



Department of Computer Science  
UNIVERSITY OF COLORADO **BOULDER**



## Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 20

Slides adapted from Chenhao Tan, Jordan Boyd-Graber, Chris Ketelsen

## Logistics

---

- Project proposal due today

## Learning objectives

---

- Understand the general idea behind ensembling
- Understand bagging
- Intro. to boosting

## Overview

---

Ensemble methods

Bagging and random forest

General idea of boosting

## Outline

---

Ensemble methods

Bagging and random forest

General idea of boosting

## Ensemble methods

---

We have learned

- Decision trees
- Perceptron
- KNN
- Naïve Bayes
- Logistic regression
- Neural networks
- Support vector machines

A meta question: why do we only use a single model?

In practice, especially to win competitions, many models are used together.

## Ensemble methods

---

There are many techniques to use multiple models.

- Bagging
  - Train classifiers on subsets of data
  - Predict based on majority vote
- Boosting
  - Build a sequence of dumb models
  - Modify training data along the way to focus on difficult examples
  - Predict based on weighted majority vote of all the models
- Stacking
  - Take multiple classifiers' outputs as inputs and train another classifier to make final prediction

## Outline

---

Ensemble methods

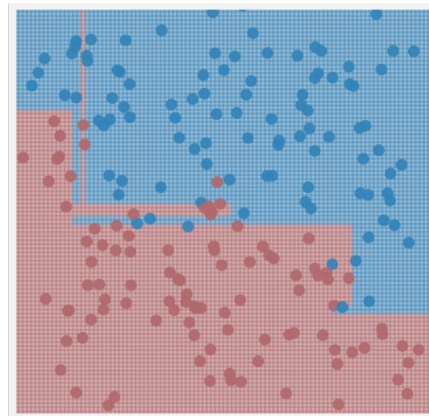
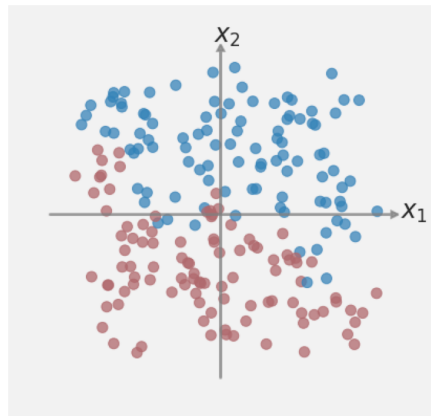
Bagging and random forest

General idea of boosting



## Recap of decision tree

Full decision tree classifiers tend to overfit

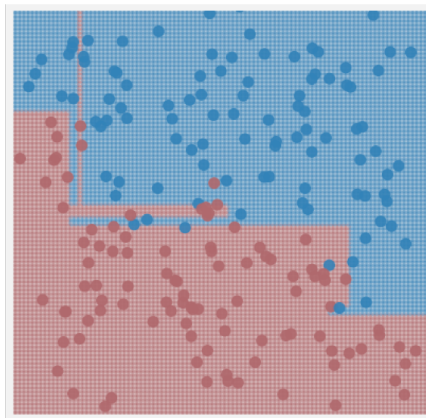


## Recap of decision tree

Full decision tree classifiers tend to overfit

We could alleviate this a bit with pruning.

- Prepruning
  - Don't let the tree have too many levels
  - Stopping conditions
- Postpruning
  - Build the complete tree
  - Go back and remove vertices that don't affect performance too much



## Bagging

---

We will see how we can use many decision trees.

General idea:

- Train numerous slightly different classifiers
- Use each classifier to make a prediction on a test instance
- Predict by taking majority vote from individual classifiers

## Bagging

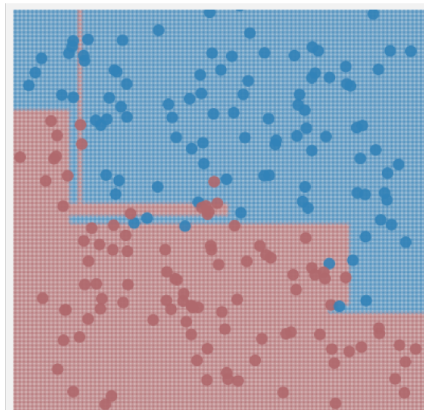
---

Today we will look at two different types of ensembled decision tree classifiers. The simplest is called *bootstrapped aggregation* (or bagging).

## Bagging

Today we will look at two different types of ensembled decision tree classifiers. The simplest is called *bootstrapped aggregation* (or bagging).

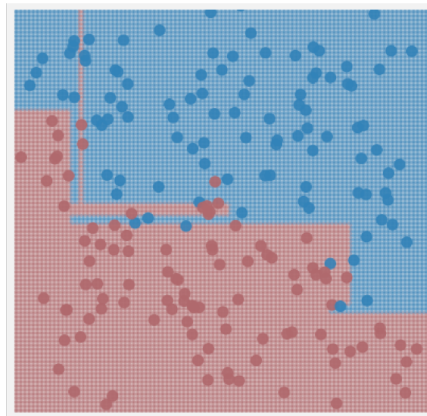
Idea: What would have happened if those training examples that we clearly overfit to weren't there?



## Bagging

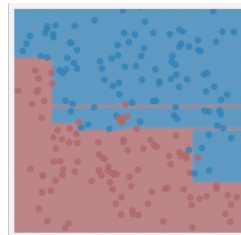
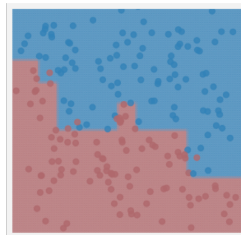
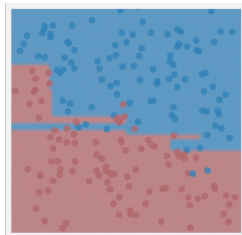
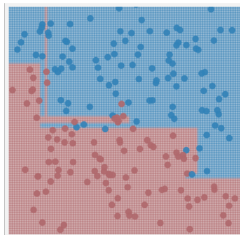
The simplest is called *bootstrapped aggregation* (or bagging).

- Sample  $n$  training examples with replacement
- Fit decision tree classifier to each sample
- Repeat many times
- Aggregate results by majority vote



## Bagging

---



## Bagging

---

In general, assume that  $y \in \{-1, 1\}$  and the prediction of each classifier is  $h_k$   
Then we can define the bagged classifier as

$$H(x) = \text{sign} \sum_{k=1}^K h_k(x)$$



## Bagging Pros and Cons

---

### Pros:

- Results in a much lower variance classifier than a single decision tree

### Cons:

- Results in a much less interpretable model

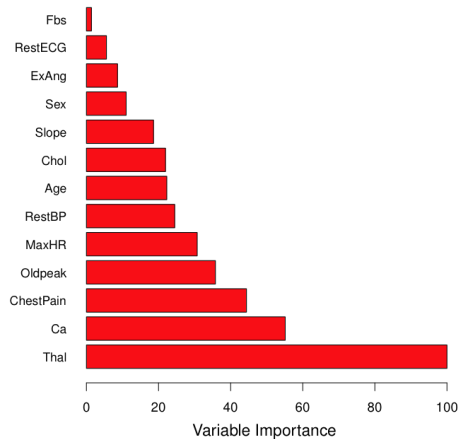
We essentially give up some interpretability (feature importance) in favor of better prediction accuracy.

But we can still get insight into what our model is doing using bagging.

## Bagging and feature importance

We can estimate average feature importance of single features.

Feature importance in a single tree can be defined as information gain achieved by splitting on this feature.



## **An even better approach: random forests**

---

We can actually take the bagging idea and make it even better.

Suppose you have a particular feature that is a very strong predictor for the response.

So strong that in almost all Bagged Decision Trees, it's the first feature that gets split on.

## An even better approach: random forests

We can actually take the bagging idea and make it even better.

Suppose you have a particular feature that is a very strong predictor for the response.

So strong that in almost all Bagged Decision Trees, it's the first feature that gets split on.

When this happens, the trees can all look very similar. They're too correlated.

Always splitting on the same feature isn't the best idea.

## An even better approach: random forests

We can actually take the bagging idea and make it even better.

Suppose you have a particular feature that is a very strong predictor for the response.

So strong that in almost all Bagged Decision Trees, it's the first feature that gets split on.

When this happens, the trees can all look very similar. They're too correlated.

Always splitting on the same feature isn't the best idea.

Like with bootstrapping training examples, we split on a **random subset of features** too.

## Random forests

---

- Still doing bagging, in the sense that we fit bootstrapped resamples of training data
- But every time we have to choose a feature to split on, don't consider all  $p$  features
- Instead, consider only  $\sqrt{p}$  features to split on

## Random forests

---

- Still doing bagging, in the sense that we fit bootstrapped resamples of training data
- But every time we have to choose a feature to split on, don't consider all  $p$  features
- Instead, consider only  $\sqrt{p}$  features to split on

### Advantages:

- Since at each split, considering less than half of features, this gives very different trees
- Very different trees in your ensemble decreases correlation, and gives more robust results
- Also, slightly cheaper because you don't have to evaluate each feature to choose best split

## Random forests

Prediction 15 cancer states given 500 different genes in tissue samples.

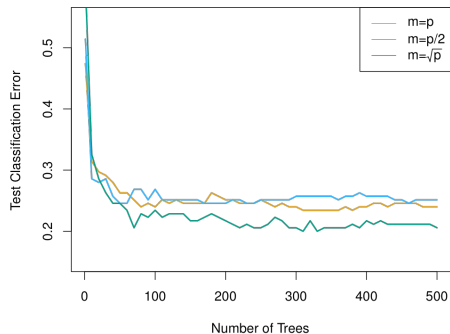


Image from [James et al.]



## Outline

---

Ensemble methods

Bagging and random forest

General idea of boosting

## Boosting intuition

---

Boosting is an ensemble method, but with a different twist.

Idea:

- Build a sequence of dumb models
- Modify training data along the way to focus on difficult to classify examples
- Predict based on weighted majority vote of all the models

Challenges:

- What do we mean by dumb?
- How do we promote difficult examples?
- Which models get more say in vote?

## Boosting intuition

---

What do we mean by dumb?

Each model in our sequence will be a **weak learner**

$$\text{err} = \frac{1}{2} - \gamma$$

where  $\gamma > 0$  is a small constant

Most common weak learner in Boosting is a decision stump - a decision tree with a single split

## Boosting intuition

---

How do we promote difficult examples?

After each iteration,

- increase the importance of training examples that we got wrong on the previous iteration
- decrease the importance of examples that we got right on the previous iteration

Each example will carry around a weight  $w_i$  that will play into the decision stump and the error estimation

Weights are normalized so they act like a probability distribution

$$\sum_{i=1}^m w_i = 1$$

## Boosting intuition

---

Which models get more say in vote?

The models that performed better on training data get more say in the vote

For our sequence of weak learners:  $h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_K(\mathbf{x})$

Boosted classifier defined by

$$H(\mathbf{x}) = \text{sign} \left[ \sum_{k=1}^K \alpha_k h_k(\mathbf{x}) \right]$$

Weight  $\alpha_k$  is measure of accuracy of  $h_k$  on training data