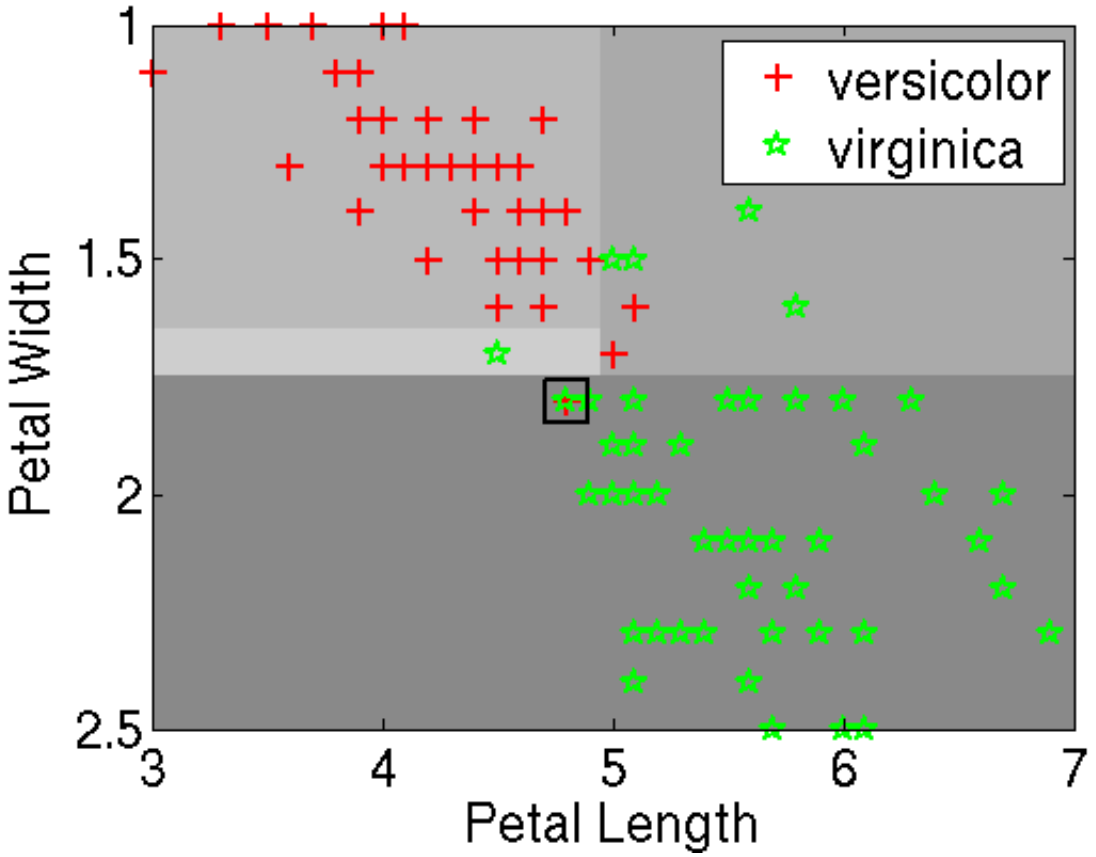
Boosted decision trees and random forest

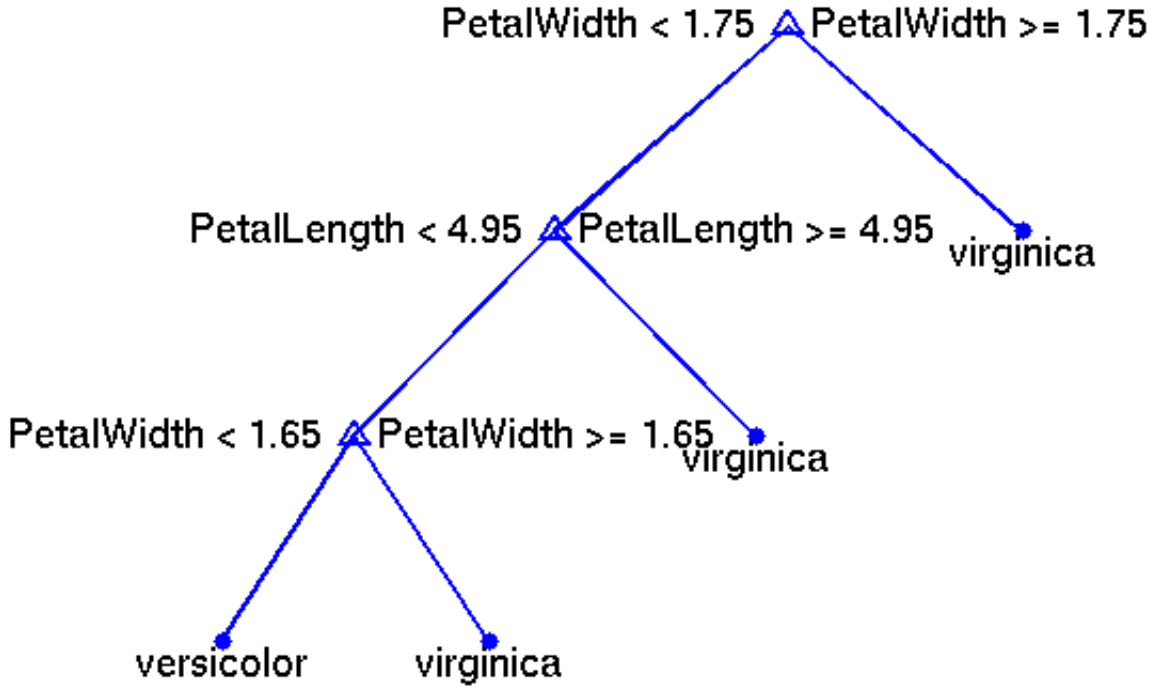
Ilya Narsky

Copyright

Decision trees

World in (D-dimensional) boxes





Incarnations

CART	Breiman, Friedman, Stone & Olshen, Classification and Regression Trees, 1984	`tree` **(R)** `fitctree` **(MATLAB)** **Salford Systems**
C4.5	Quinlan, papers and books in 1986-1993	C50 **(R)** J48 **(WEKA, Java)** C5.0 **(RuleQuest, C)**
CHAID	Kass, 1980	CHAID **(R)** Statistica **(StatSoft)**

Tree for binary classification

Impurity of node t for class fractions p and q:

$$i(t) = \varphi(p, q); \quad p + q = 1$$

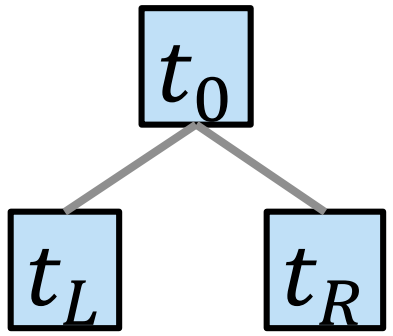
Good node: $i(t) = 0$ Bad node $i(t) = 0.5$

Node impurity weighted by node probability:

$$I(t) = P(t)i(t)$$

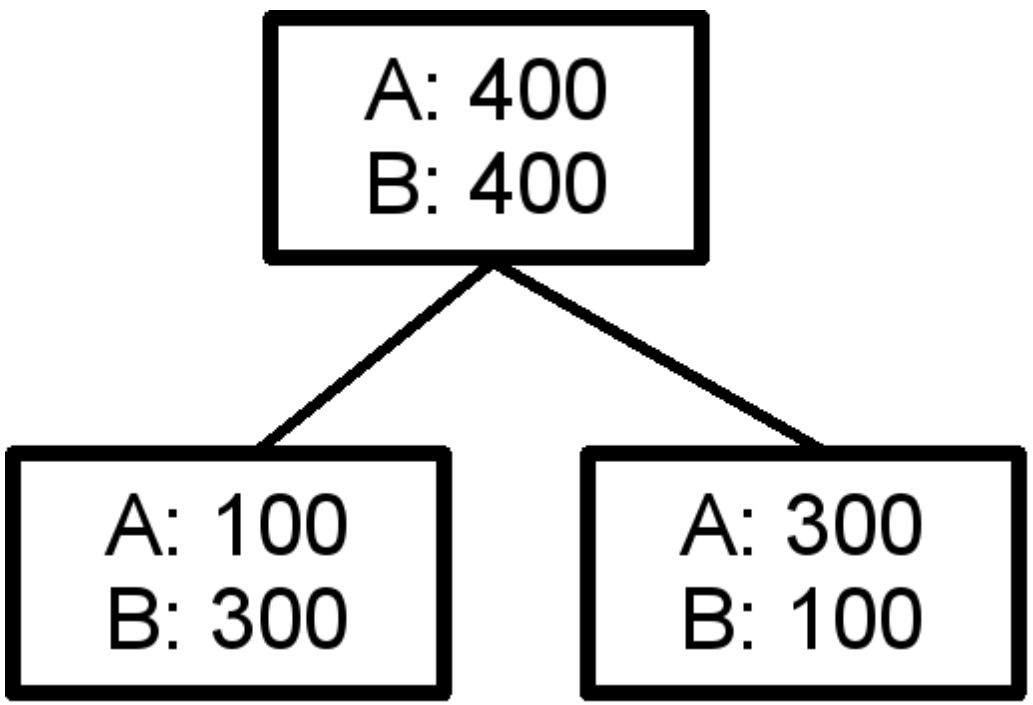
Impurity measures:

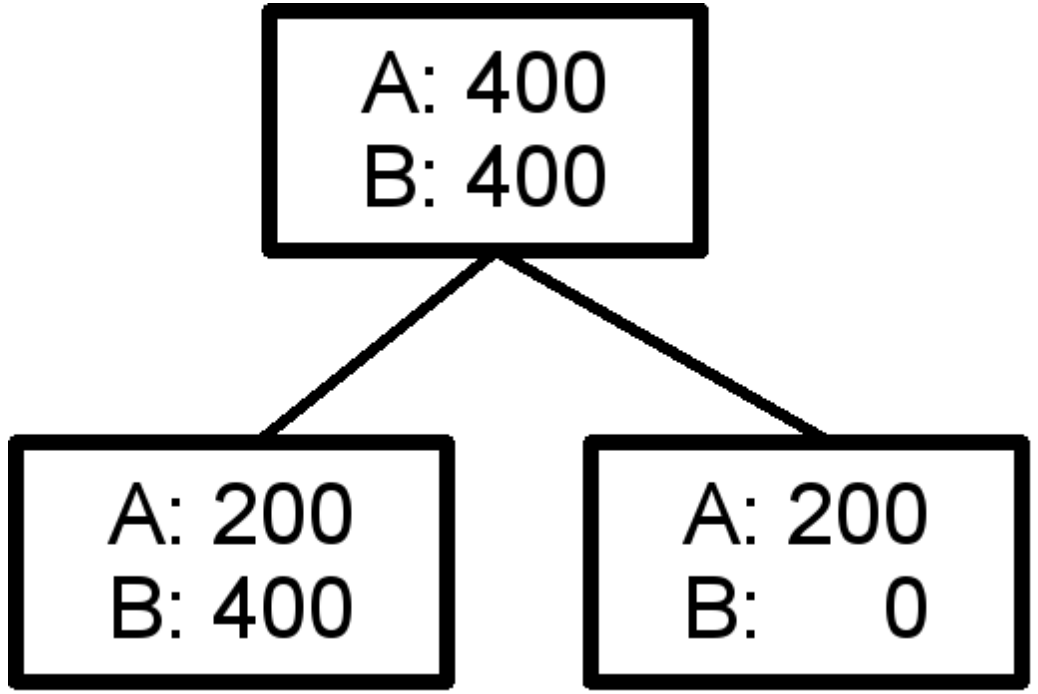
- Classification error $i(t) = \min(p, q)$
- Gini index: $i(t) = 1 - p^2 - q^2$
- Cross-entropy:
  - $i(t) = -\frac{1}{2}(p \log_2 p + q \log_2 q)$



Maximize impurity gain $\Delta I = I(t_0) - I(t_L) - I(t_R)$

Minimizing classification error





- The two splits give equal classification error.
- Yet one of them gives a pure node, while the other one does not.
- It seems natural to prefer big pure nodes.

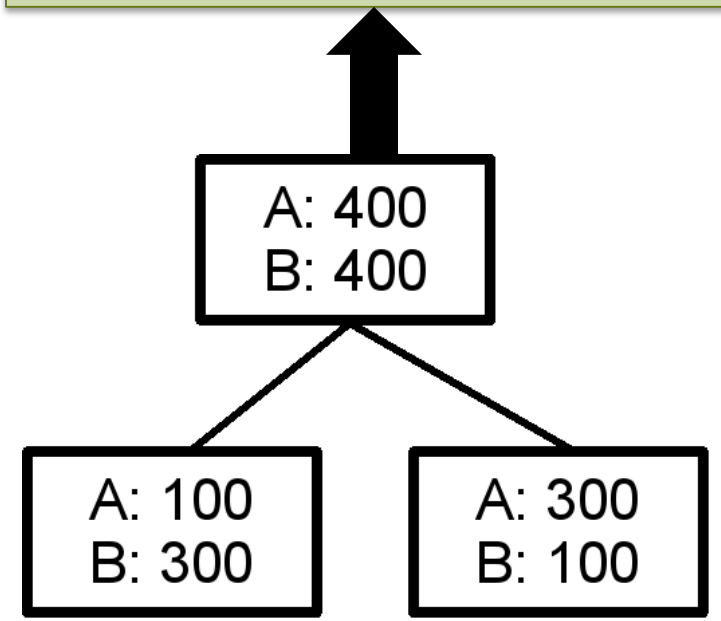
Gini index

Maximize $\Delta I = I(t_0) - I(t_L) - I(t_R)$

$i(t_L) = \frac{3}{8}$ $\quad i(t_R) = \frac{3}{8}$

$P(t_L) = \frac{1}{2}$ $\quad P(t_R) = \frac{1}{2}$

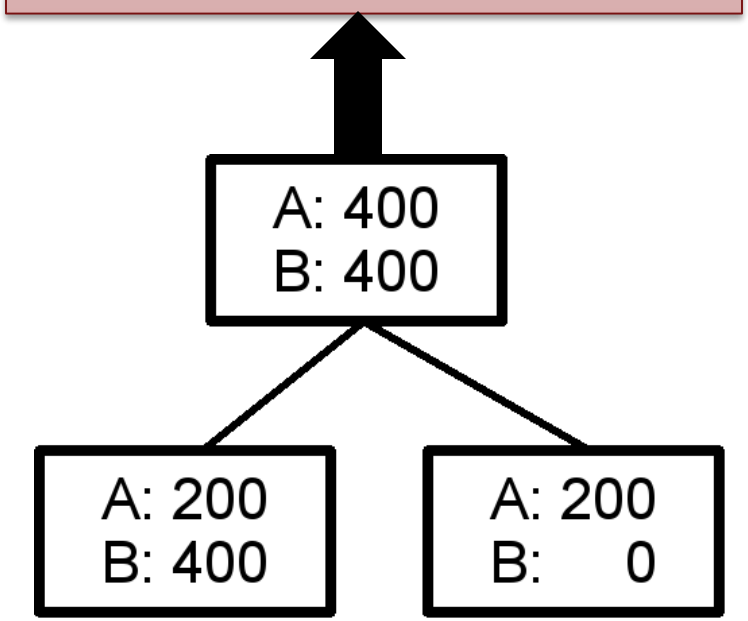
$$I(t_L) + I(t_R) = \frac{3}{8}$$

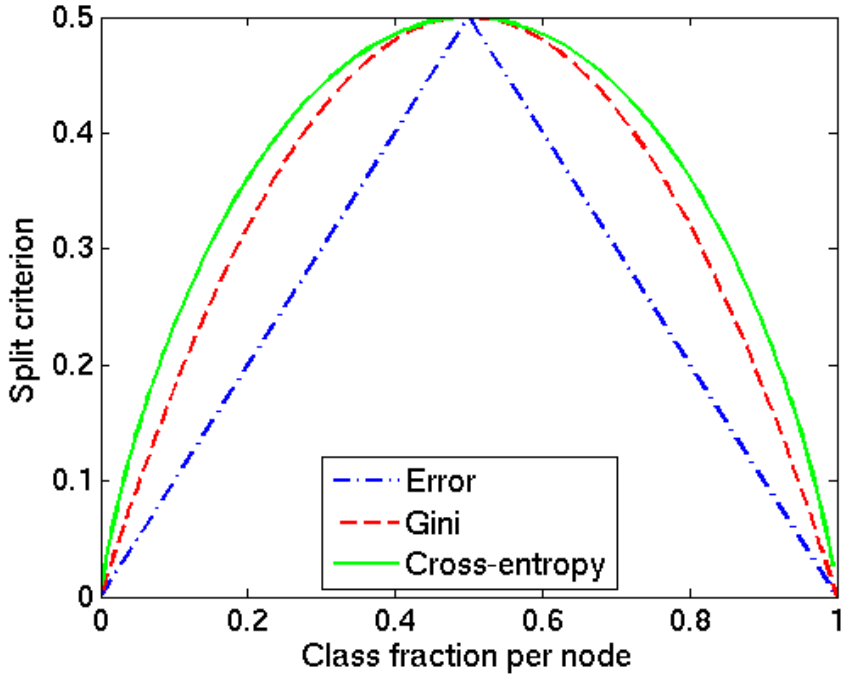


$i(t_L) = \frac{4}{9}$ $\quad i(t_R) = 0$

$P(t_L) = \frac{3}{4}$ $\quad P(t_R) = \frac{1}{4}$

$$I(t_L) + I(t_R) = \frac{1}{3}$$





Nonlinear concavity is essential

Impurity measures for multiple classes

Binary impurity measures:

- Classification error $i(t) = \min(p, q)$
- Gini index: $i(t) = 1 - p^2 - q^2$
- Cross-entropy:
  - $i(t) = -\frac{1}{2}(p \log_2 p + q \log_2 q)$

Generalize for K classes with probabilities p_k

$$i(t) = 1 - \max_k p_k$$

$$i(t) = 1 - \sum_{k=1}^{K} p_k^2$$

$$i(t) = -\frac{1}{2} \sum_{k=1}^{K} p_k \log_2 p_k$$

Balanced splits by twoing

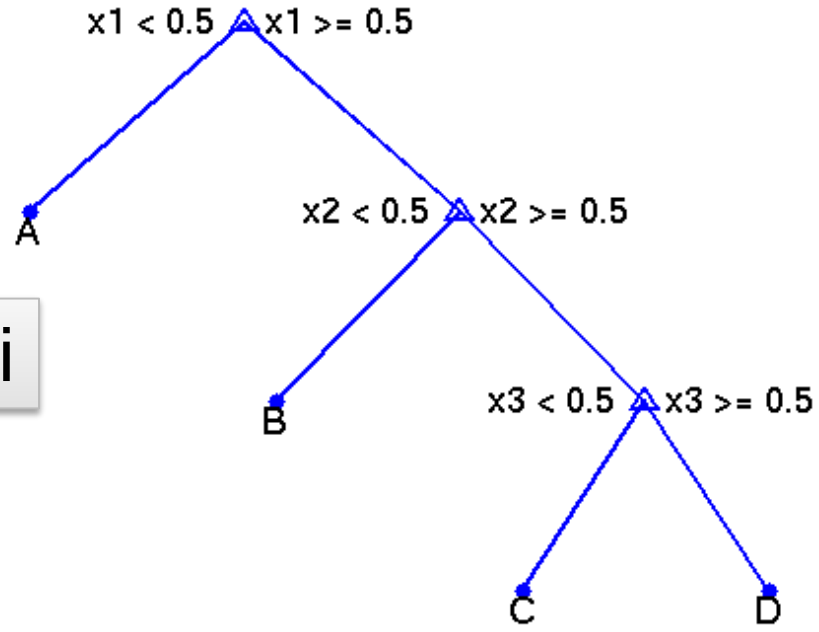
$$\Delta I_{\text{twoing}} = \frac{P(t_L)\,P(t_R)}{4}\left[\sum_{k=1}^{K} |P(k|t_L) - P(k|t_R)|\right]^2$$

How does twoing work?

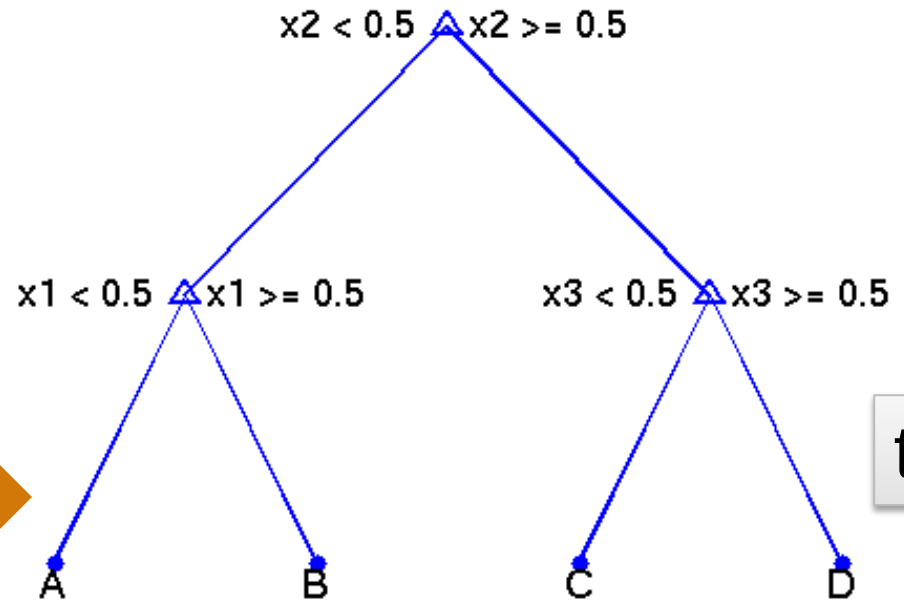
Class	x1	x2	x3
A	0	0	0
B	1	0	0
C	1	1	0
D	1	1	1

Synthetic data with 4 classes and 3 binary variables

Gini



The tree grown by twoing is more balanced



twoing

Splits on numeric variables

x1	x2
1	1
2	2
3	3
1	4
2	5
3	6
1	7
2	8
3	9
1	10

2 possible splits

9 possible splits

- Inspect all possible splits on all variables. Choose the one with the largest impurity gain.
- More splits ⟹ larger probability of finding a spurious pattern.
- Decision tree prefers variables with many possible splits, that is, variables with many possible values.
- A deep tree can find many spurious patterns.

Eliminating bias toward multivalued variables

Loh and Shih, *Split selection methods for classification trees*, Statistica Sinica, 7:815–840, 1997

Contingency tables: Test for independence

- Bin each continuous variable into quartiles ($B = 4$)
- Let $\hat{\pi}_{kb} = \frac{n_{kb}}{N}$ be the estimated probability for class k in bin b
- Let $\hat{\pi}_{k+} = \sum_{b=1}^{B} \frac{n_{kb}}{N}$ and $\hat{\pi}_{+b} = \sum_{k=1}^{K} \frac{n_{kb}}{N}$ be marginal probabilities for classes and bins
- If the variable and class labels are independent, must have $\pi_{kb} = \pi_{k+}\pi_{+b}$
- Let $t = N \sum_{k=1}^{K} \sum_{b=1}^{B} \frac{(\hat{\pi}_{kb} - \hat{\pi}_{k+}\hat{\pi}_{+b})^2}{\hat{\pi}_{k+}\hat{\pi}_{+b}}$ be the test statistic

n_{11}	n_{12}	n_{13}	n_{14}	n_{1+}
n_{21}	n_{22}	n_{23}	n_{24}	n_{2+}
n_{+1}	n_{+2}	n_{+3}	n_{+4}	N

$$t \xrightarrow[N\to\infty]{} \chi^2_{(K-1)(B-1)}$$

under hypothesis of independence

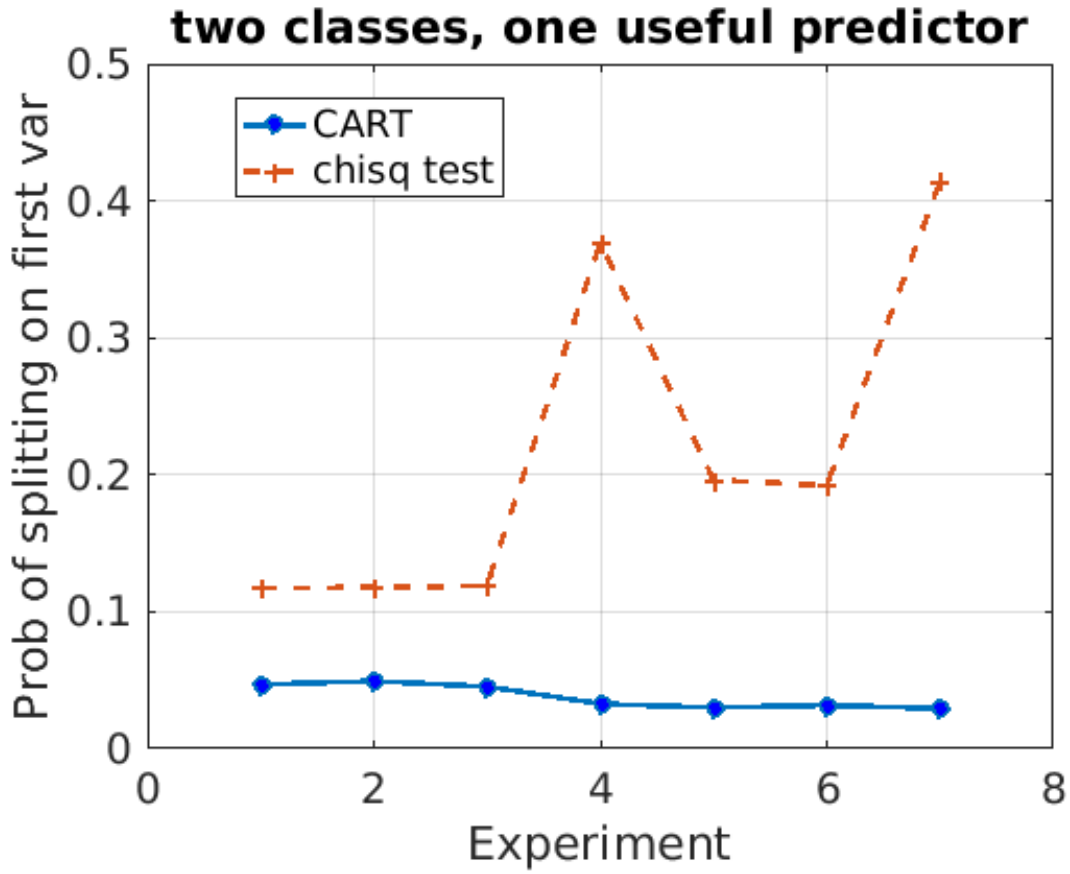
Eliminating bias toward multivalued variables: Recipe

- For each variable, compute the t statistic under hypothesis of independence between this variable and class labels
- Choose the variable with largest value of the statistic
- Find the optimal split on this variable in the usual way (by maximizing impurity gain)

Variable selection is not affected by how many splits can be imposed. Bias toward variables with many splits is eliminated!

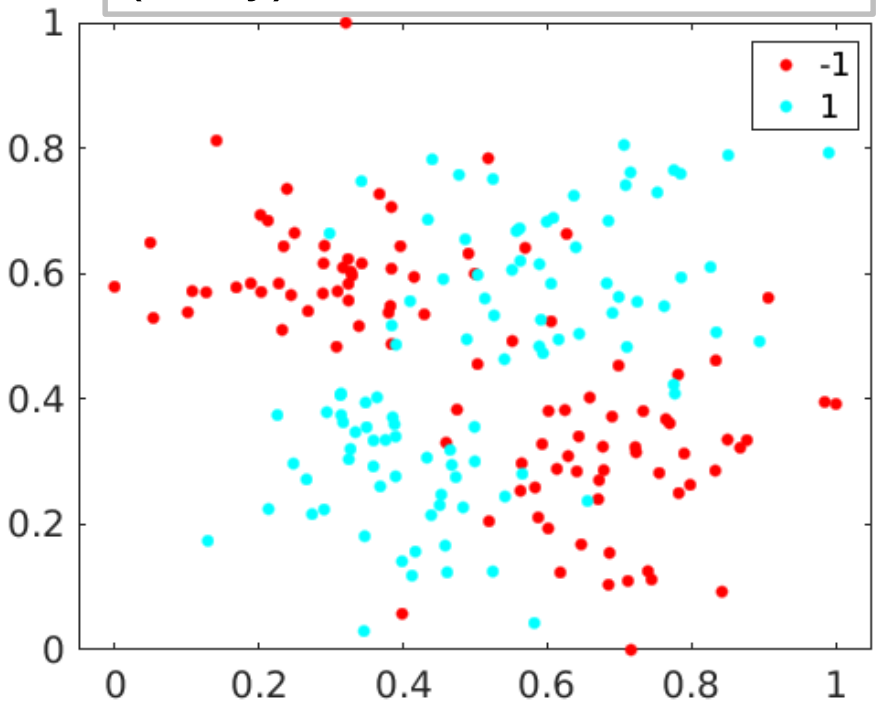
Eliminating bias toward multivalued variables: Synthetic example from Loh & Shih

- Binary classification
- One useful variable (with few values) and 19 useless variables (with many values)
- Limit the number of splits per tree to one (decision stump)
- Measure probability of splitting on the first variable
- Ideally, should be one



Accounting for variable interactions

Yang dataset: Two interacting variables (shown) mixed with (many) useless variables

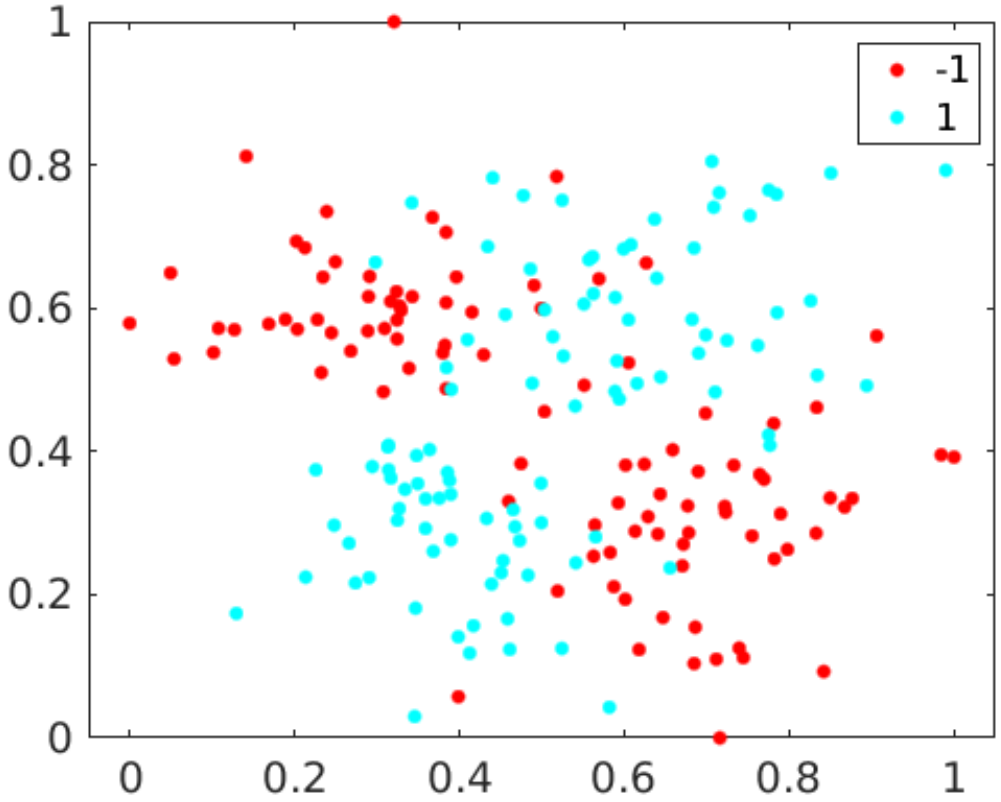


For D variables, consider $D(D-1)/2$ interactions.

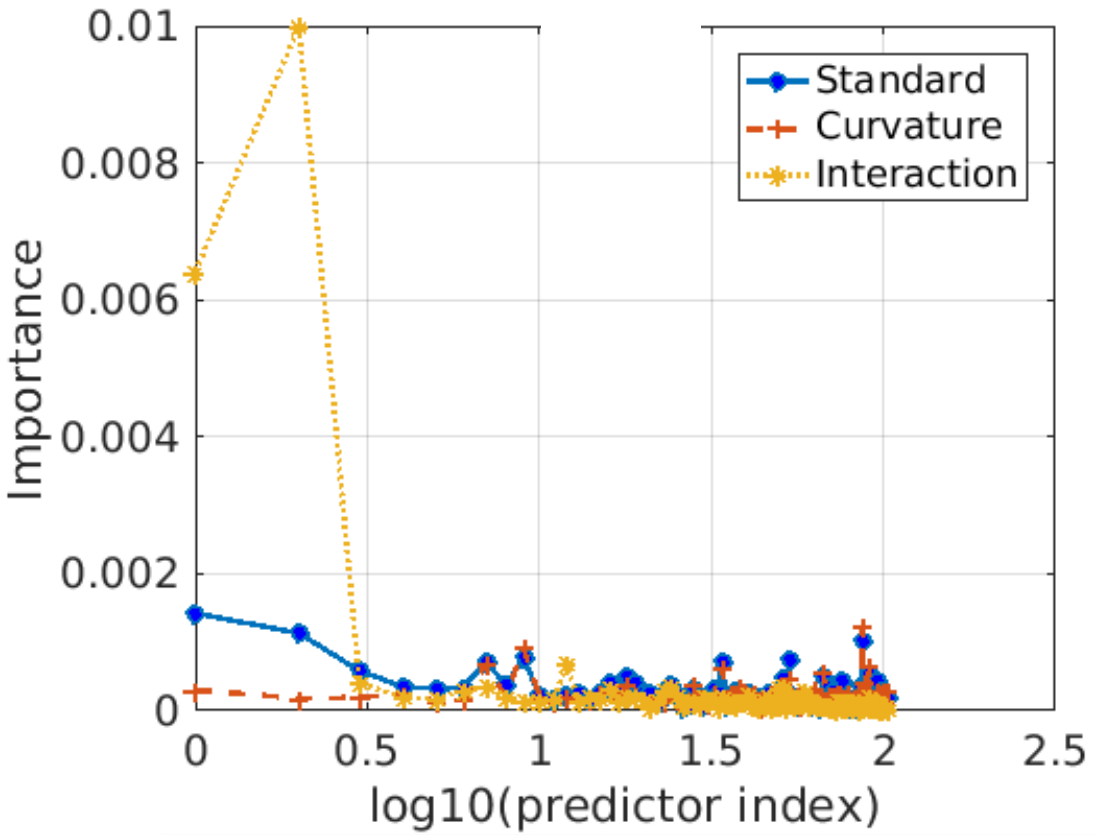
$$t = N \sum_{k=1}^{K} \sum_{b=1}^{B} \frac{(\hat{\pi}_{kb} - \hat{\pi}_{k+}\hat{\pi}_{+b})^2}{\hat{\pi}_{k+}\hat{\pi}_{+b}}$$

- Same test statistic applied to a pair of variables
- Bins are now formed by dividing a 2D plane into quadrants (or smaller pieces)
- Inspect all pairs of variables and choose the one with largest value of the statistic
- Split on one of the two variables in that pair. (The split is still one-dimensional.)

Decision tree performance on this dataset



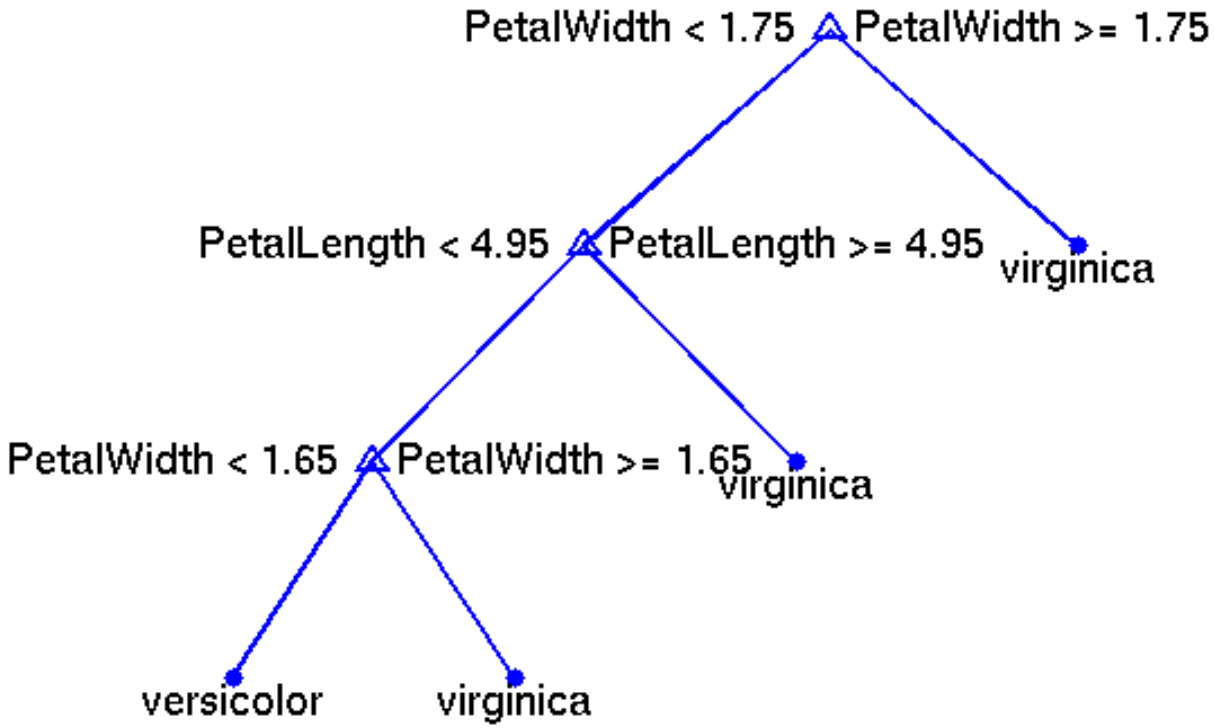
Two strongly relevant variables (shown) diluted with 100 irrelevant variables (not shown).



Interaction test is essential for detecting relevant variables.

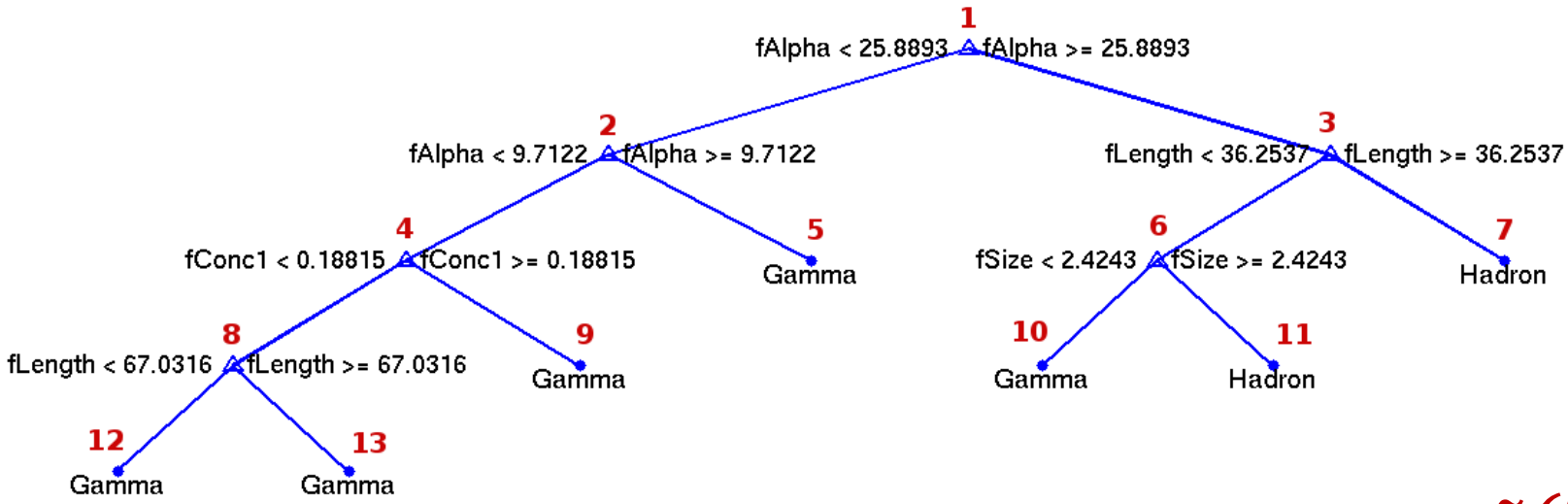
Stopping criteria

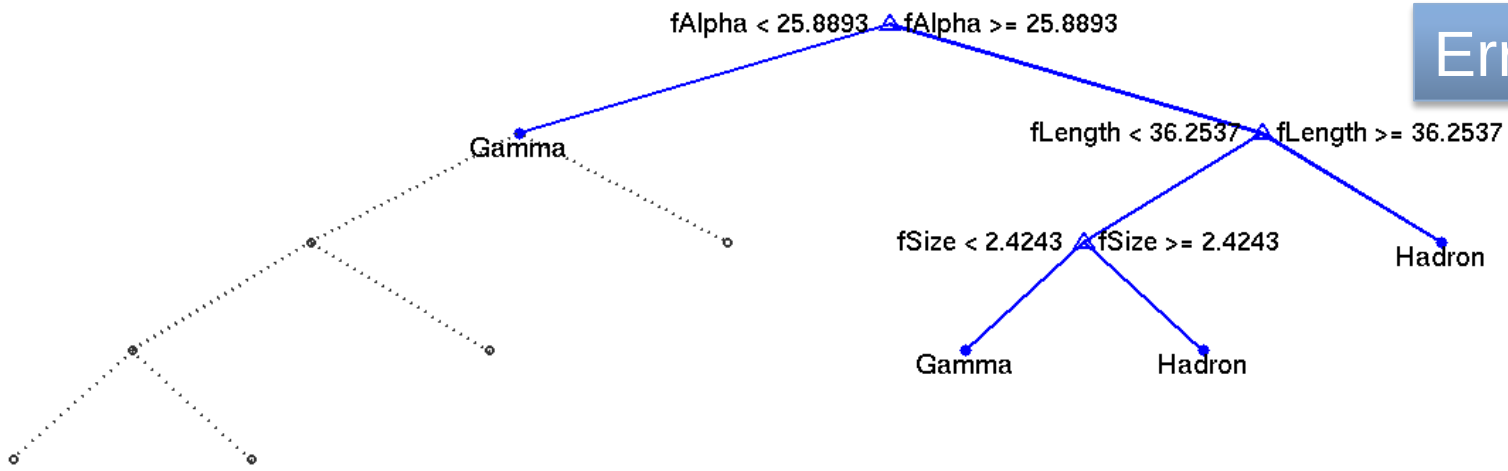
- Pure node (one class only)
- Minimal size of a branch node
  - Size = number of observations
- Minimal size of a leaf node
- Maximal number of splits (branch nodes)
- Minimal impurity gain per split
- Maximal tree depth
- …



Cost-complexity pruning: Alternative to growing many trees

- MAGIC telescope data (UCI ML repository)
- Synthetic dataset
- 19k observations and 10 variables





Penalty coefficient

$$\tilde{r}(T) = r(T) + \alpha|L(T)|$$

Error of tree T

Number of leaves in tree T

Strategies for optimizing tree for accuracy

- **Always estimate accuracy by cross-validation or using an independent validation set**

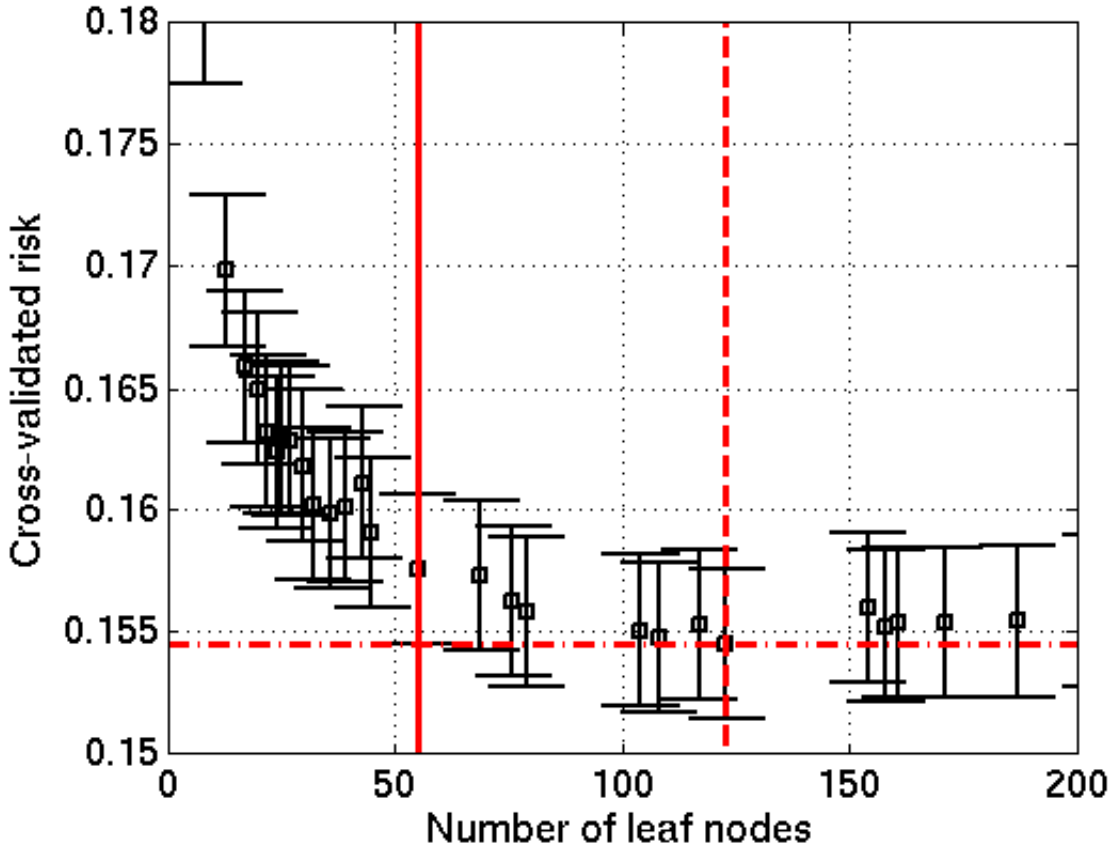
Approach 1

- **Find a good stopping criterion by minimizing error (risk)**

Approach 2

- **Grow a deep tree (as deep as you can) and prune back to minimal validation error (risk). Optimize penalty coefficient α (previous slide).**

- **Instead of pruning back to min validation error, prune back to the level where validation error is one standard deviation away from minimum.**

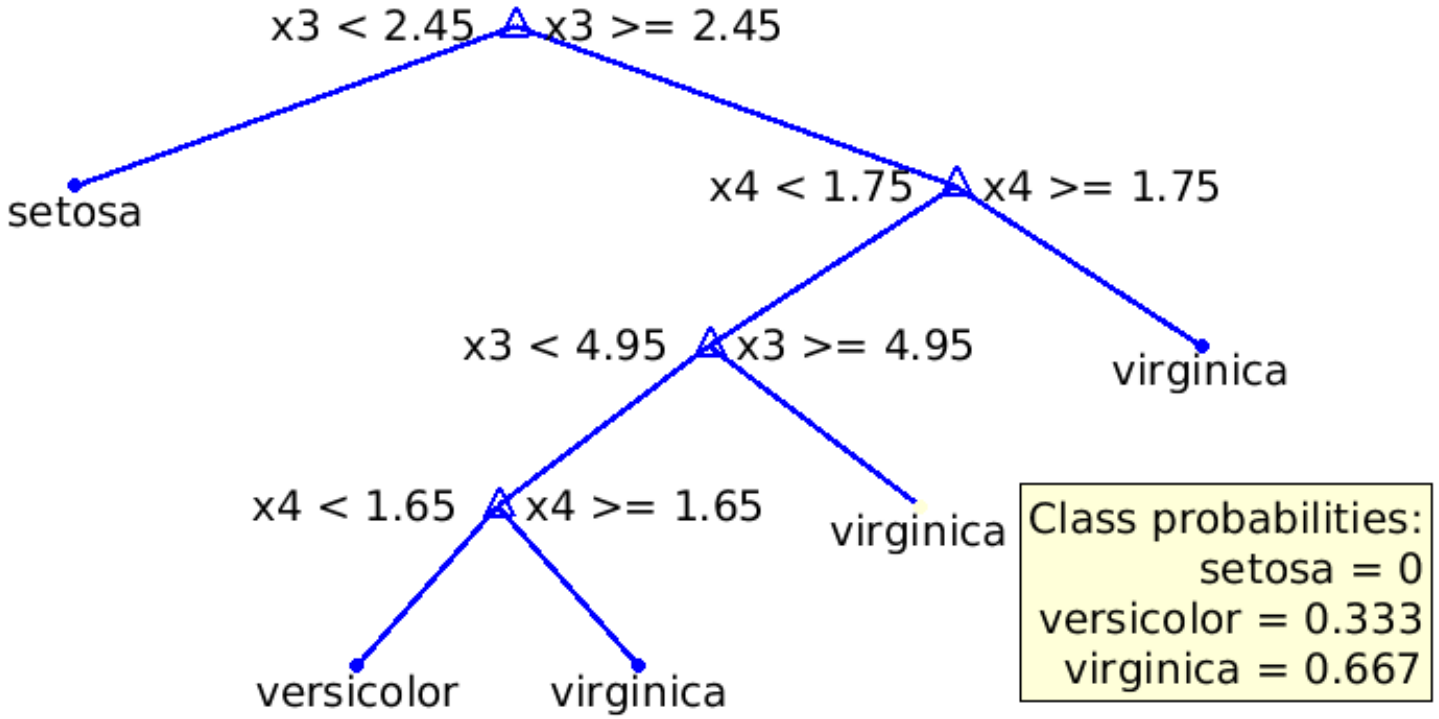


Prediction

$$P(k|t) = \frac{n_k(t)}{n(t)}$$ (fraction of class k in node t)

Predict into the class with the largest posterior

Example: Fisher iris data



Weighted data

$$P(k|t) = \frac{n_k(t)}{n(t)} \quad \Rightarrow \quad P(k|t) = \frac{w_k(t)}{w(t)}$$

- $P(k|t)$ - posterior probability of class k in node t
- $n_k(t)$ - number of observations of class k in node t
- $n(t)$ - total number of observations in node t

- $P(k|t)$ - posterior probability of class k in node t
- $w_k(t)$ - weight of observations of class k in node t
- $w(t)$ - total weight of observations in node t

Generalization of splitting criteria for weighted data easily follow.

Decision trees: Good and bad

- Can easily handle categorical and continuous variables.
- Robust in high dimensions.
- Specialized techniques for missing data support.
- Highly interpretable when shallow.
- Compute posterior probabilities.

- Inaccurate. (Does not refer to ensembles.)
- Unstable: A slight change in the data can change the optimal split and the entire subtree underneath this branch.

Prefer trees when interpretability is favored over accuracy.

Decision tree ensembles

Decision tree features for ensemble learning

Feature	Decision tree	Tree ensemble for accuracy	Tree ensemble for variable selection
Handle categorical and continuous variables	✓	✓	✓
Robust in high dimensions	✓	✓	✓
Efficient missing data support	✓	✓	✓
Interpretability	✓	×	×
Twoing for multiple classes	✓	×	×
Unbiased variable selection	✓	×	✓
Account for variable interactions	✓	×	✓
Pruning	✓	×	×
Posterior probability estimates	✓	●	●

✓ used
× not used
● partially used

Popular types of decision tree ensembles

- Random forest
  - **Training**: Grow many trees on replicas of the data obtained by **unweighted resampling**
  - **Prediction**: Average predictions across trees. (Democratic unweighted vote.)
- Boosted trees
  - **Training**: Grow many trees on **weighted** replicas of the data. Weights are different for every tree.
  - **Prediction**: Take a weighted vote across trees.
- Theory and optimal tree settings for the two types of ensembles are very different

Random forest

Random forest

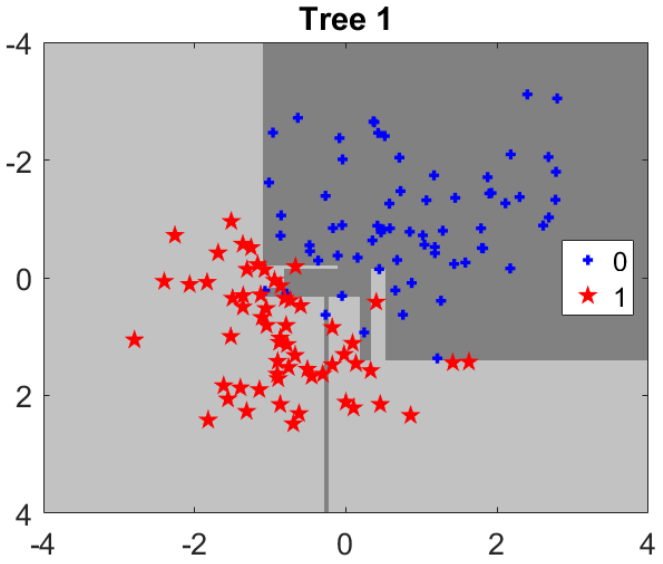
Breiman, L. (2001) Random forests. *Mach. Learn.*, **45**, 5–32.

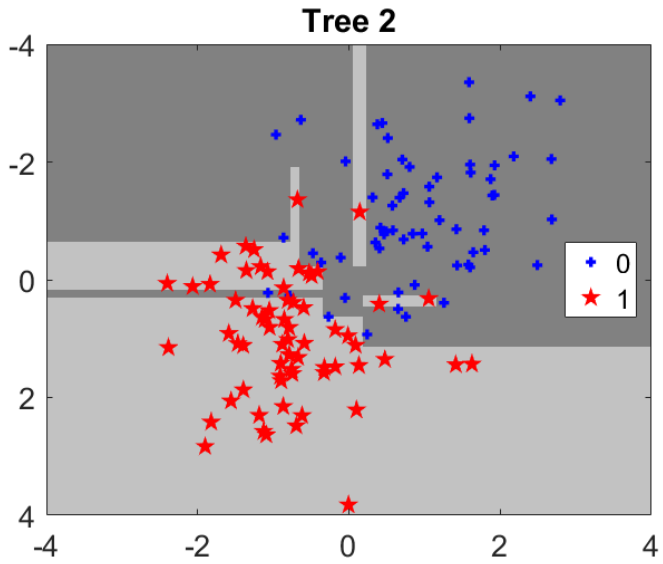
http://www.stat.berkeley.edu/~breiman/RandomForests/

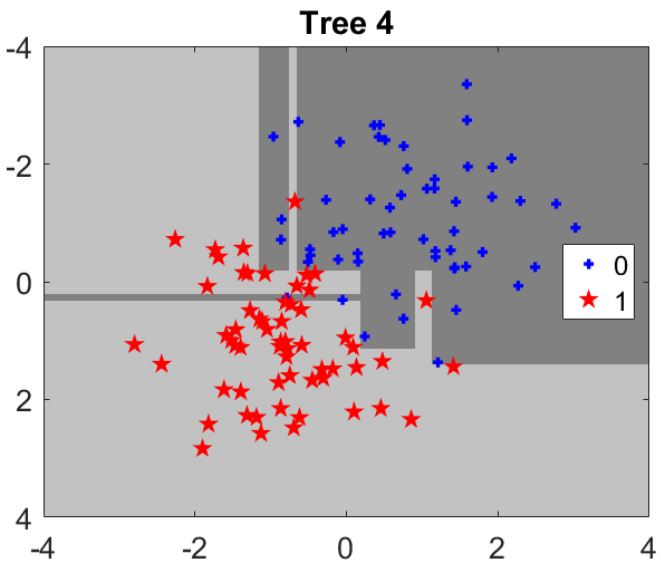
Average over many trees grown on bootstrap replicas of the data

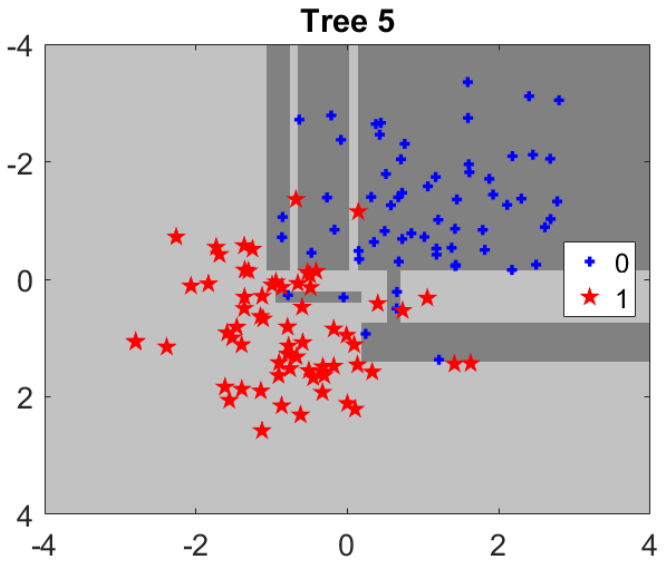
- Bootstrap replica: Sample N observations out of N with replacement
- On average, 63% observations in each replica are unique.

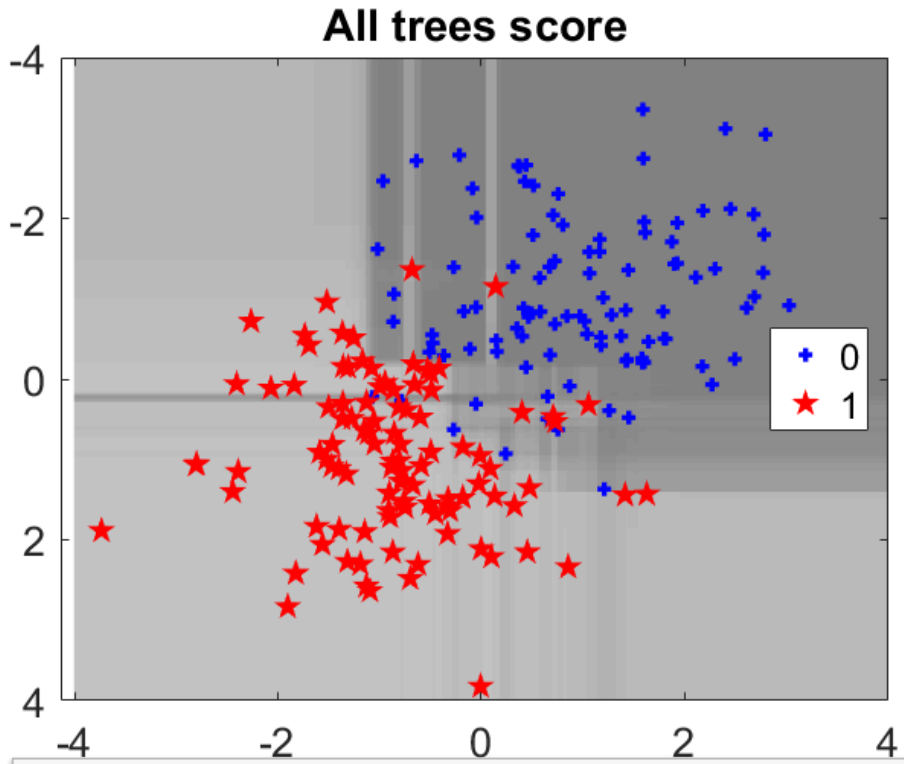
Example: Learning a simple decision boundary







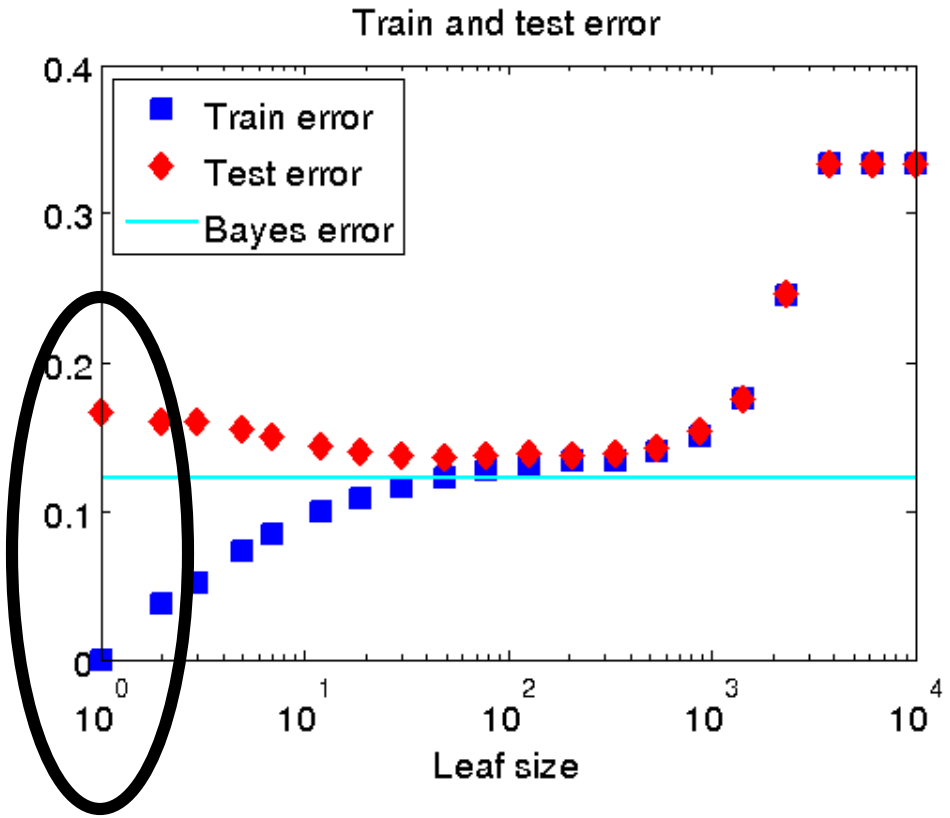




Averaged over a few hundred trees

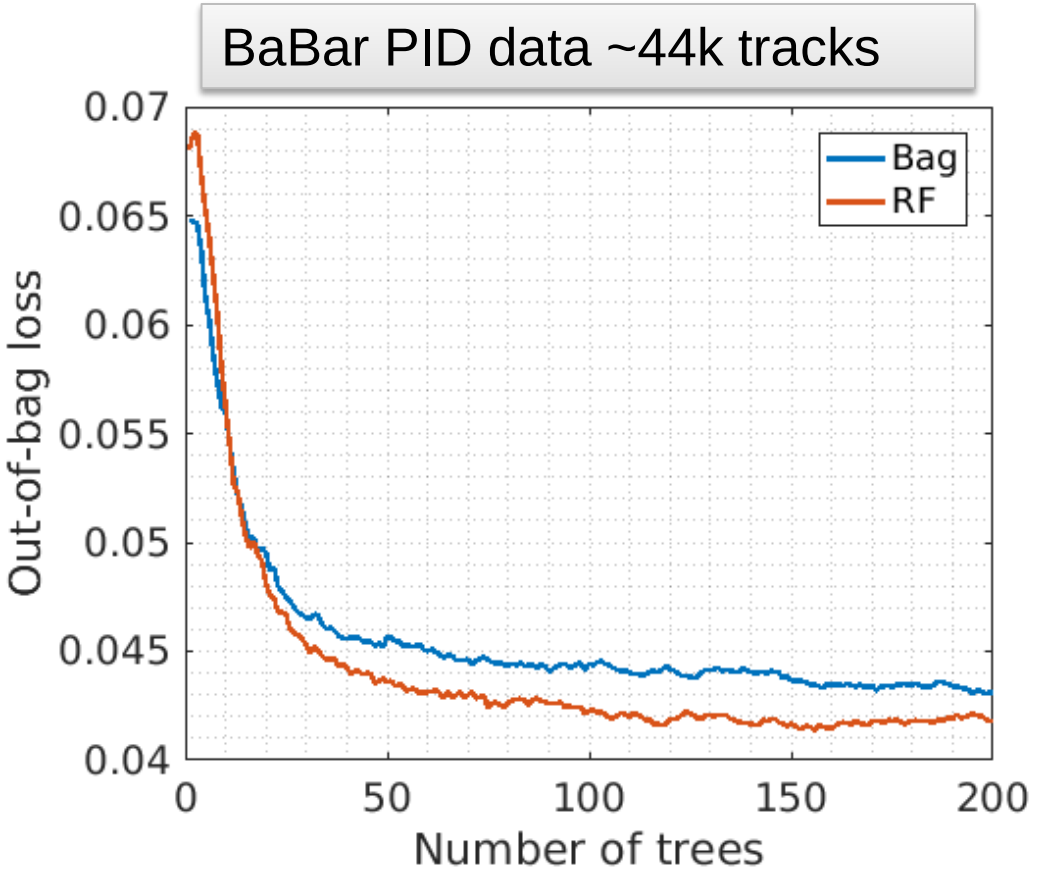
Why does averaging work?

- Grow very deep trees.
  - Minimal leaf size = 1
- Each tree overfits.
- Each tree has low bias and high variance.
- Explore instability of decision tree. Reduce variance by averaging over many trees.



Bagging vs random forest

- Bootstrap aggregation = bagging
- Trees need to be diverse to probe various parts of the input space
- Random forest: Diversify trees by choosing variable candidates for splits at random.
  - At every branch node, choose $d < D$ variables at random.
  - Then apply the usual split selection procedure to these d variables.

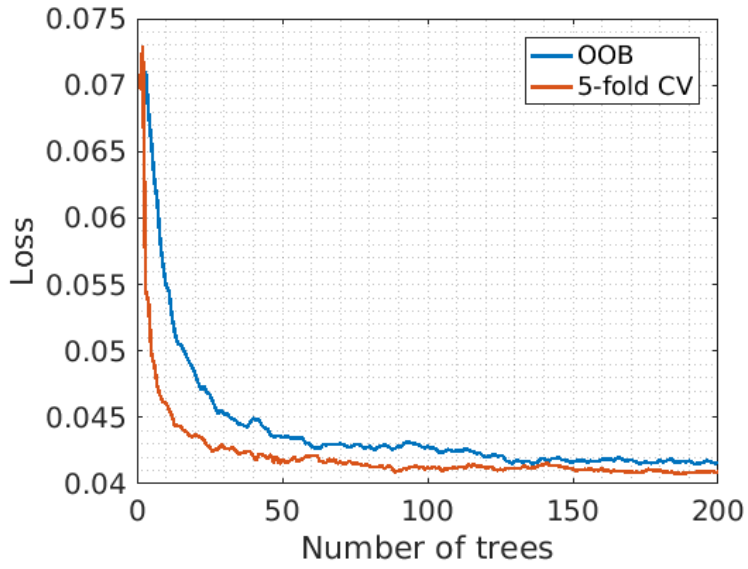


Suggested setting: $d = \sqrt{D}$

Out-of-bag error

- One bootstrap replica gets, on average, 63% of observations in the training set. The remaining 37% are ***out of bag***.
- To predict label for each observation in the training set, use only trees trained on replicas without this observation, that is, trees for which this observation is ***out of bag***.
- Average estimated posterior probabilities across trees or take a majority vote across trees for the class label.
- Compare with true labels to get OOB error.

- Breiman showed (empirically) that OOB error is an unbiased estimate of the generalization error
- Estimating OOB error is typically faster than cross-validating.



Out-of-bag estimates of variable importance

- **For every variable:**
  - **Replace values of this variable with noise**: For every tree, permute values of this variable across OOB observations at random while keeping OOB class labels at the same positions.
  - **How much worse does prediction get due to this variable being replaced with noise?** Record increase in the classification error of this tree due to this permutation.
  - **Average** estimated increases across all trees and normalize by their standard deviation.

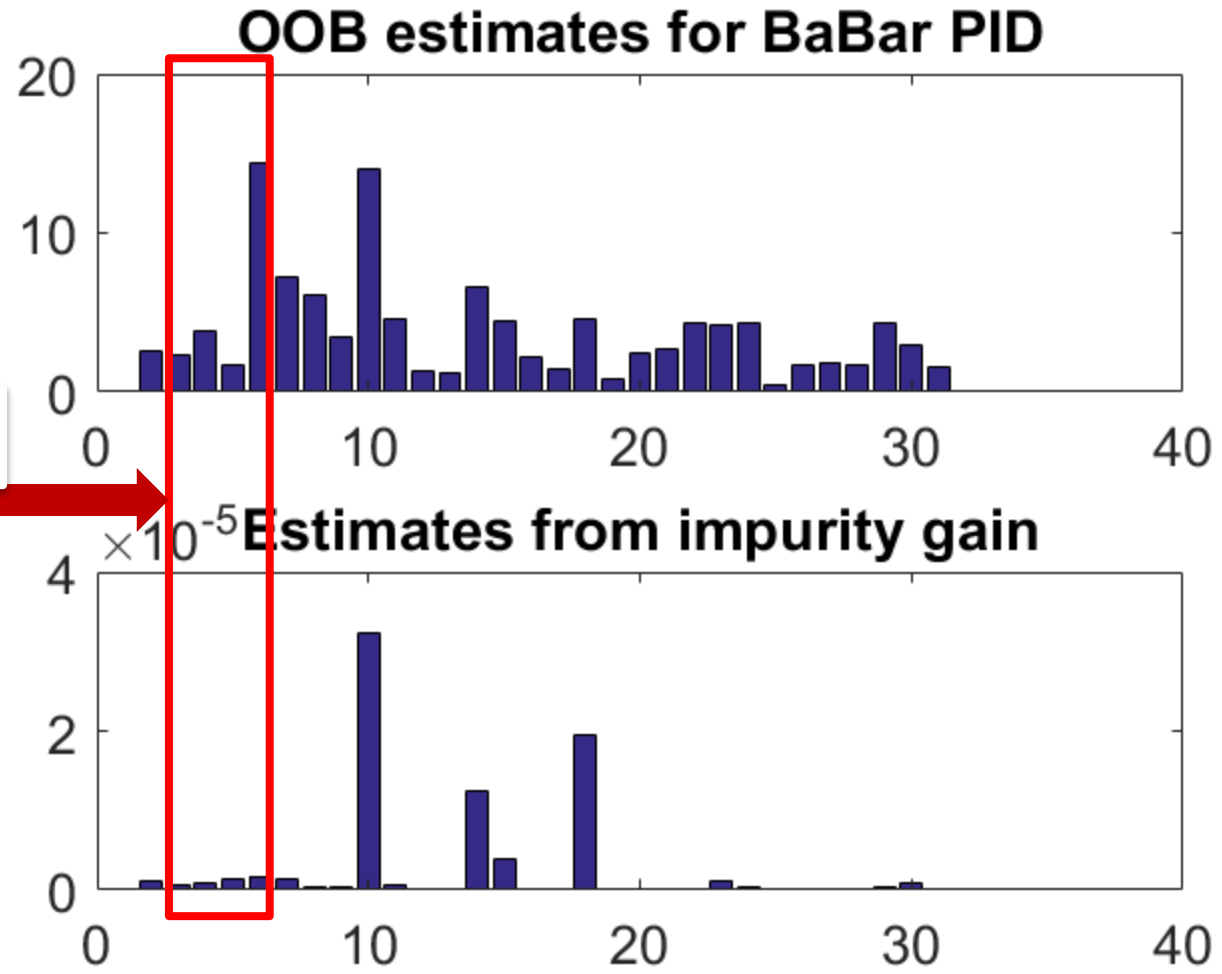
Recommendations:

- Prefer OOB estimates of variable importance over other estimates for decision trees such as counts of splits per variable, average impurity gain per variable etc. **Most accurate!**
- If the primary goal is estimation of variable importance, use all variables for every split (do not subsample).
- Use unbiased selection of split variables (see earlier slides).

Estimates of variable importance: example

Classify K, π, e, p using several detectors

dEdx variables

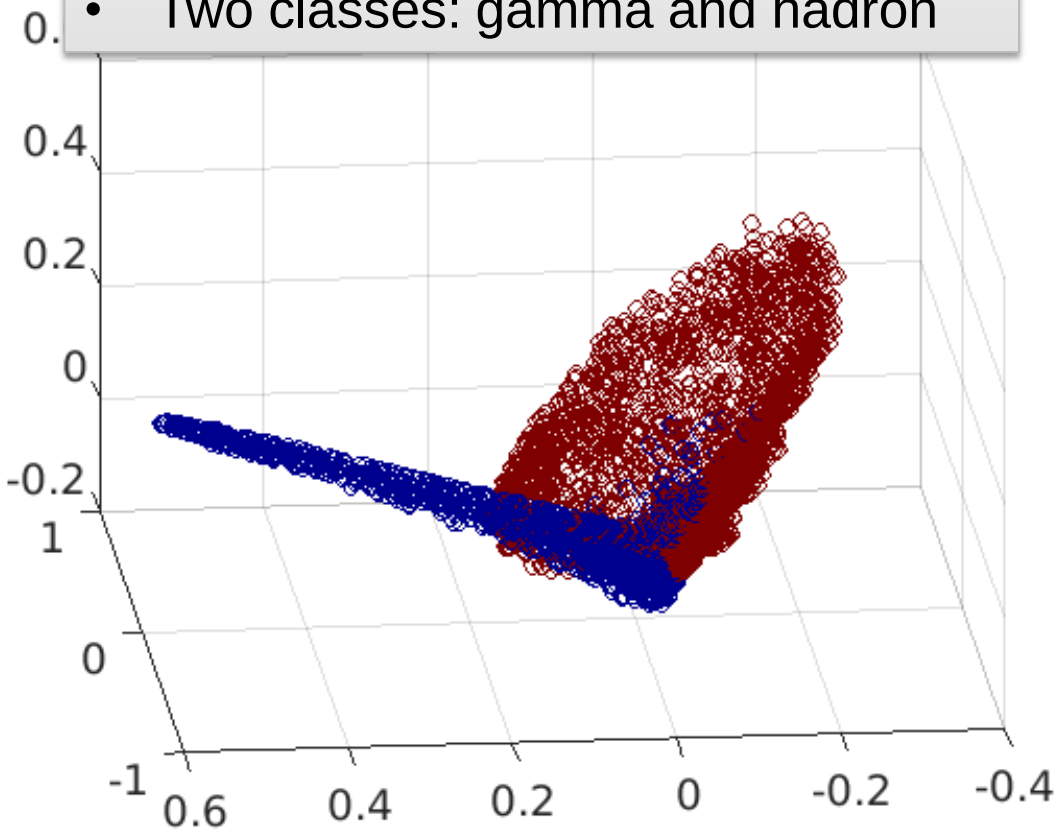


Conclusions can be very different from different techniques!

Proximities for visualization

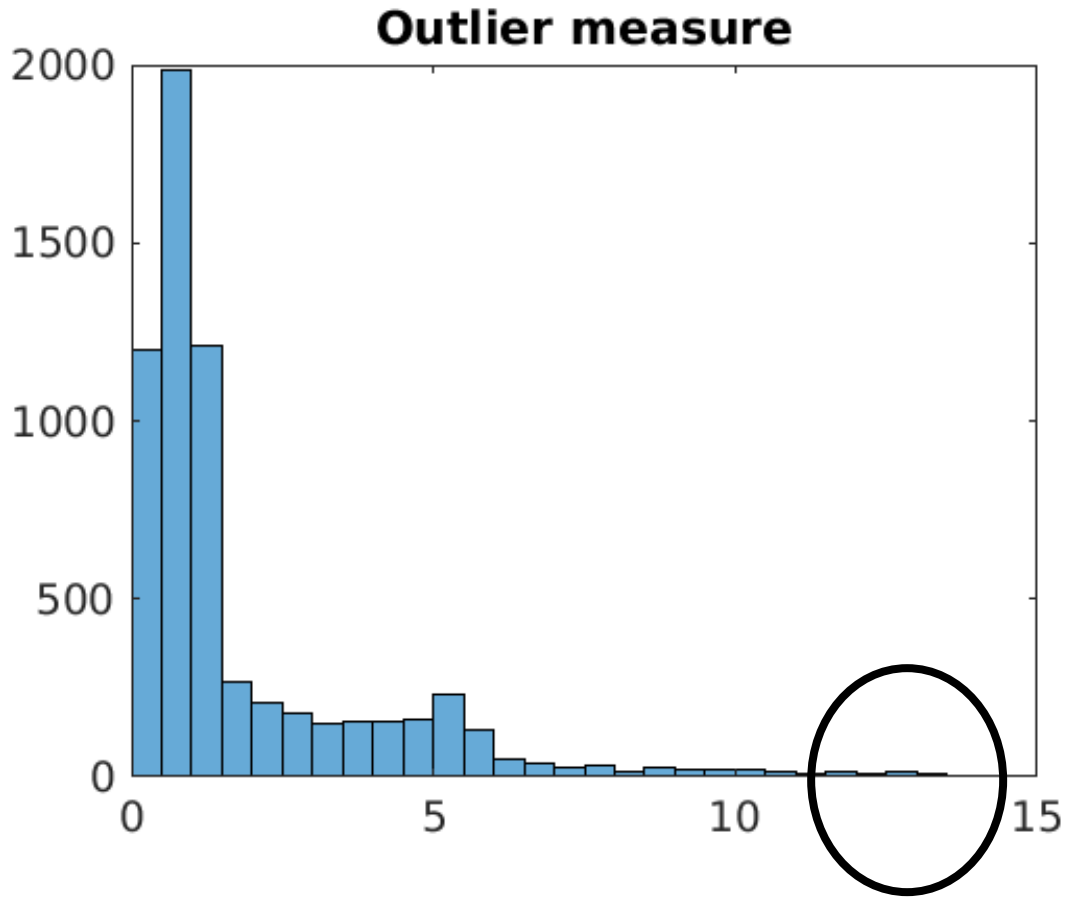
- Compute pairwise proximities $P_{N\times N}$:
  - If two observations land on the same node of the same tree in RF, increase their proximity by 1
  - Otherwise do nothing
  - Normalize $P_{N\times N}$ by the number of trees
- Apply classical multidimensional scaling
- Discover interesting structures in the data!

MAGIC telescope data
- N=6000
- D=10
- Two classes: gamma and hadron



Outliers

- Histogram proximities and identify observations in the tails of the distribution
- Observations with high outlier measures are unlike any other observations in the training set
- These observations may be worth a closer look



Random forest: Practical advice

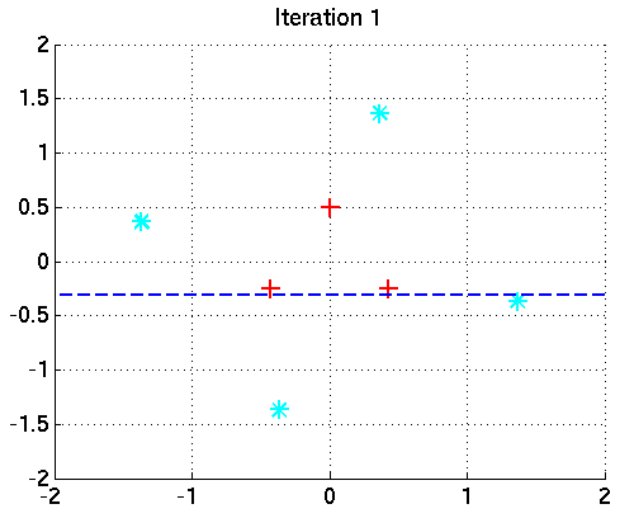
- Two most important tuning knobs are:
  - minimal leaf size
  - number of features to select at random for split candidates
- For classification, Breiman recommends
  - min leaf size = 1
  - number of variables to select at random = $\sqrt{D}$
- These settings are usually pretty good for accuracy on tall datasets $N \gg D$
  - Invest in their optimization only if you want to reduce the memory footprint or squeeze the last drop of accuracy
- 100-200 trees should suffice for most problems
- Good tool for variable selection!
  - If using for variable selection, use all variables for every split

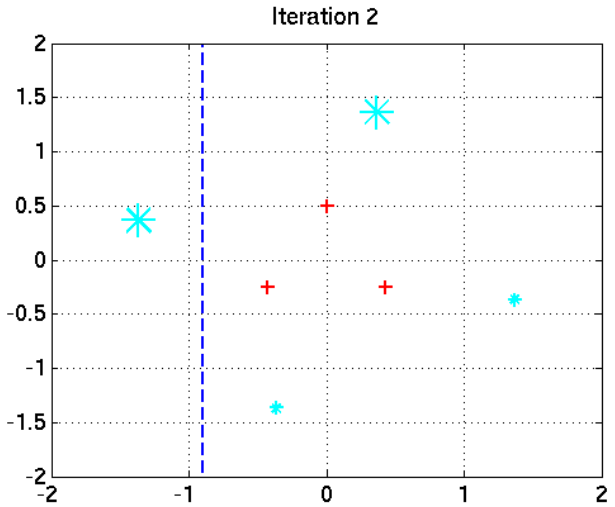
Boosting

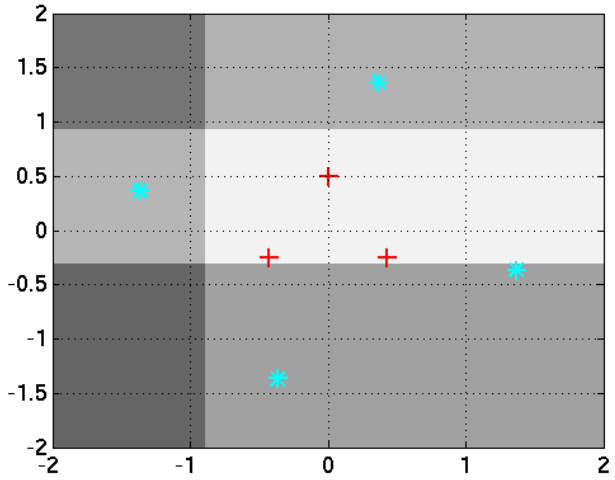
Boosting: Intuitive explanation

- Grow a shallow decision tree (for example, decision stump)
- Find observations in the training set misclassified by the tree
- Increase weights of misclassified observations
- Grow another shallow decision tree on the reweighted set
- Continue for a certain number of iterations, typically a few hundred
- Compute prediction of the ensemble by combining (possibly weighted) predictions from individual trees
  - If weighted, the weight of each tree is determined by its accuracy

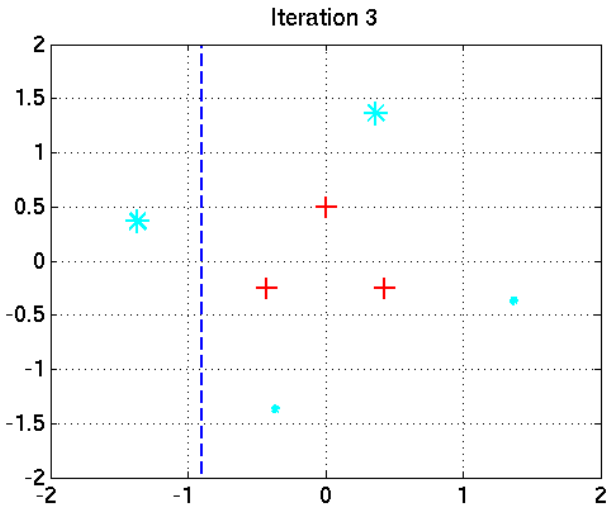
Example using decision stumps (stump = tree with one split)

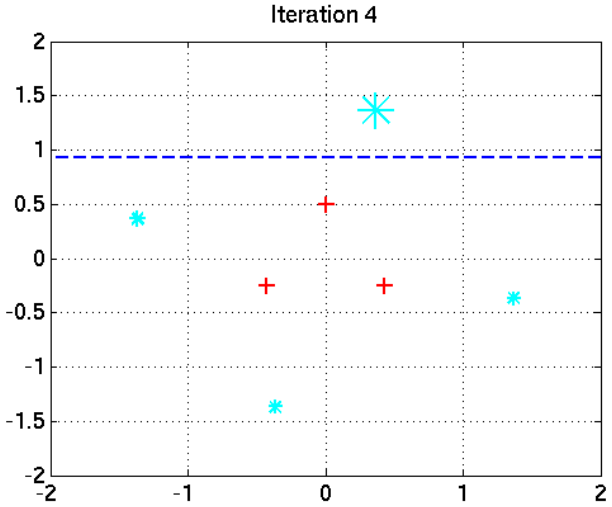


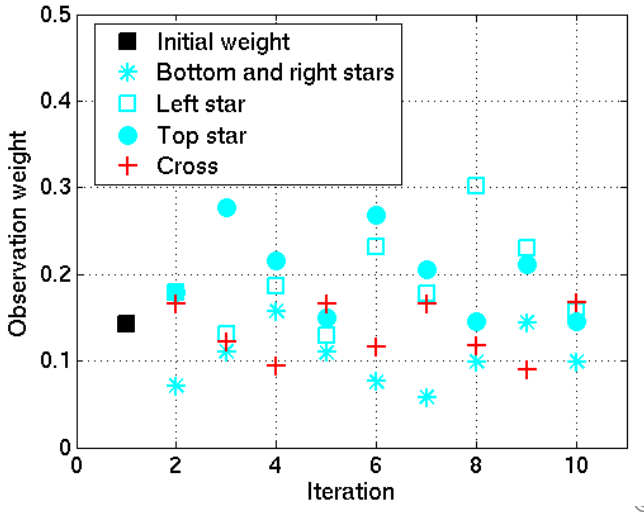




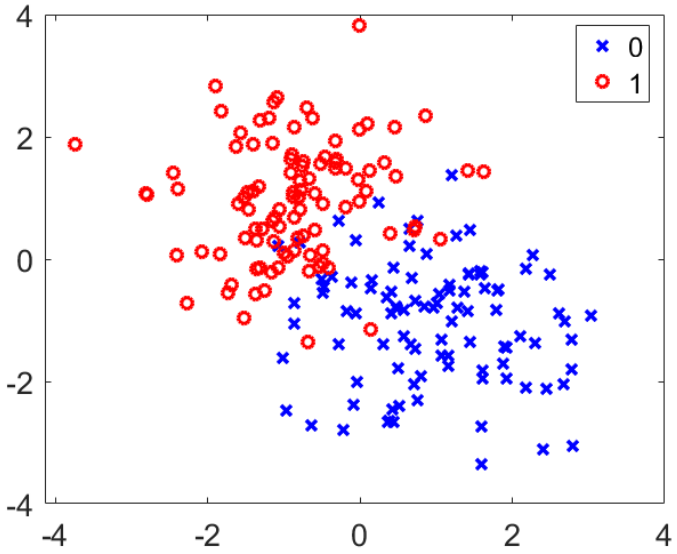
After 10 stumps

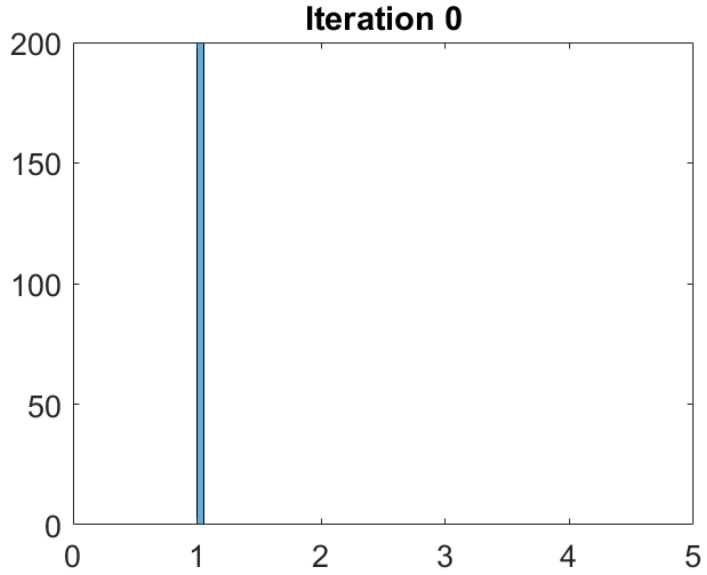


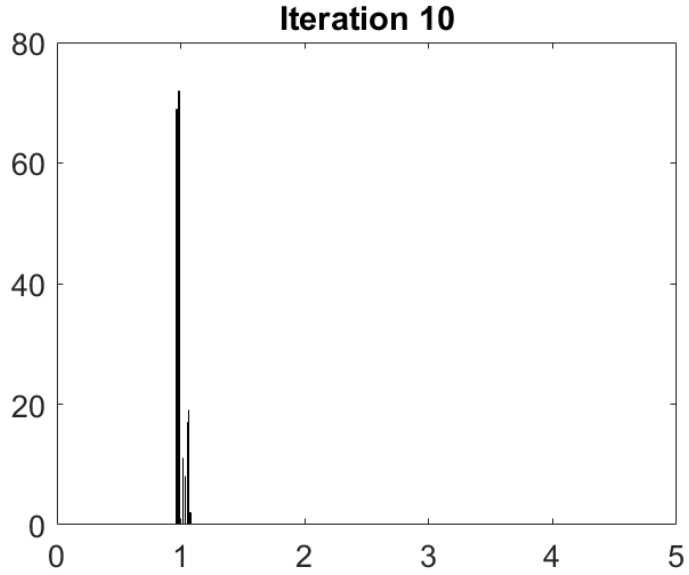


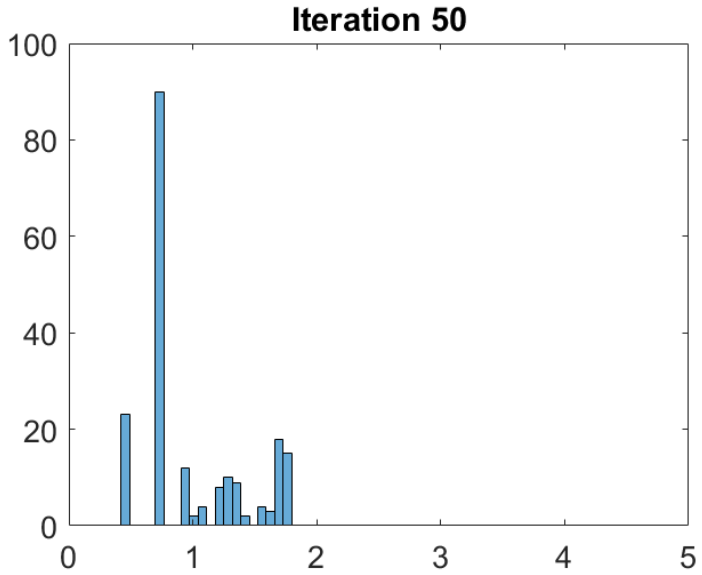


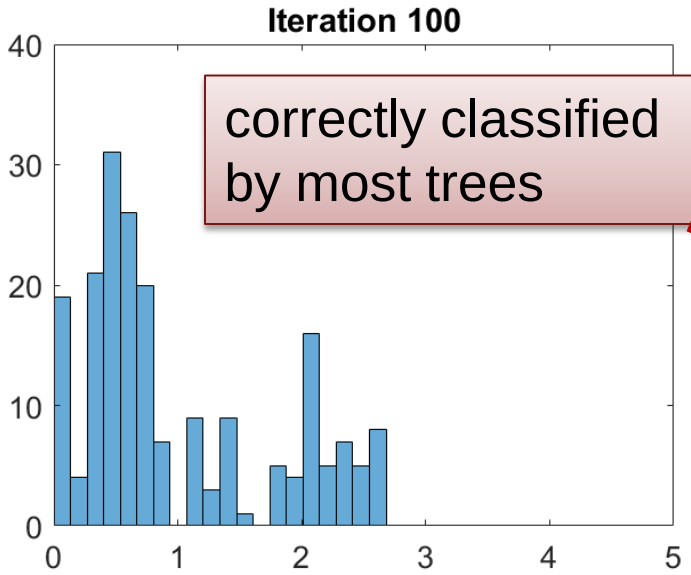
Example: evolution of observation weights

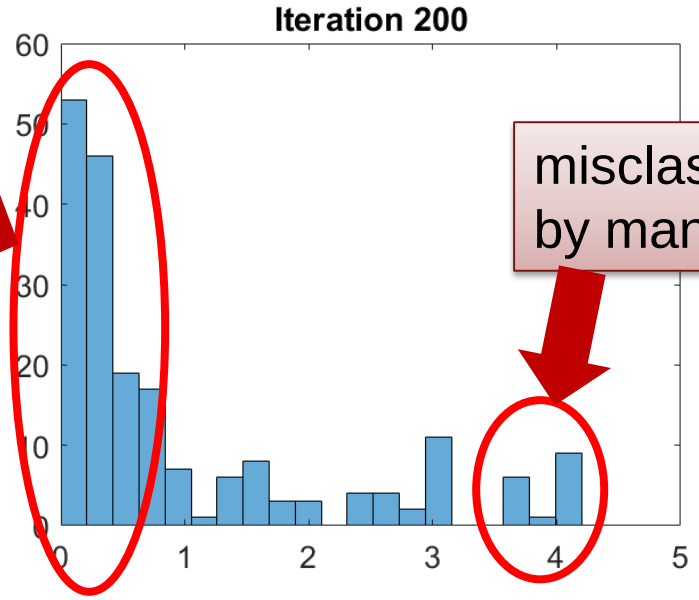












correctly classified by most trees

misclassified by many trees

Types of boosting

- **Gradient boosting, aka stagewise additive modeling**
  - J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics, Vol. 29, No. 5 (2001), pp. 1189-1232*
  - J. Friedman, T. Hastie and R. Tibshirani, Additive Logistic Regression: A Statistical View of Boosting, *The Annals of Statistics, Vol. 28, No. 2 (2000), pp. 337-407*
  - Appears more popular, at least judging by software survey. Better known in HEP.
- **Maximization of the minimal margin**
  - R. Schapire, Y. Freund, P. Bartlett and W.S. Lee, Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods, *The Annals of Statistics, Vol. 26. No. 5 (1998), pp. 1651-1686*
  - R. Schapire and Y. Freund, Boosting, MIT Press, 2012
  - Interesting extensions such as robust boosting (non-convex optimization)

Gradient boosting: Problem formulation

Loss per observation:

$$\ell\big(y, f(x)\big) = \exp\big(-yf(x)\big)$$

$y \in \{-1, +1\}$ (true label)

$f(x) \in (-\infty, +\infty)$ (predicted soft score)

Prediction at iteration t:

$$f_t(x) = \begin{cases} 0 & \text{if } t = 1 \\ \sum_{i=1}^{t-1} \alpha_i h_i(x) & \text{if } t > 1 \end{cases}$$

Objective: Minimize expected loss:

$$L = E_{X,Y}\big[\exp\big(-Yf(X)\big)\big]$$

by minimizing expected loss at x at iteration $t + 1$:

$$L_{t+1}(x) = E_Y\left[\exp\left(-Y\big(f_t(x) + \alpha_t h_t(x)\big)\right)\right]$$

α_t - ?

h_t - ?

Gradient boosting for binary classification (GentleBoost)

Assume $\alpha_t = 1$

Contribution of n-th observation to the empirical loss:

$$w_n^{(t+1)} \ell\left(y_n, f_{t+1}(x_n)\right) = w_n^{(t)} \exp\left(-y_n\left(f_t(x_n) + h_t(x_n)\right)\right)$$

Update weights: $w_n^{(t+1)} = w_n^{(t)} \exp\left(-y_n h_t(x_n)\right)$

$$E_Y^{(t+1)}[Y|x] = -\frac{\frac{\partial L_{t+1}(x)}{\partial h_t(x)}}{\frac{\partial^2 L_{t+1}(x)}{\partial h_t^2(x)}}$$

Newton update

$$h_t(x) = E_Y^{(t+1)}[Y|x]$$

$E_Y^{(t+1)}[Y|x]$: fit y_n by least squares on x_n with weights $w_n^{(t)}$

Copied from Additive Logistic Regression
by Friedman, Hastie & Tibshirani

Gentle AdaBoost

1. Start with weights $w_i = 1/N, i = 1, 2, \ldots, N, F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:

   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$.
   (c) Update $w_i \leftarrow w_i \exp(-y_i f_m(x_i))$ and renormalize.

3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

Gradient boosting for binary classification – continued

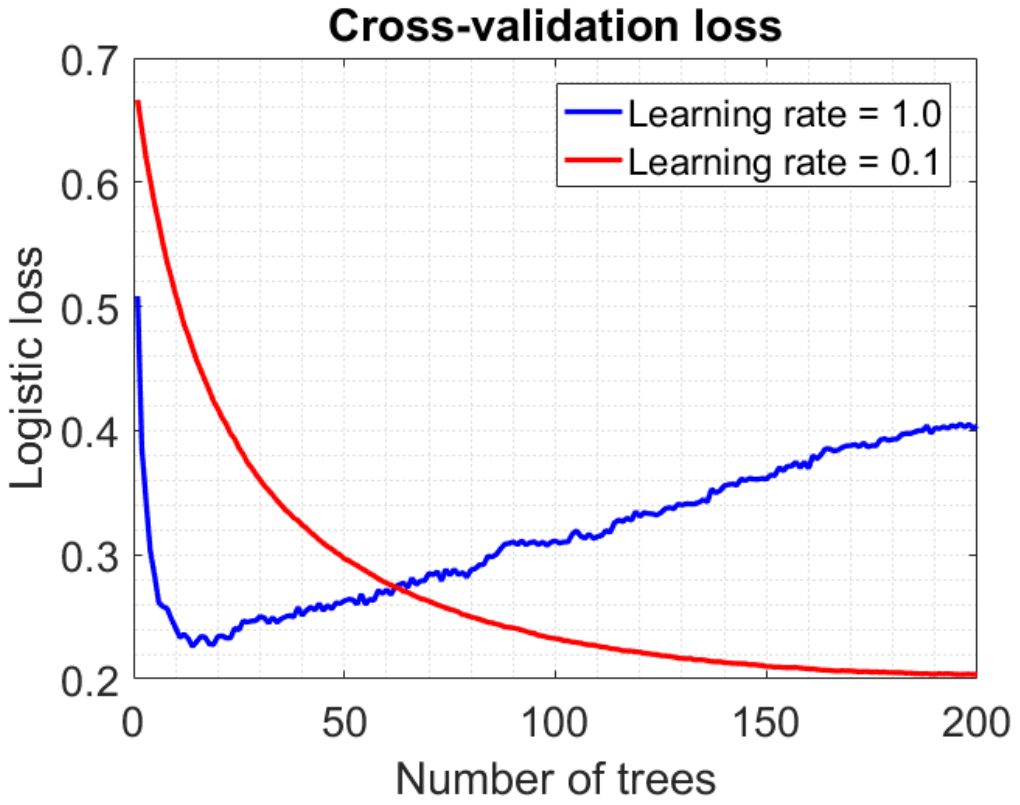
From soft scores to posterior probabilities:

$$f(x) = \arg\min_{q(x)} E_Y\left[\exp\left(-Yq(x)\right)\right] \quad \Leftrightarrow \quad f(x) = \frac{1}{2}\log\frac{P(Y=+1|x)}{P(Y=-1|x)}$$

Overfitting is an issue!

Learning rate can help prevent overfitting:

$$h_t(x) = \eta E_Y^{(t)}[Y|x] \qquad 0 < \eta \leq 1$$



Boosting: Practical advice

- Boosting is typically used with shallow trees
  - Decision stumps is the case most often discussed in the literature
- Most important tuning knobs:
  - number of trees in the ensemble
  - stopping criterion for tree (min leaf size, max number of splits etc)
  - for some algorithms (notably, LogitBoost and GentleBoost), also learning rate
- Unlike random forest, adding more trees does not necessarily help
  - Monitor the appropriate loss function versus number of trees on a validation set or by cross-validation
- With proper parameter tuning, boosted trees offer superb accuracy
- Because trees are typically shallow, boosted trees are not a good tool for estimation of variable importance
  - Few splits per variable, not enough statistics

Random forest vs boosted trees

Feature	Random forest	Boosted trees
Accuracy	●	✓
Speed of training	●	●
Speed of prediction	×	✓
Easy selection of optimal parameters	✓	×
Memory requirements	×	✓
Estimation of variable importance and variable interactions	✓	×
Out-of-bag estimates (quick estimates of test error)	✓	×
Estimation of proximities and outliers	✓	×

✓ good
● OK
× poor

The accuracy and speed ratings are very approximate and do not hold for every dataset. Take with a grain of salt!

The End