

# CS109 – Data Science

Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau

[vkaynig@seas.harvard.edu](mailto:vkaynig@seas.harvard.edu)

staff@cs109.org

# Announcements

- HW2 is due today!
- Please execute your notebooks, but without test output.
- Help with lecture material

# Books

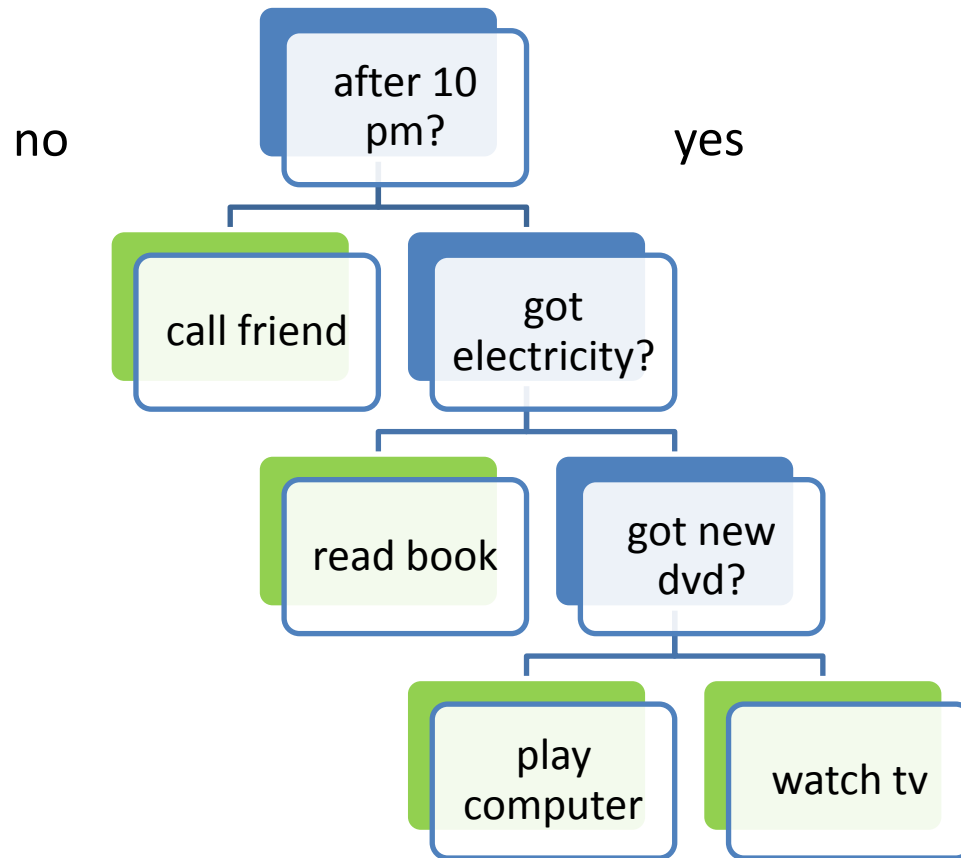
- “Elements of Statistical Learning”
- <http://statweb.stanford.edu/~tibs/ElemStatLearn/>
- “Pattern Recognition and Machine Learning”
- <http://research.microsoft.com/en-us/um/people/cmbishop/PRML/>

# Next Topics

- Tree classifier
- Bagging
- Random Forest



# Decision Tree

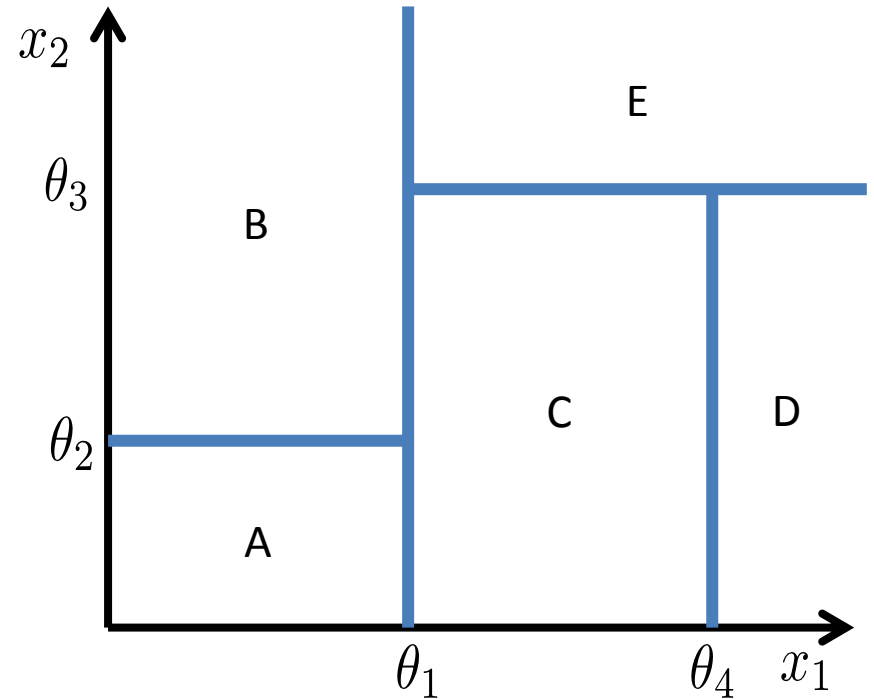
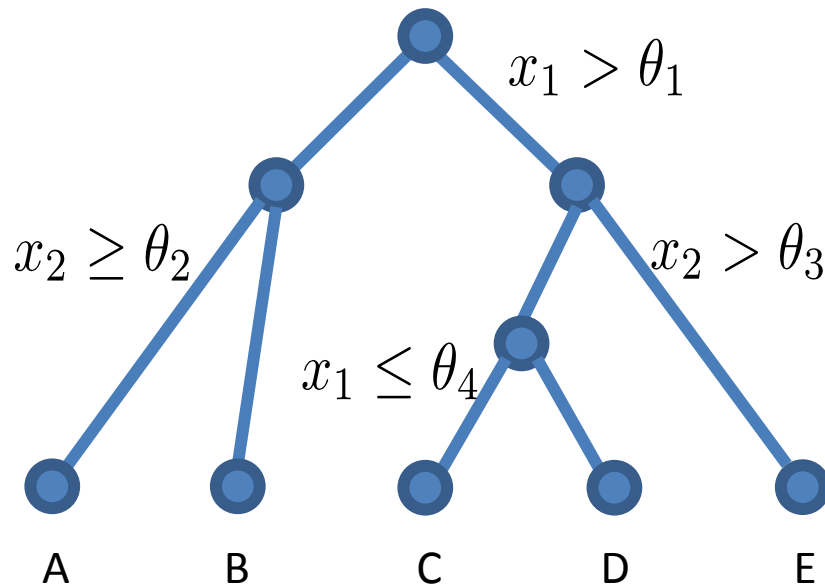


# Decision Trees

- Fast training
- Fast prediciton
- Easy to understand
- Easy to interpret

<http://en.akinator.com/personnages/jeu>

# Decision Tree - Idea

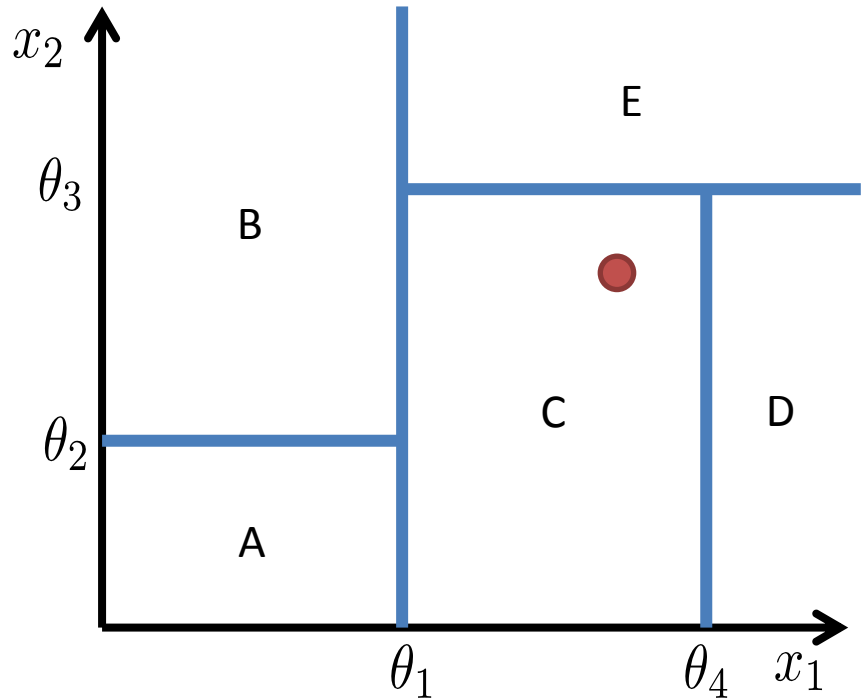
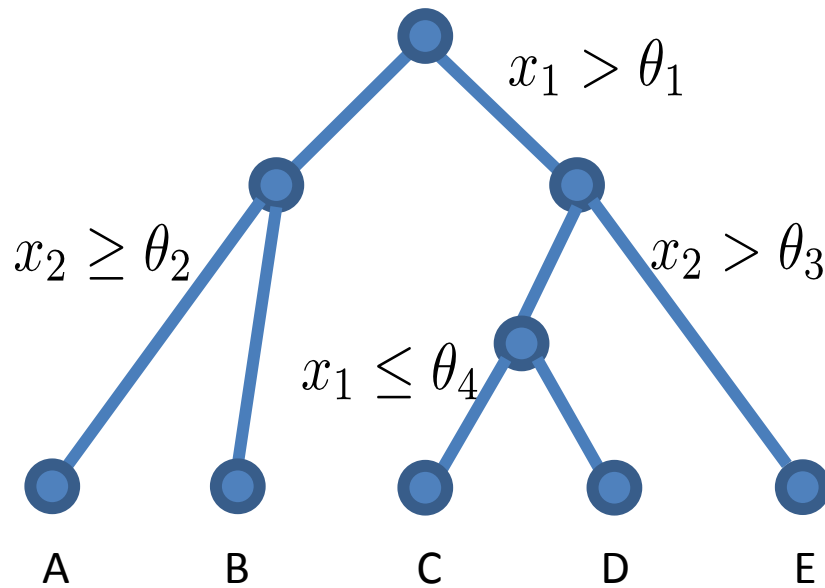


# Decision Tree - Idea

- What is the benefit of using only one feature at a time?
- What is the drawback?

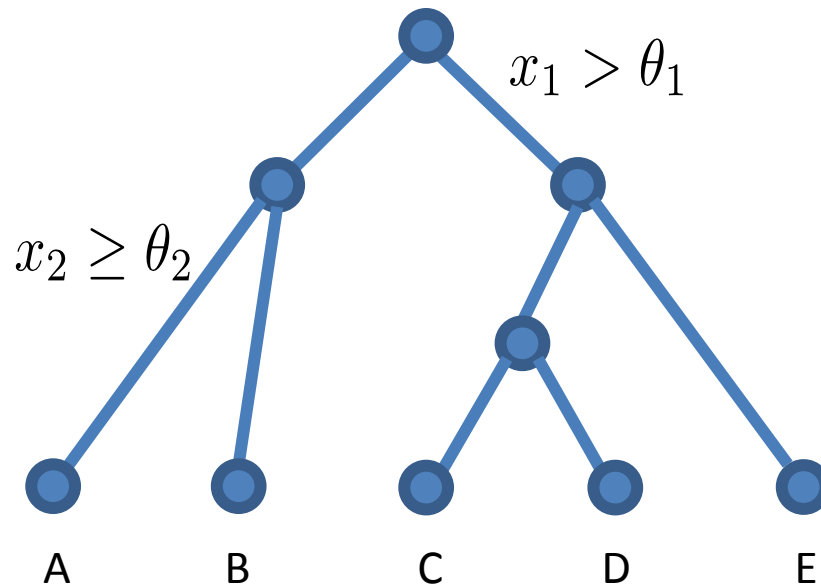


# Decision Tree - Prediction

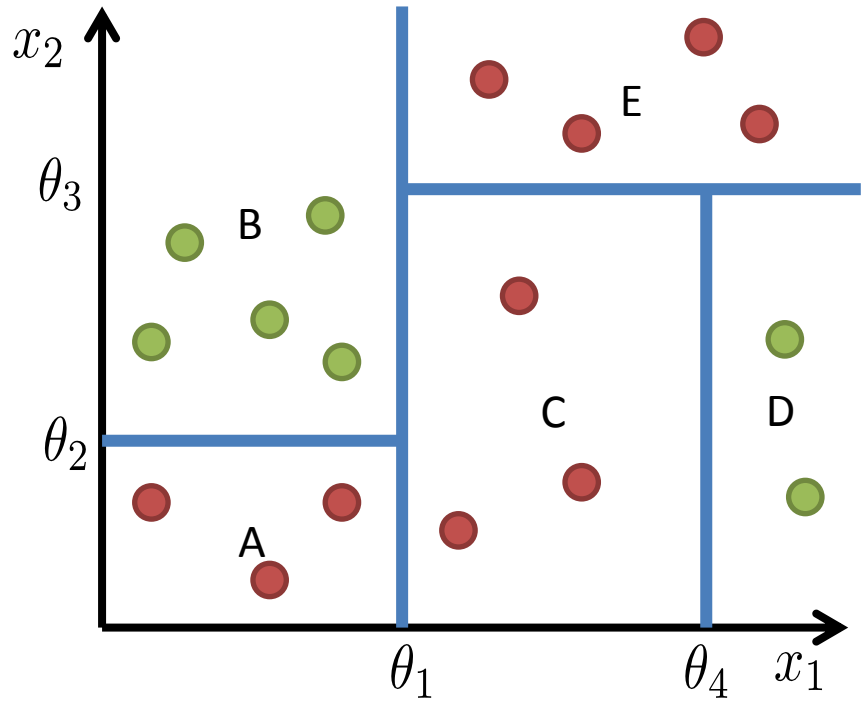
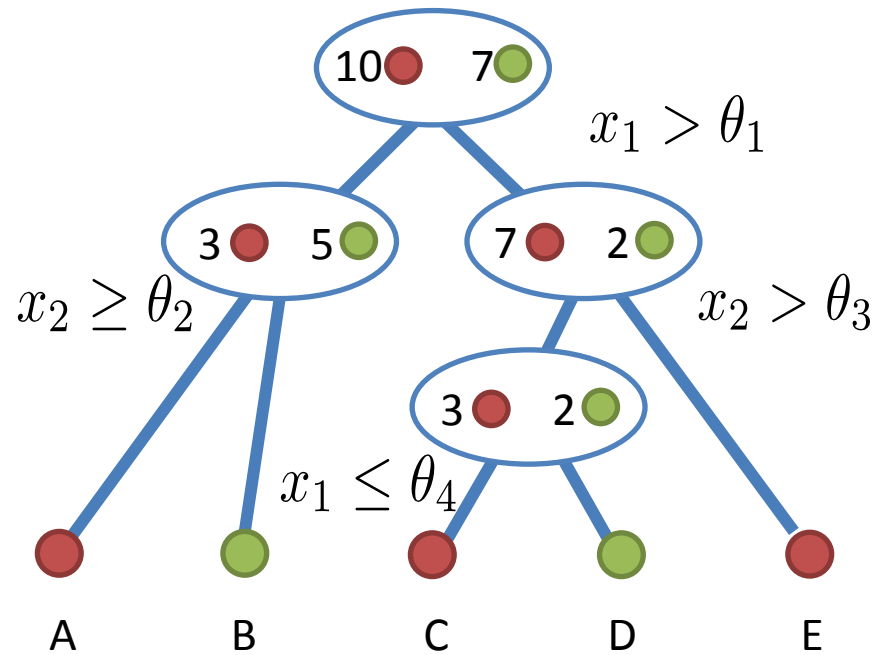


# Decision Tree -Training

- Learn the tree structure:
  - which feature to query
  - which threshold to choose

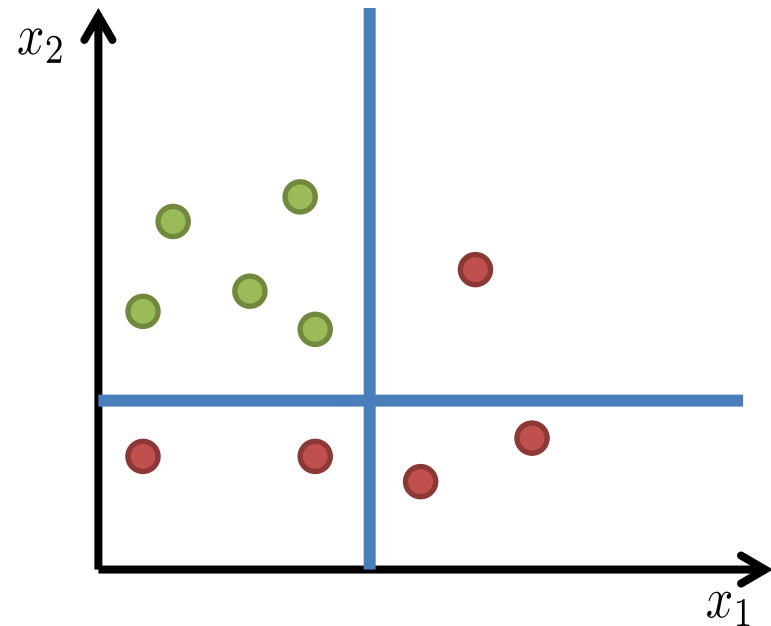


# Node Purity



# Gini Impurity

- Expected error
- if you randomly choose a sample
- and predict the class of the entire node based on it.



# Gini Impurity

Example:

4 **red**, 3 **green**, 3 **blue** data points

- Class probabilities:

— **red**:  $4/10$       **green**:  $3/10$       **blue**:  $3/10$

- misclassification:

— **red**:  $4/10 * (3/10 + 3/10)$

Picking  
red

Making an  
error

# Gini Impurity

- misclassification:

— red:

$$4/10 * (3/10 + 3/10) = 0.24$$

— green and blue:

$$3/10 * (4/10 + 3/10) = 0.21$$

- gini impurity: **0.24** + **0.21** + **0.21** = 0.66

# Gini Impurity

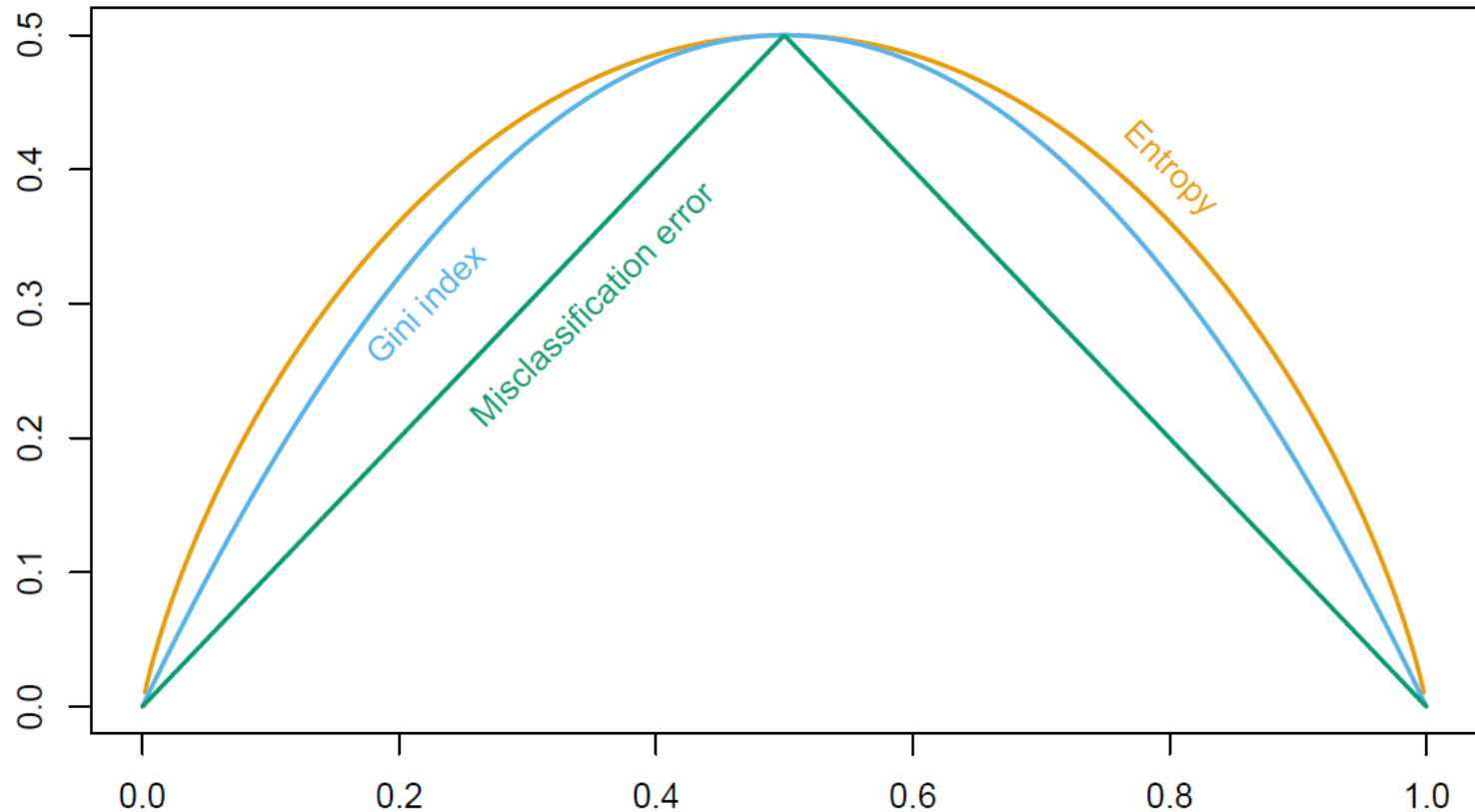
- Number of classes:  $C$
- Number of data points:  $N$
- Number of data points of class  $i$ :  $N_i$

$$I_G = \sum_{i=1}^C \frac{N_i}{N} \left( 1 - \frac{N_i}{N} \right)$$

true  
class

wrong  
prediction

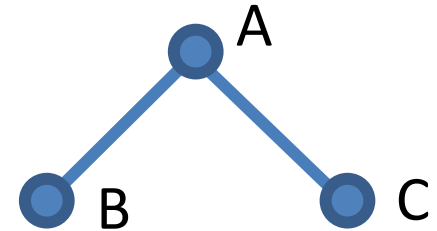
# Gini Impurity





# Node Purity Gain

- Compare:
  - Gini impurity of parent node
  - Gini impurity of child nodes



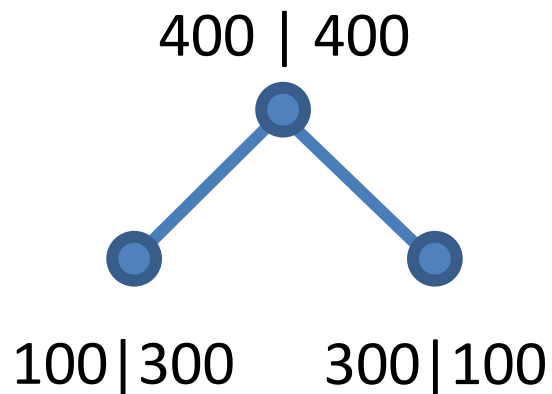
$$\Delta I_G = I_G(A) - \frac{N(B)}{N(A)} I_G(B) - \frac{N(C)}{N(A)} I_G(C)$$

# Misclassification

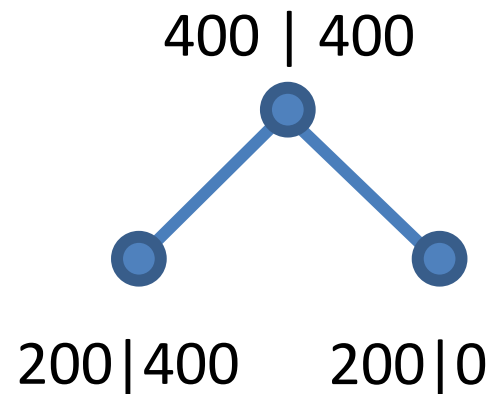
- $\frac{1}{N} \sum_i^N \mathbf{1}(\hat{y}_i \neq y_i)$
- not differentiable

# Comparison Gini vs Misclassification

- Binary problem: 400 samples per class



Misclassification: 0.25  
Gini gain: 0.125



Misclassification: 0.25  
Gini gain: 0.166

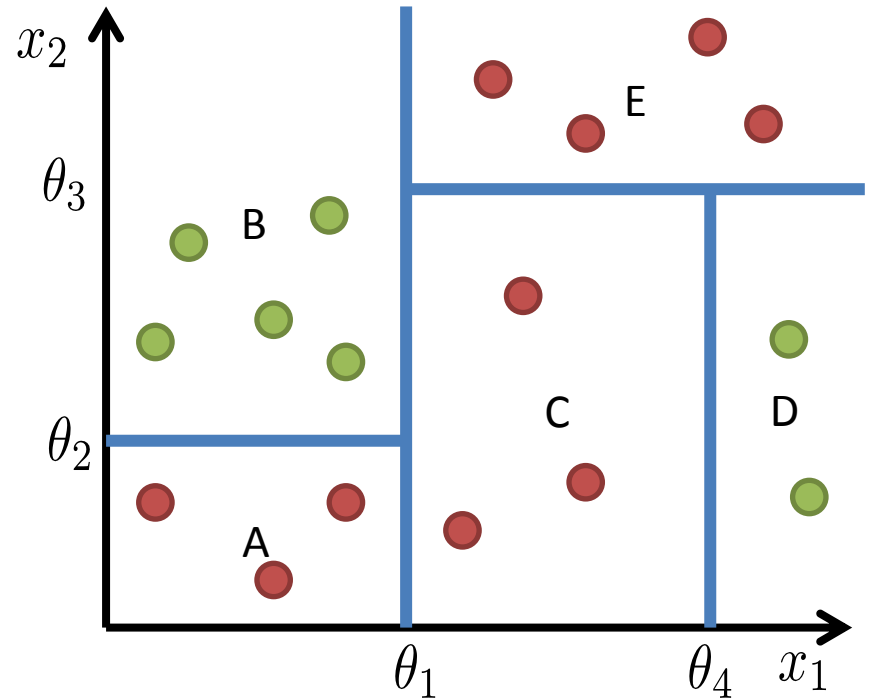
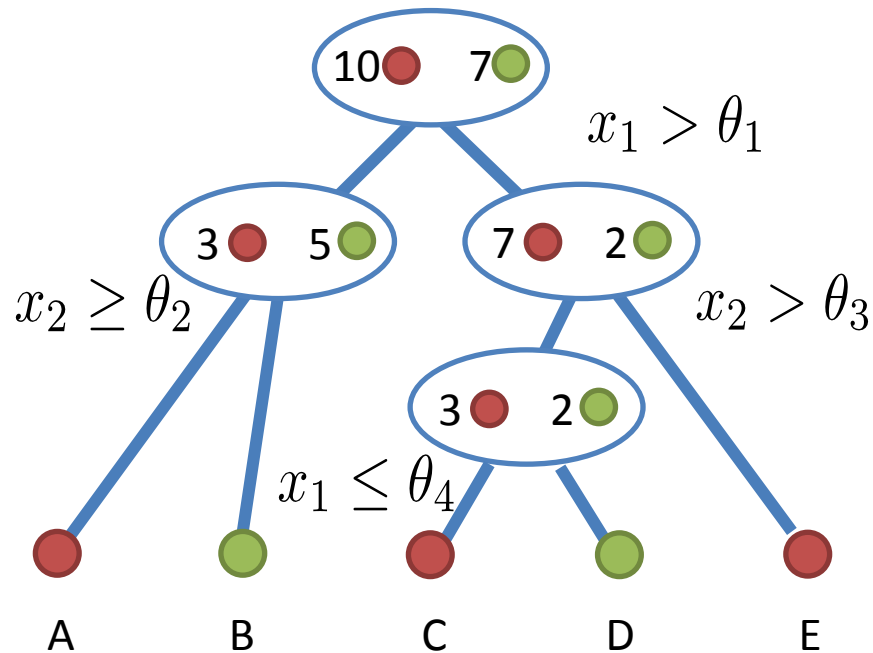
# Pseudocode

- Check if already finished
- For each feature  $x_i$ 
  - Calculate the gain from splitting on  $x_i$
  - Let  $x_{best}$  be the feature with highest gain
- Create a decision *node* that splits on  $x_{best}$
- Repeat on the sub-nodes
  
- Does this produce an optimal tree?
- What does optimal tree mean?

# When to Stop

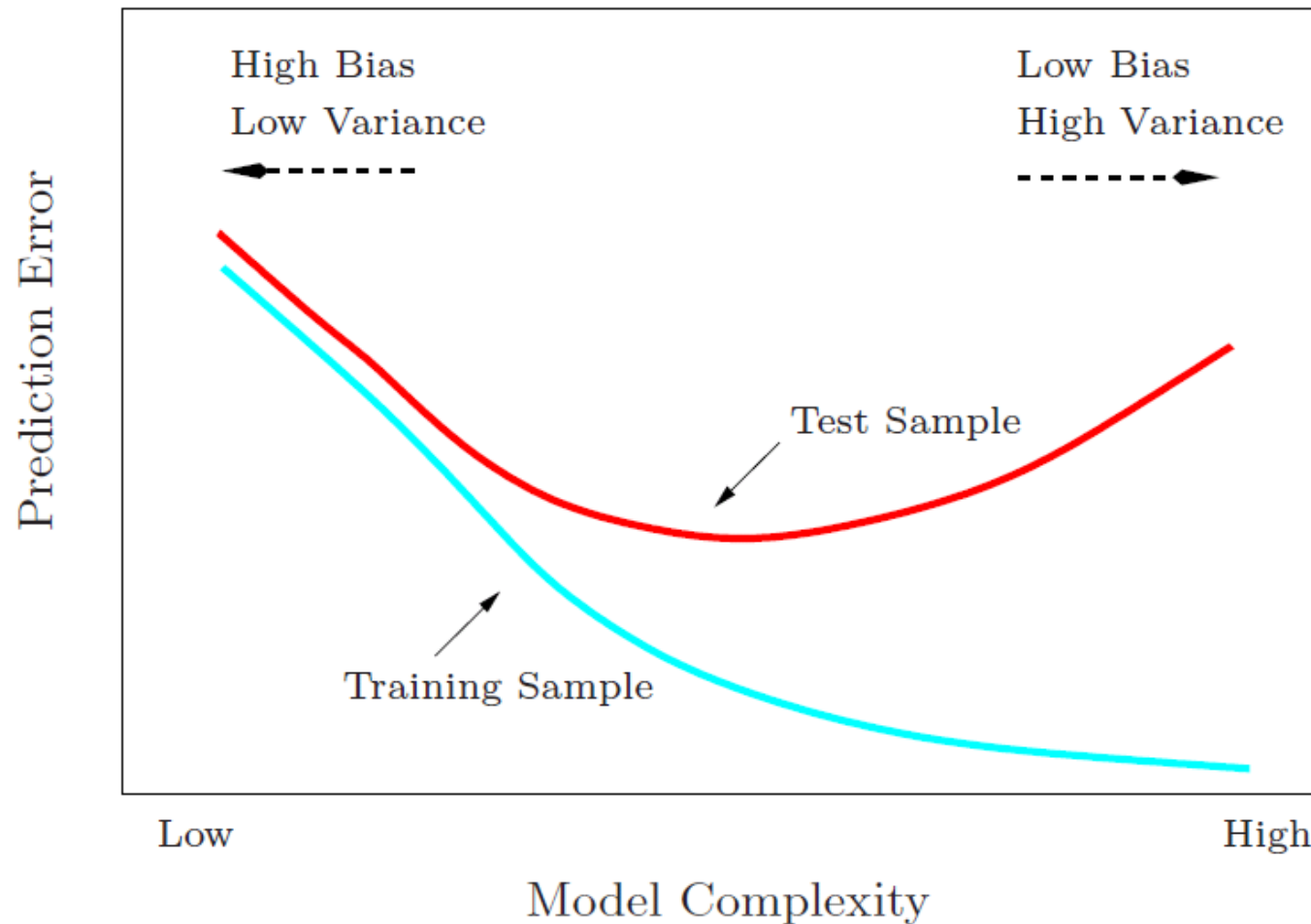
- node contains only one class
- node contains less than  $x$  data points
- max depth is reached
- node purity is sufficient
- you start to overfit => cross-validation

# Tree Pruning



How do you make a prediction for the merged cell?

# Pruning and Complexity



# Decision Trees - Disadvantages

- Sensitive to small changes in the data
- Overfitting
- Only axis aligned splits



# Decision Trees vs SVM

Characteristic	SVM	Trees
Natural handling of data of “mixed” type	▼	▲
Handling of missing values	▼	▲
Robustness to outliers in input space	▼	▲
Insensitive to monotone transformations of inputs	▼	▲
Computational scalability (large $N$ )	▼	▲
Ability to deal with irrelevant inputs	▼	▲
Ability to extract linear combinations of features	▲	▼
Interpretability	▼	◆
Predictive power	▲	▼

# Wisdom of Crowds

The collective knowledge of a **diverse and independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

James Surowiecki





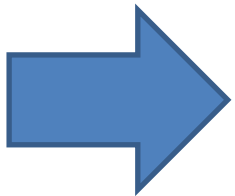
**Netflix Prize**

# Netflix Prize

- Take home messages:

# Ensemble Methods

- A single decision tree does not perform well
- But, it is super fast
- What if we learn multiple trees?



We need to make sure they do not all just learn the same.

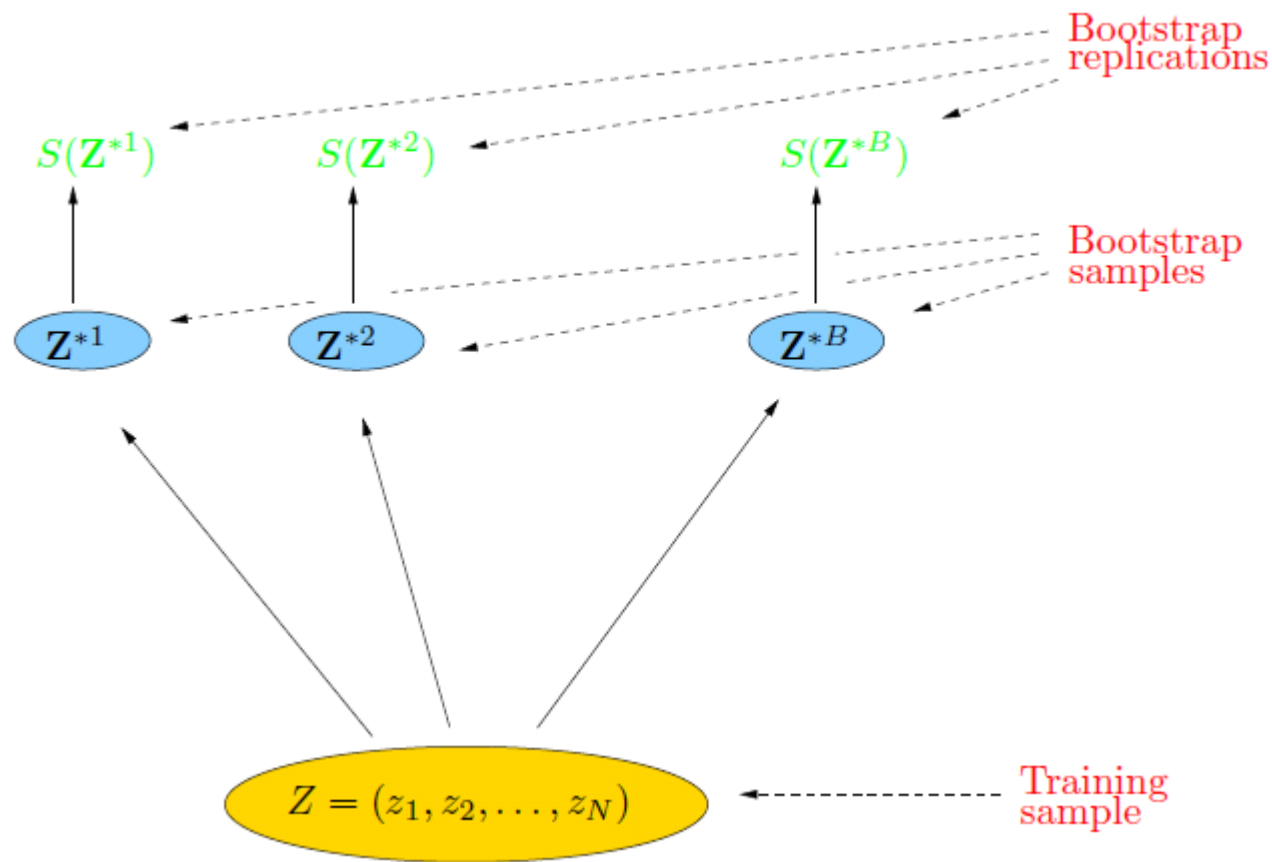
# Bootstrap



# Bootstrap

- Resampling method from statistics
- Useful to get error bars on estimates
- Take  $N$  data points
- Draw  $N$  times with replacement
- Get estimate from each bootstrapped sample

# Bootstrap



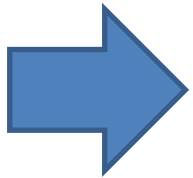


# Bootstrap

- I can generate more data!
- Can I do cross validation on this?

# Bootstrap vs Cross-validation

- Bootstrap has overlap in data sets



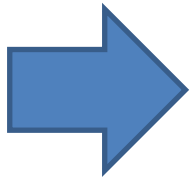
Do not use simple bootstrap to generate train and test data from same data set.

$$p(n \in Z^{*i}) = \frac{1}{N}$$

Probability of choosing n

# Bootstrap vs Cross-validation

- Bootstrap has overlap in data sets



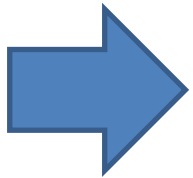
Do not use simple bootstrap to generate train and test data from same data set.

$$p(n \in Z^{*i}) = 1 - \frac{1}{N}$$

Probability of not choosing n

# Bootstrap vs Cross-validation

- Bootstrap has overlap in data sets



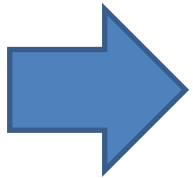
Do not use simple bootstrap to generate train and test data from same data set.

$$p(n \in Z^{*i}) = \left(1 - \frac{1}{N}\right)^N$$

Probability of not choosing  $n$  in  $N$  draws

# Bootstrap vs Cross-validation

- Bootstrap has overlap in data sets



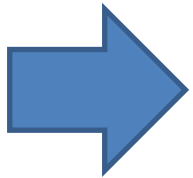
Do not use simple bootstrap to generate train and test data from same data set.

$$p(n \in Z^{*i}) = 1 - \left(1 - \frac{1}{N}\right)^N$$

Probability of (not not) choosing  $n$  in  $N$  draws

# Bootstrap vs Cross-validation

- Bootstrap has overlap in data sets



Do not use simple bootstrap to generate train and test data from same data set.

$$p(n \in Z^{*i}) = 1 - e^{-1}$$

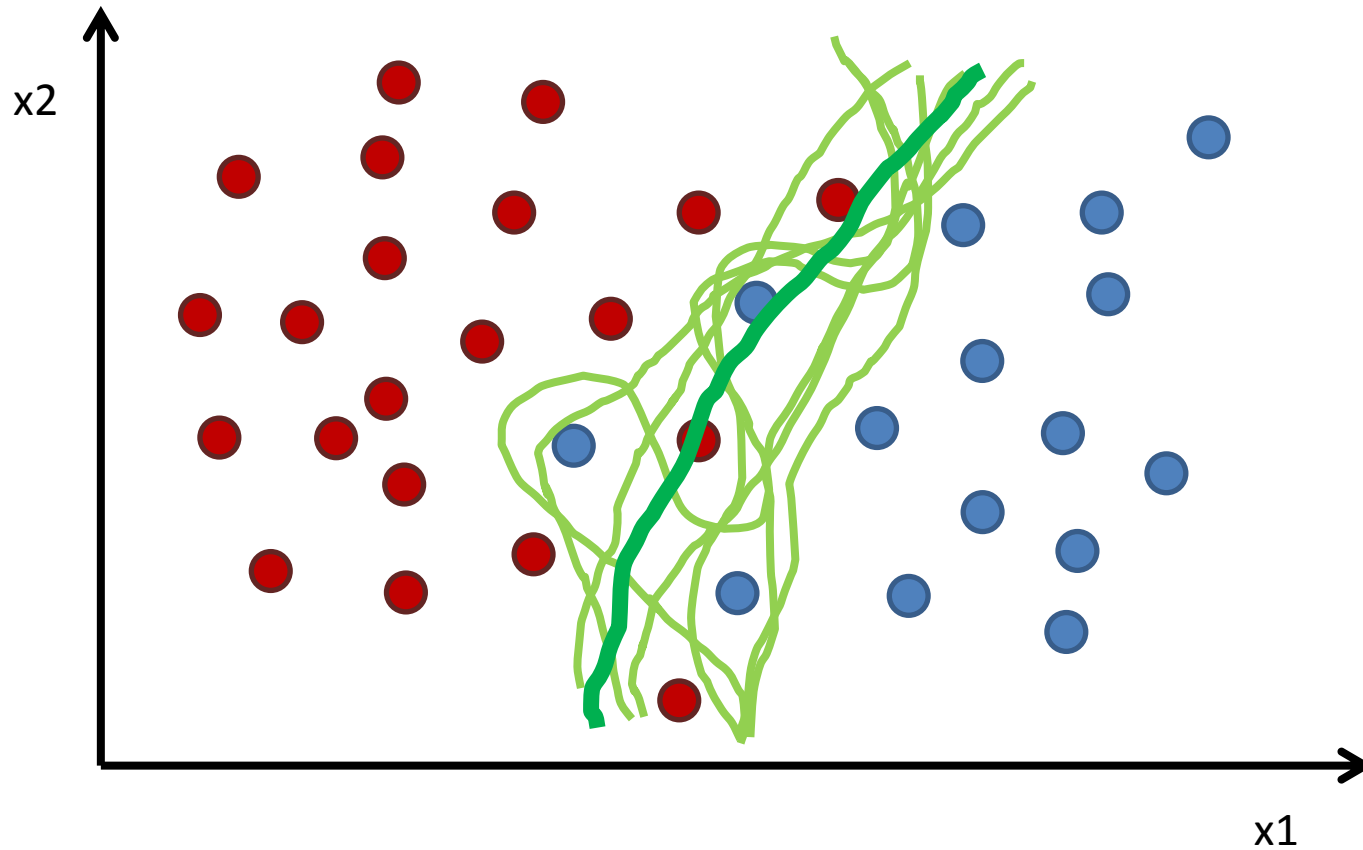
$$\approx 0.632$$

This number is important later

# Bagging

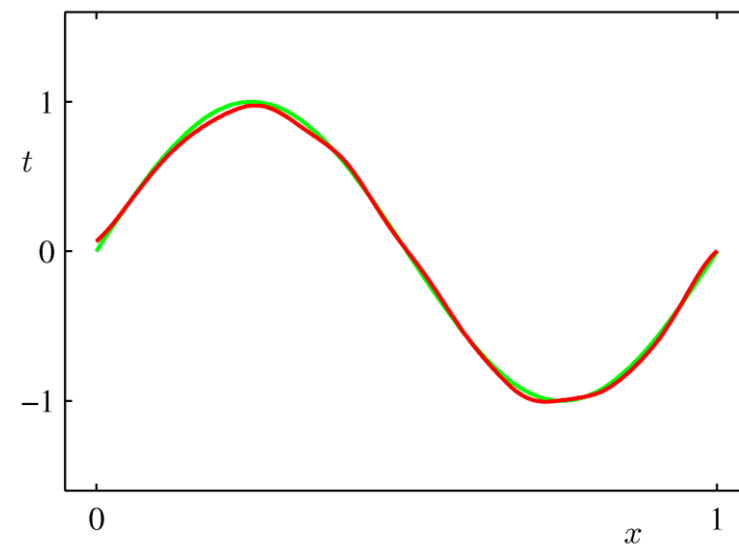
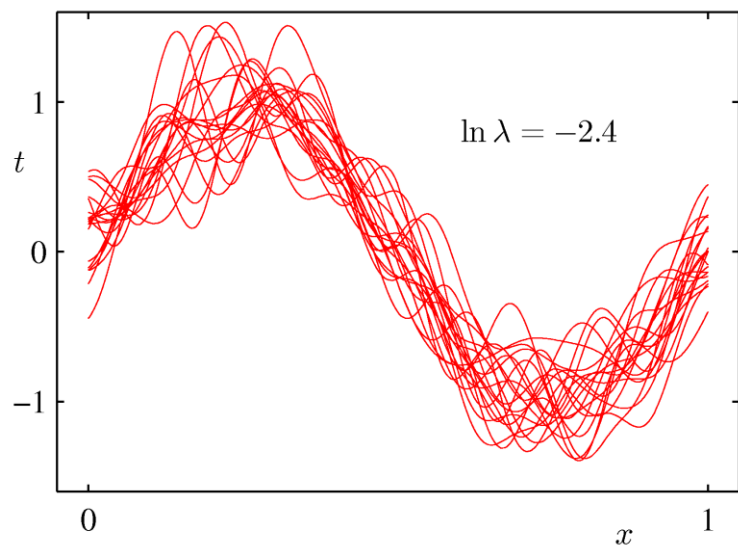
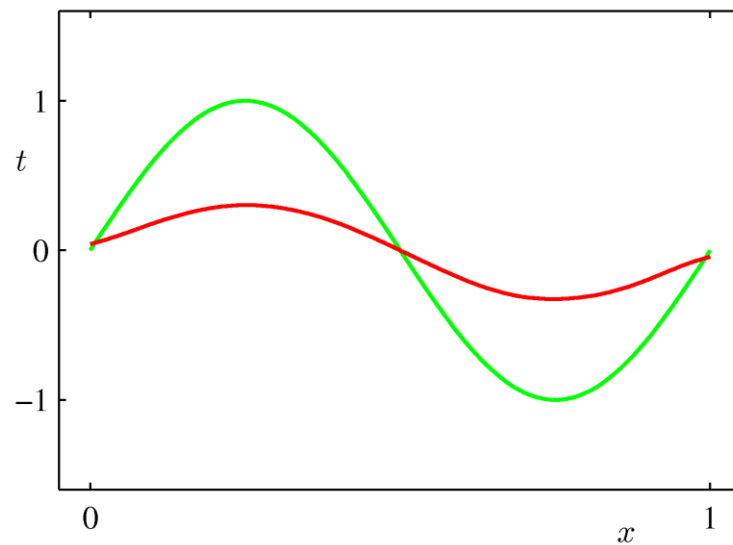
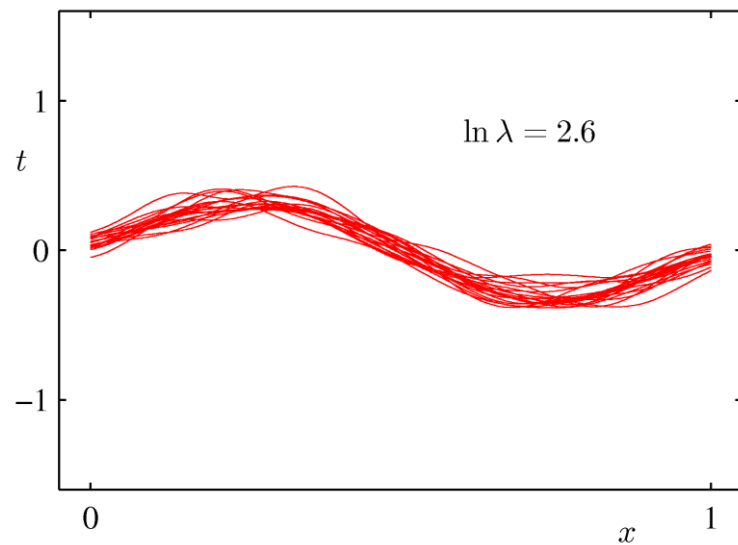
- Bootstrap aggregating
- Sample with replacement from your data set
- Learn a classifier for each bootstrap sample
- Average the results

# Bagging Example

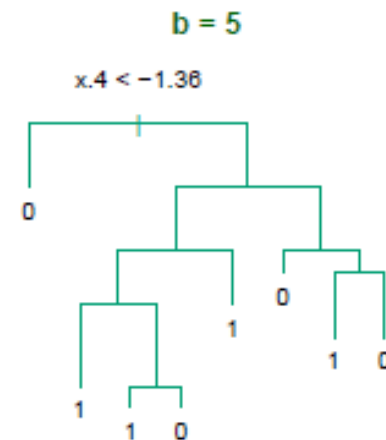
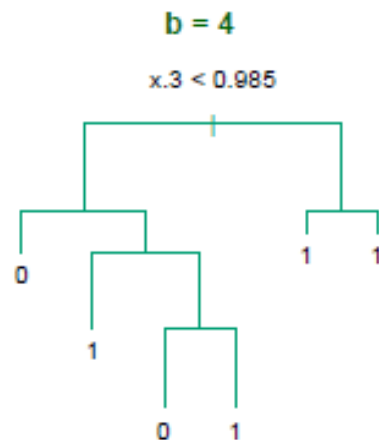
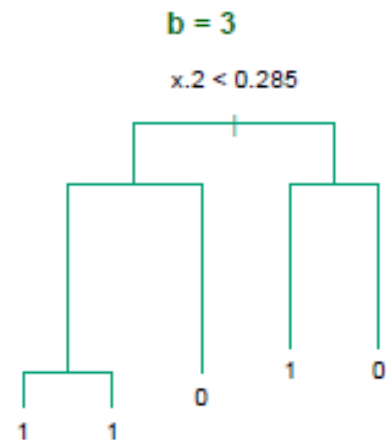
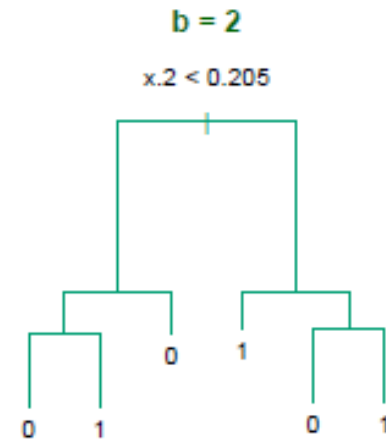
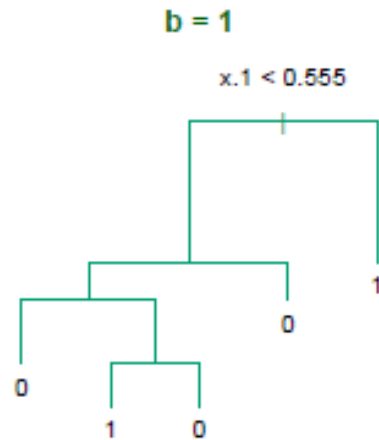
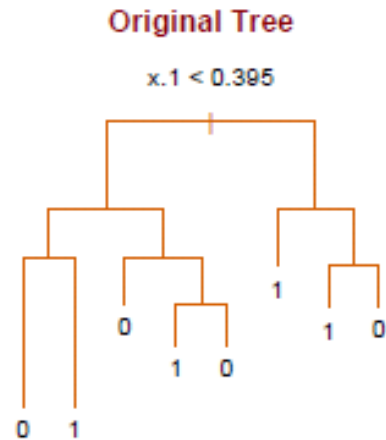




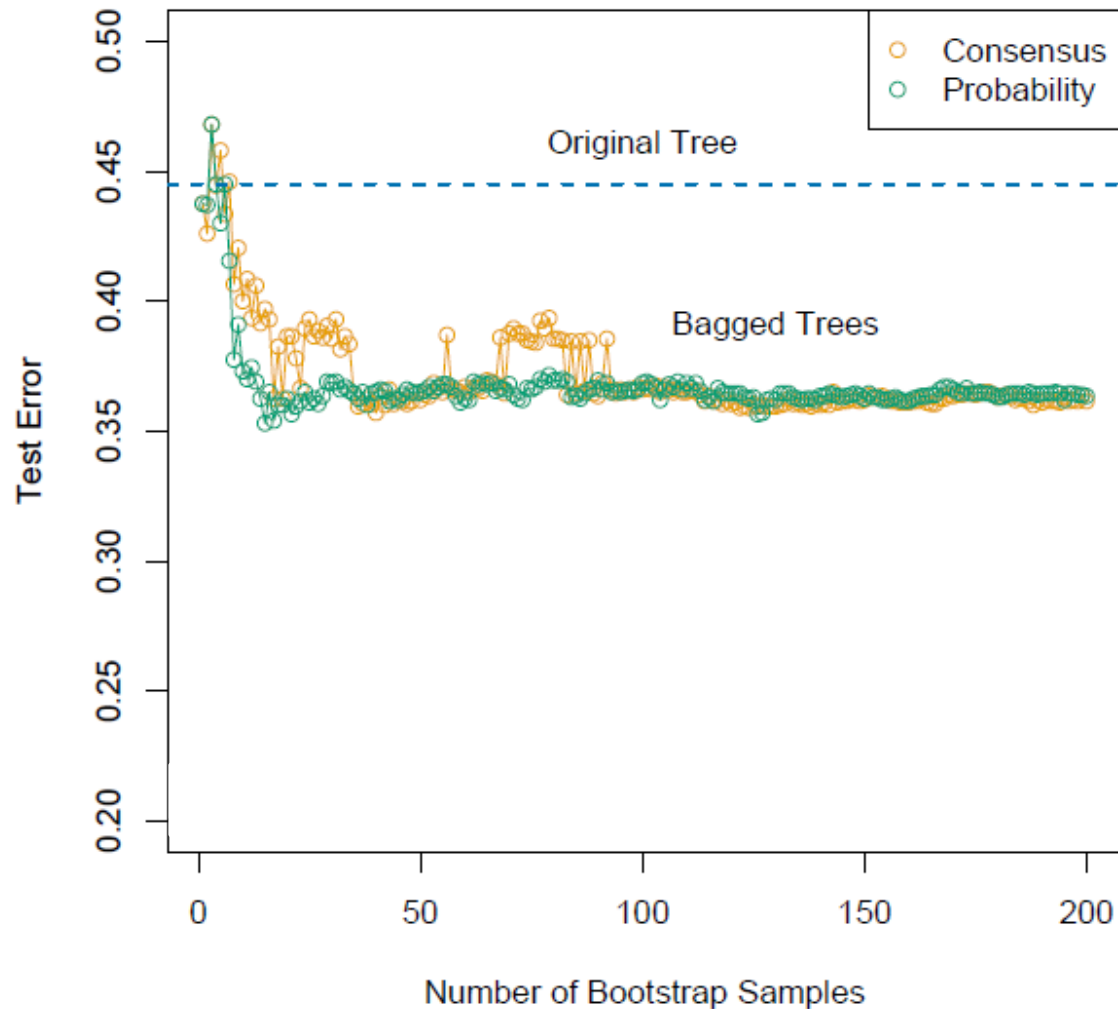
# Bias-Variance Trade-off



# Bagging Decision Trees



# Bagging Decision Trees

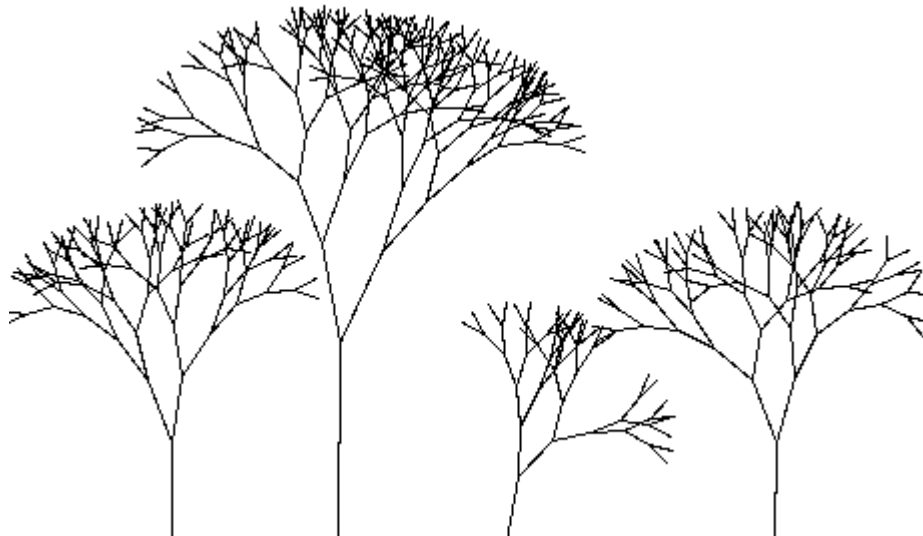


# Bagging

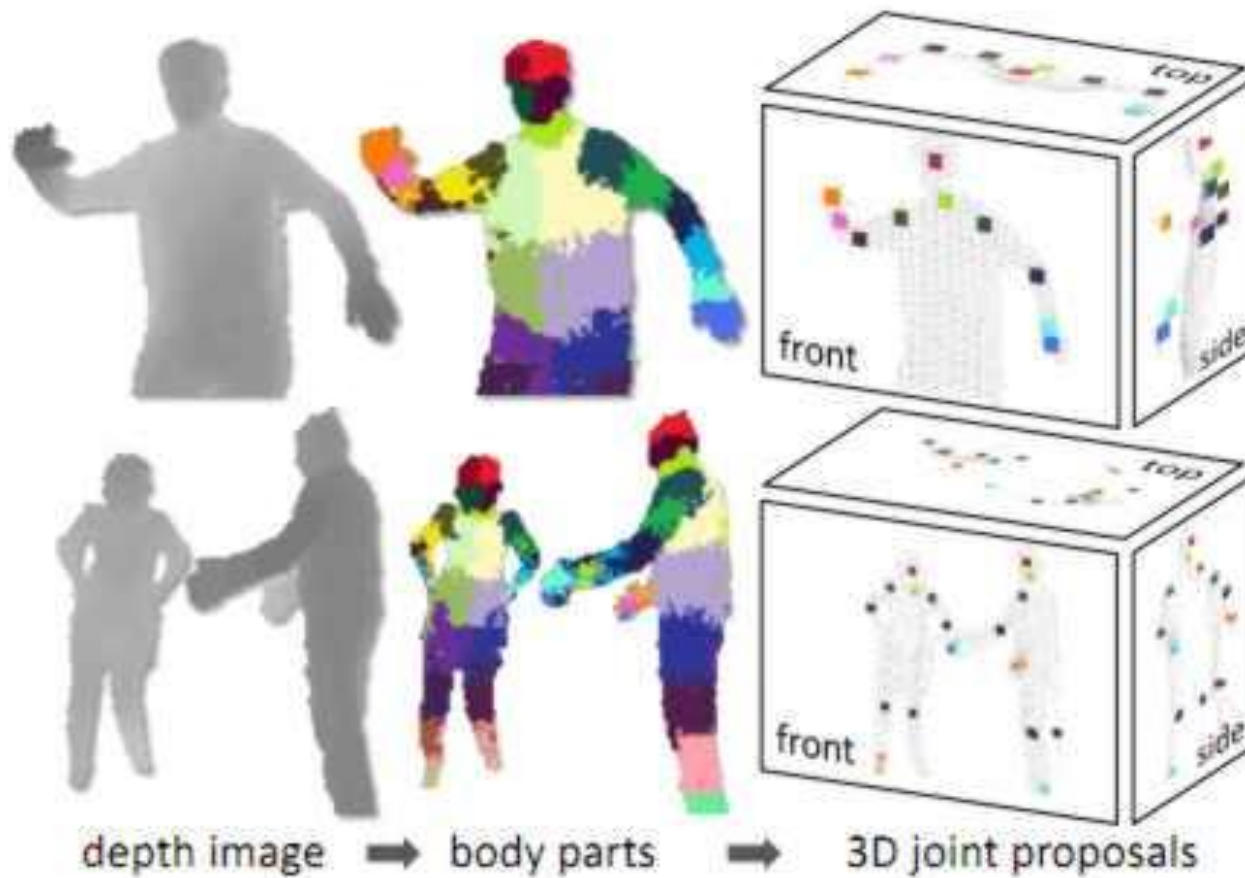
- Reduces overfitting (variance)
- Normally uses one type of classifier
- Decision trees are popular
- Not helping with linear models
- Easy to parallelize

# Random Forest

- Builds upon the idea of bagging
- Each tree build from bootstrap sample
- Node splits calculated from **random feature subsets**



# Random Forest – Fun Fact





hand\_tracking\_kinect.mp4

# Random Forest

- All trees are fully grown
- No pruning
- Two parameters
  - Number of trees
  - Number of features



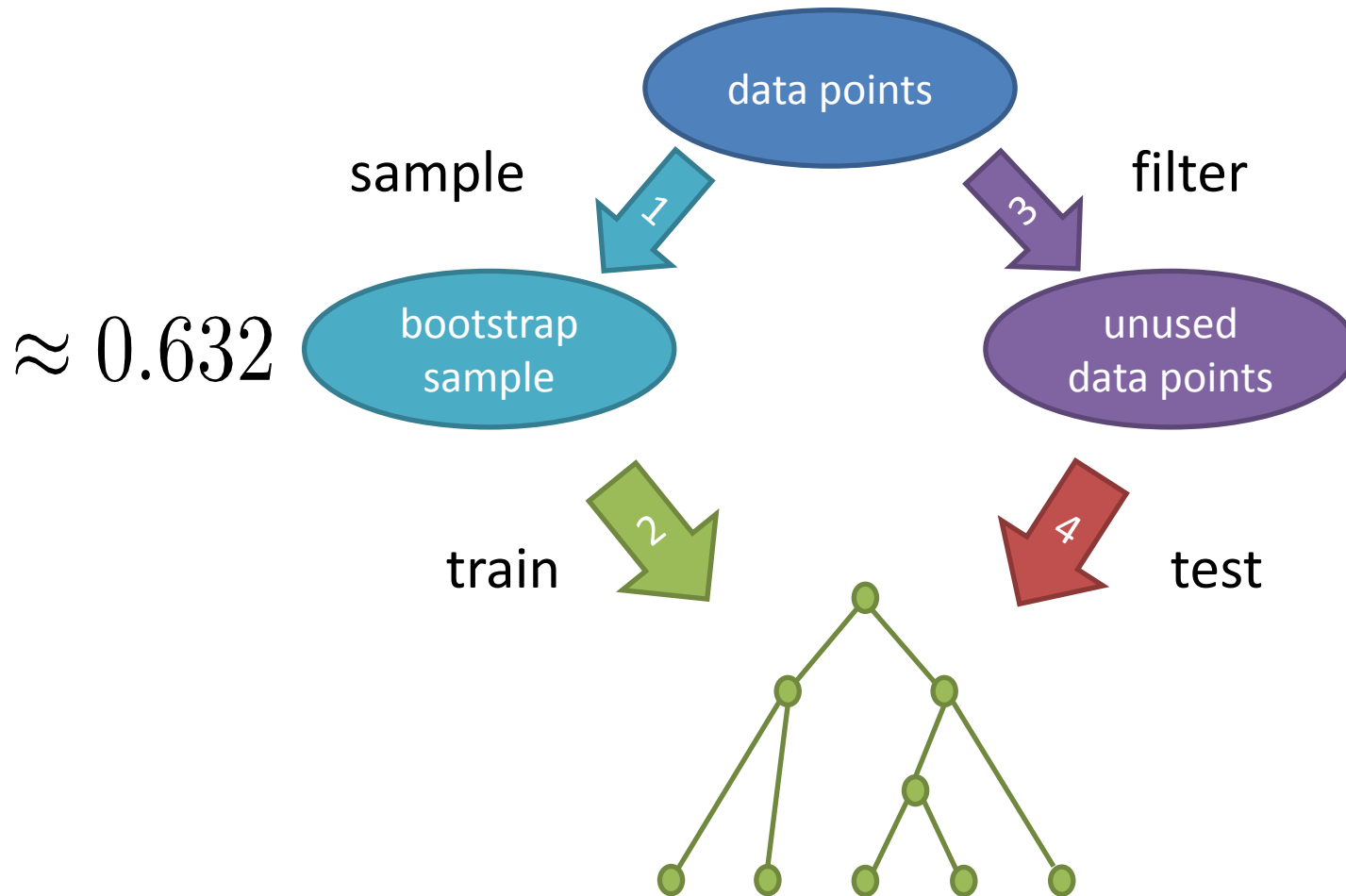
# Random Forest Error Rate

- Error depends on:
  - Correlation between trees (higher is worse)
  - Strength of single trees (higher is better)
- Increasing number of features for each split:
  - Increases correlation
  - Increases strength of single trees

# Out of Bag Error

- Each tree is trained on a bootstrapped sample
- About  $1/3$  of data points not used for training
- Predict unseen points with each tree
- Measure error

# Out of Bag Error



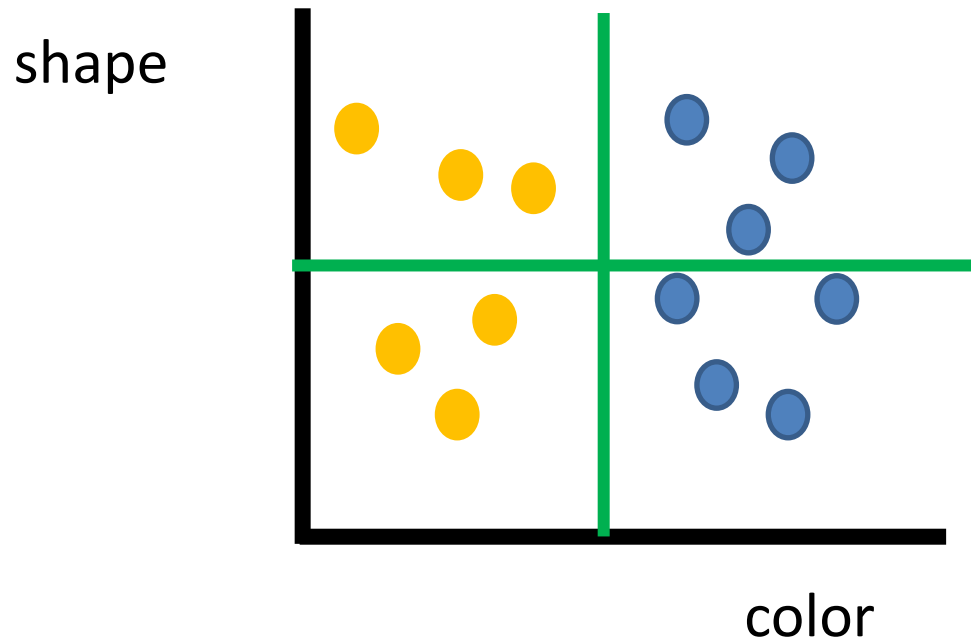
# Out of Bag Error

- Very similar to cross-validation
- Measured during training
- Can be too optimistic

# Variable Importance - 1

- Again use out of bag samples
- Predict class for these samples
- Randomly permute values of one feature
- Predict classes again
- Measure decrease in accuracy

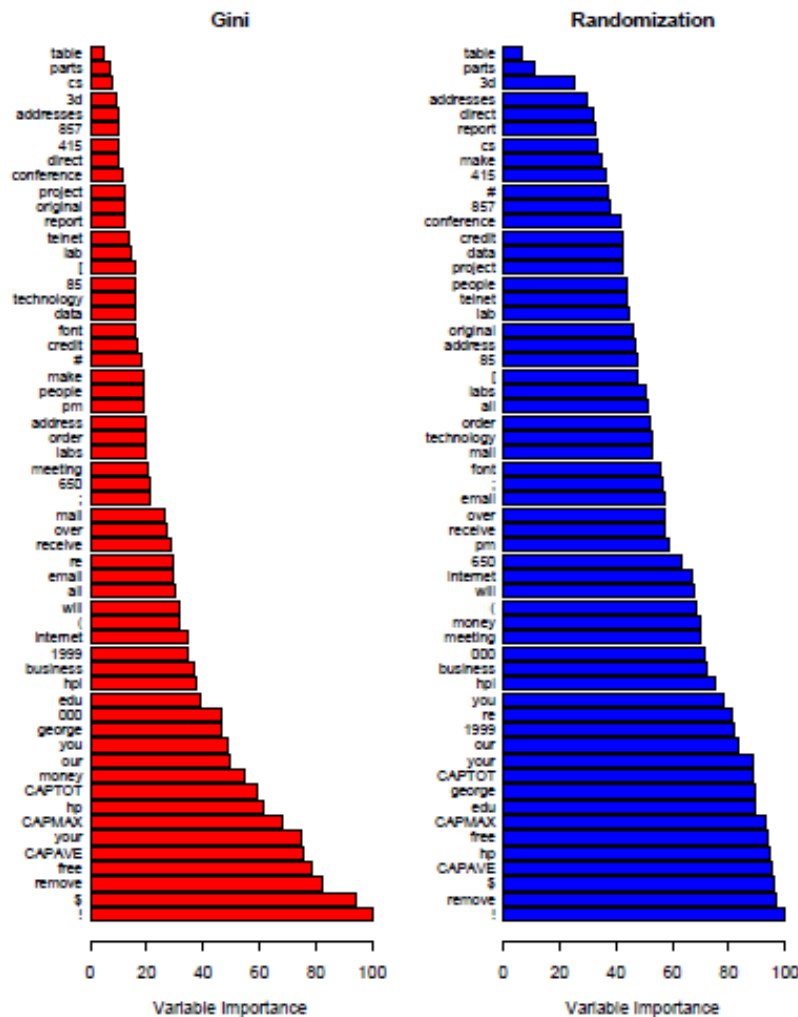
# Variable Importance - 1



# Variable Importance - 2

- Measure split criterion improvement
- Record improvements for each feature
- Accumulate over whole ensemble

# Example: Spam classification

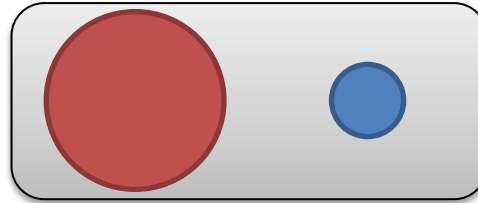


Randomization tends to spread out the variable importance more uniformly.

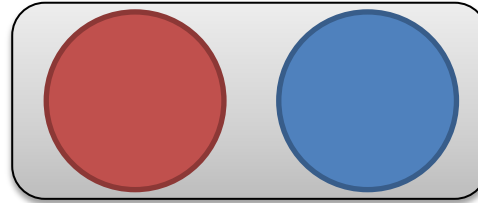


# Unbalanced Classes

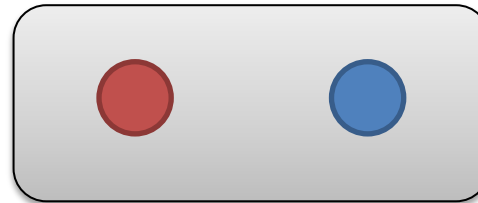
- The Problem:



- Oversample:

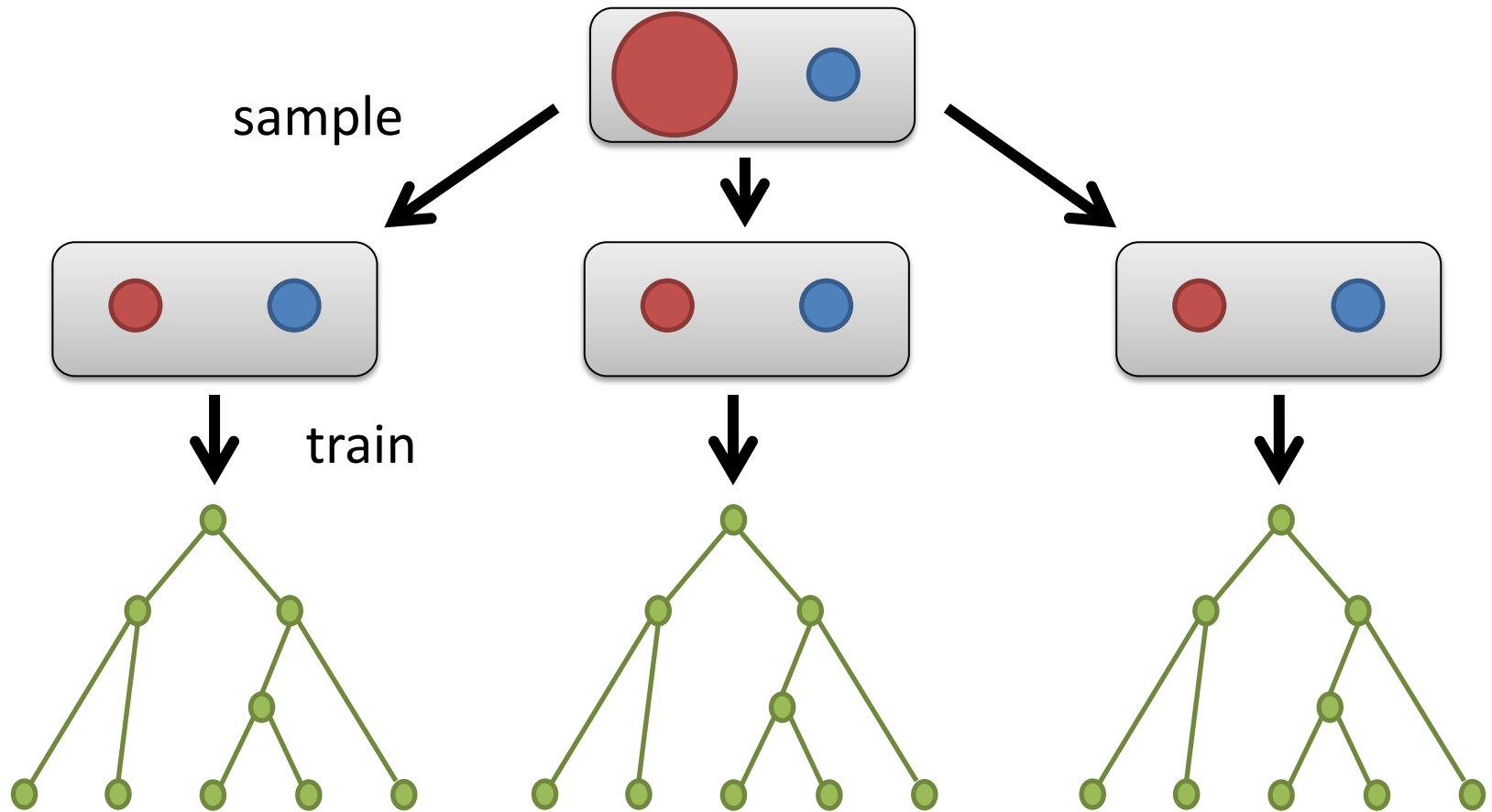


- Subsample:



- Subsample for each tree!

# Random Forest Subsampling



# Random Forest

- Similar to Bagging
- Easy to parallelize
- Packaged with some neat functions:
  - Out of bag error
  - Feature importance measure
  - Proximity estimation