- Different *models*
  - Regression, resistant regression, K-class classification, and risk modelling
- Consists of *additive trees* to e.g. represent the logits in logistic regression
- *Inherits the pros* of trees (variable selection, mix of variables numerical/categorical, missing data handling), and also *improves the cons* - i.e. prediction performance
- *Tree size* is an important factor

Gradient Tree Boosting - algorithm

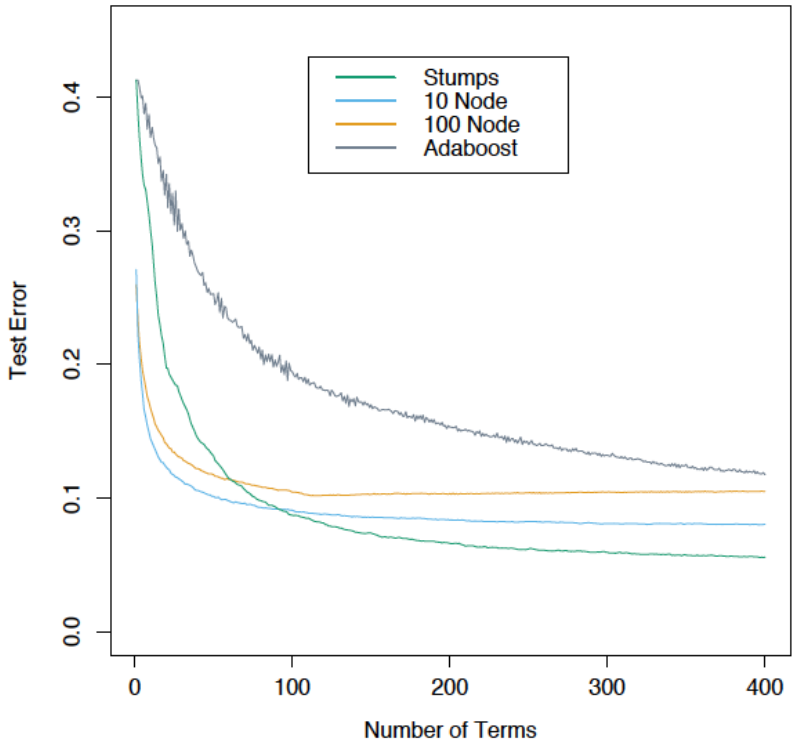
1. Start with $F_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$ and $m = 0$
2. For $m = 1$ to M:
   a. For $i = 1$ to N, compute
   $$r_{im} = -\left[\frac{dL(y_i, F(x_i))}{dF(x_i)}\right]_{F=F_{m-1}} \tag{6}$$
   b. Fit a regression tree to the negative gradient r_{im} giving terminal regions R_{jm}, $j = 1, 2, ..., J_m$.
   c. For $j = 1$ to J_m compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma). \tag{7}$$
   d. Update $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $G(x) = F_m(x)$

Tree Size

ANOVA expansion of our target, η (minimized loss function):

$$\eta(X) = \sum_j \eta_j(X) + \sum_{jk} \eta_{jk}(X_j, X_k) + \sum_{jkl} \eta_{jkl}(X_j, X_k, X_l) + \ldots \quad (8)$$

The depth of the tree determines the interaction order of the model. In example 10.2 the model is additive (no interactions), so stumps perform the best.



Multiclass boosting

- Use the loss function $-\sum_{k=1}^{K} y_k \log(P(y_k = |x))$
- With a score function per class, k: F_k, $k = 1, ..., K$ (initialize as zeros).
- From the scores derive the probabilities:

$$P_k(y = 1|x) = \frac{e^{F_k(x)}}{\sum_{k=1}^{K} e^{F_k(x)}} \tag{9}$$

- Calculate KL-divergence (gradient of loss function): $-g_k(x_i) = y_i - P_k(x_i)$
- Minimize KL-divergence through fitting trees to the negative gradient (the residuals)

Summing up

- Random forests
  - What is the trick again?
  - Pros and cons
- Boosting
  - How does it work again?
  - Pros and cons
  - Relation to additive models
  - Relation to logistic regression
  - Stochastic gradient desent
  - Multi-class