# STATS 415: Ensemble methods

Prof. Liza Levina

Department of Statistics, University of Michigan

# Ensemble Classifiers

Classification trees are simple but often unstable. Solution: combine many trees

- Bagging (Breiman 1996): Fit many large trees to bootstrap resampled versions of the training data, and classify by majority vote.
- Random forests (Breiman 1999): Improvements over bagging with randomized trees.
- Boosting (Freund & Schapire 1996): Fit many large or small trees to reweighed versions of the training data. Classify by weighted majority vote.
- Generally, bagging tends to dominate a single tree, boosting tends to dominate bagging, and random forests and boosting are often comparable.

# Bootstrap Samples

- Training data $(x_1, y_1), \ldots, (x_n, y_n)$.
- Randomly draw one pair; make a copy $(x_1^*, y_1^*)$, and put it back (sample with replacement).
- Repeat $n$ times, obtaining $(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)$. This is called a bootstrap sample.
- Repeat everything $B$ times, obtaining $B$ bootstrap samples, of size $n$ each.

# Bagging

Bagging (Bootstrap AGGregatING averages a given procedure over many bootstrap samples, to reduce its variance.

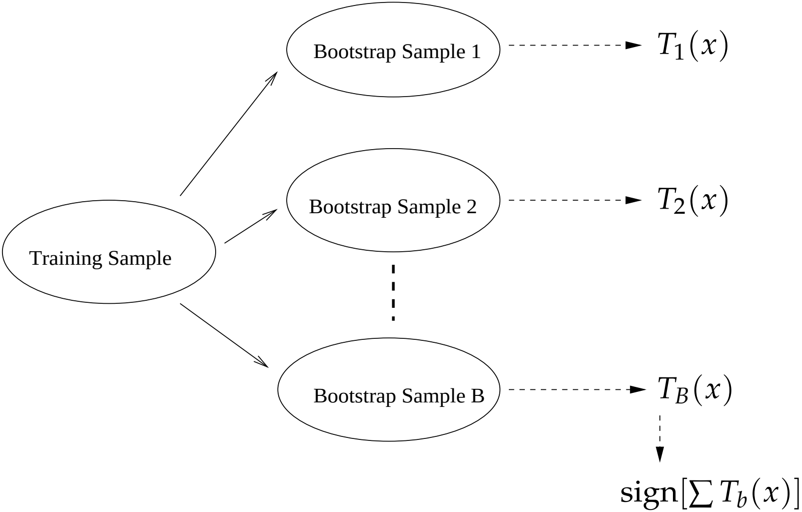
- If $T(x)$ is a classifier (e.g. a classificate tree), we get a version $T^{(b)}$ by training on each of the bootstrap samples $b = 1, \ldots, B$. Then bagging predicts

$$\hat{C}_{\text{bag}}(x) = \text{Majority Vote}\left\{T^{(b)}(x)\right\}_{b=1}^{B}$$

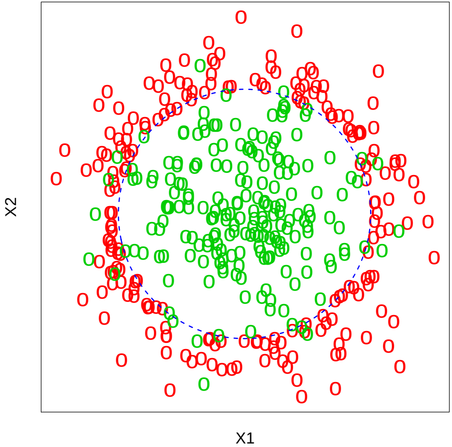
- If $T(x)$ is a regression method (e.g. a regression tree), we also get a version $T^{(b)}$ by training on each of the bootstrap samples $b = 1, \ldots, B$, and bagging predicts

$$\hat{f}_{\text{bag}}(x) = \text{Average}\left\{T^{(b)}(x)\right\}_{b=1}^{B}$$

# Schematics of Bagging

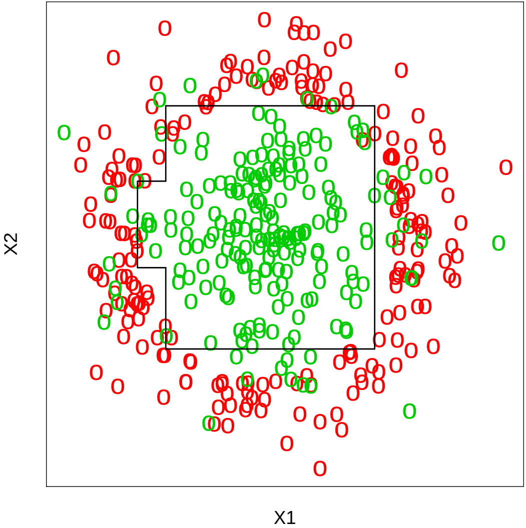


# Recall Nested Spheres

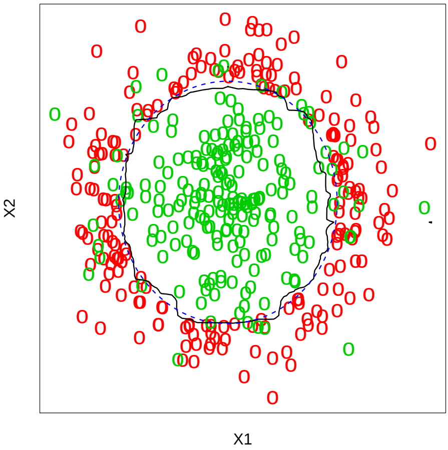



- Green class: two independent standard normal inputs $X_1$, $X_2$
- Red class: conditioned on $X_1^2 + X_2^2 \geq 4.6$

# A Single Tree Decision Boundary



# Bagging Decision Boundary:



# Remarks on bagging

- Bagging can dramatically reduce variance, and does not affect bias.
- Bagging produces very flexible decision boundaries, unlike a tree.
- Bagging a good method (e.g., a decision tree) usually makes it better, but bagging a bad method (e.g., a random decision rule) does not, and can make it worse.
- Interpretable structure (e.g. splits of a tree) is lost.

# Out-of-Bag Error Estimation

- Since bootstrap randomly selects a subset of observations to serve as training data, the remaining part can serve as test data.
- On average, each bagged tree makes use of around 2/3 of the observations, which leaves about 1/3 for testing.
- Test error can be estimated by averaging "out-of-bag" (OOB) errors over all $B$ samples.

# Variable Importance

- Bagging typically improves the accuracy over prediction using a single tree, but at the expense of interpretability.
- We now have hundreds of trees, and it is no longer clear which variables are important.
- Alternative: overall summary of the importance of each predictor using relative variable importance, aka as relative influence.

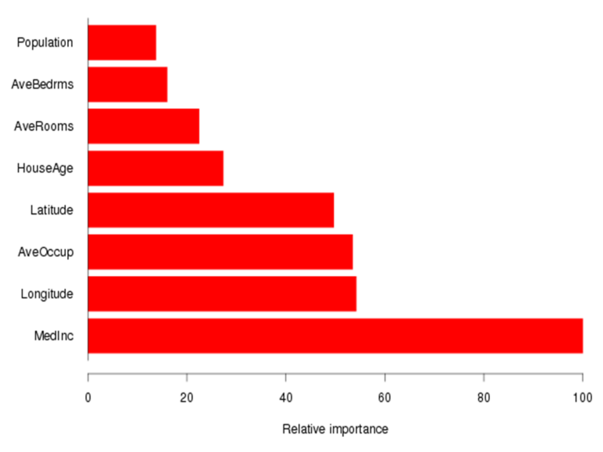
# Relative Importance Plots

How do we decide which variables are most useful in predicting the response?

- Display a score for each variable
- These scores represent the total decrease in the Gini index or RSS over all splits on a given variable, averaged over $B$ bootstrapped trees.
- The larger the score, the more important the variable is overall; the most important is scaled to 100%.

# Example: Housing Data

- "Median income" is by far the most important variable.
- "Longitude", "latitude" and "average occupancy" are the next most important.



# Random forests: main idea

- Create $B$ bootstrapped training sets as in bagging
- Grow a decision tree on each bootstrapped training data set, but consider a random subset of variables at each split. The trees are not pruned.
- Use majority vote among all the trees to classify a new object.

# Why only a subset of predictors?

- Random forests makes the trees less correlated
- Suppose we have one very strong predictor in the data set; then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split.
- All bagged trees will look similar, and predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not lead to a large variance reduction, and thus random forests “de-correlates” the bagged trees leading to more reduction in variance.

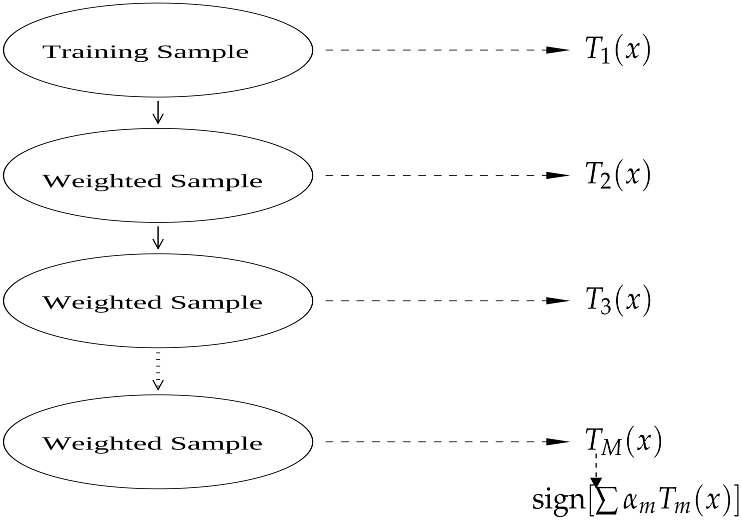
# Remarks on random forests

- Random forests are considered one of the most competitive classifiers and are popular
- Random selection of variables controls overfitting
- For most data sets results seem not too sensitive to $m$, the number of variables used for splitting
- While there is no interpretation, variables can be ranked by importance
- However, importance rankings can be much more variable that the classification results themselves

# Boosting: main idea

- Give sample points weights (initially all equal) that measure how much the point affects the classifier
- Fit a classifier and adjust weights to give more weight to the points that were misclassified
- Repeat $M$ times
- Take a weighted vote of the resulting $M$ classifiers, with more weight given to better classifiers

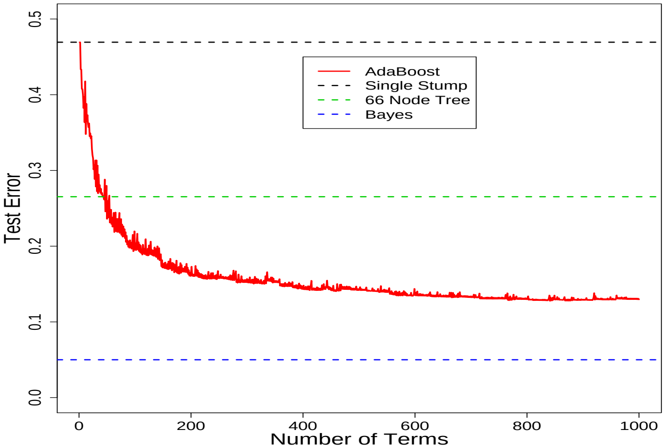
# Schematics of Boosting



# AdaBoost

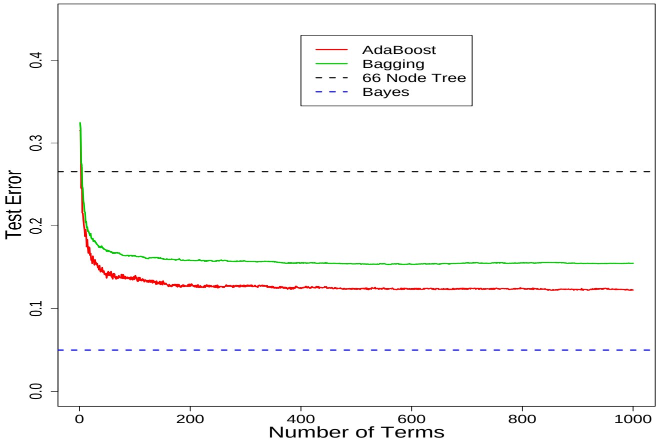
1. Initialize the observation weights $w_i = 1/n, i = 1, 2, \ldots, n$.
2. For $m = 1$ to $M$ repeat steps (a)-(d):
   (a) Fit a classifier $T_m(x)$ to the training data using weights $w_i$.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq T_m(x_i))}{\sum_{i=1}^{n} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Update weights for $i = 1, \ldots, n$:
   $$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot \mathbb{I}(y_i \neq T_m(x_i))]$$
   and renormalize $w_i$ to sum to $1$.
3. Output $\hat{C}(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m T_m(x)\right]$.

# Boosting Stumps




2000 points from nested spheres in $\mathbb{R}^{10}$. A stump is a two-leaf tree, after a single split. Boosting stumps works remarkably well on the nested-sphere problem.

# Bagging and Boosting




Boosting ten-node trees and Bagging.

# AdaBoost: Stagewise Modeling

- AdaBoost builds an additive model

$$f(x) = \sum_{m=1}^{M} \alpha_m T_m(x)$$

by stagewise fitting using the loss function

$$L(y, f(x)) = \exp(-yf(x)).$$

- Goal:

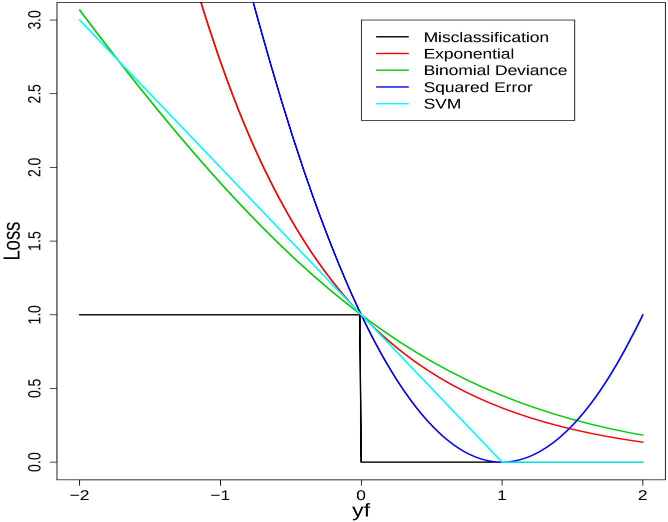
$$\boxed{\min_f \sum_{i=1}^{n} \exp(-y_i f(x_i))}$$

- After $m-1$ steps, suppose we have the model $f_{m-1}(x)$.
- At the $m$th step we solve

$$\min_{\alpha,T} \sum_{i=1}^{n} \exp\left[-y_i(f_{m-1}(x_i) + \alpha T(x_i))\right]$$

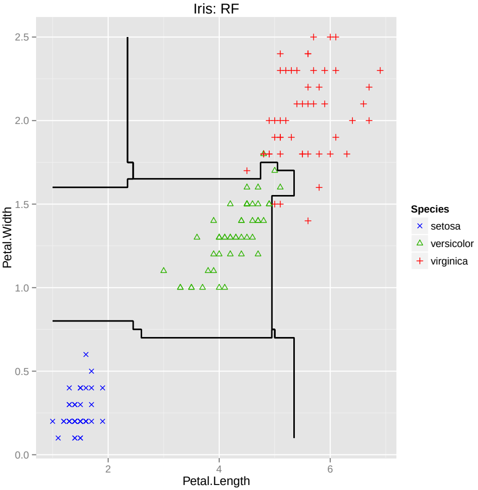
  where $T(x) \in \{-1, 1\}$ is a tree classifier and $\alpha$ is a coefficient.
- Thus the solution $\alpha_m T_m(x)$ is added to $f_{m-1}(x)$ to get $f_m(x)$.
- We can show that this leads to the AdaBoost algorithm.

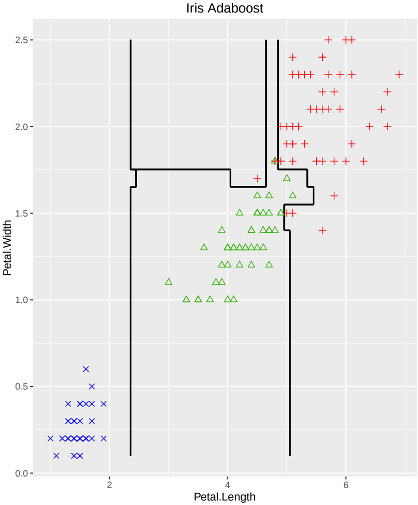
# Why Exponential Loss?



- $e^{-yf(x)}$ is a monotone, smooth upper bound of misclassification loss.
- Leads to a simple reweighting scheme.
- Other loss functions can be used too

# Iris data example



# General Stagewise Algorithm

We can do the same for general loss functions, not only exponential loss.

1. Initialize $f_0(x)$.
2. For $m = 1$ to $M$:
   (a) Compute
   $$(\alpha_m, T_m) = \arg\min_{\alpha, T} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \alpha T(x_i))$$
   (b) Set $f_m(x) = f_{m-1}(x) + \alpha_m T_m(x)$.

# MART

- Multiple Additive Regression Trees (Friedman, 2001)
- 2002 ACM Data Mining Lifetime Innovation Award: Friedman
- General boosting algorithm that works with a variety of different loss functions.
- MART inherits the good features of trees (variable selection, missing data, mixed predictors), and improves on the weak features, such as prediction performance.

# Remarks on boosting

- Boosting works well with trees, but can in principle be applied to any classifier
- Boosting can overfit, so it is important to limit $M$ (the total number of iterations) and the maximum allowed depth of each tree
- Interpretable structure is also lost
- Multi-class extensions are available