S&DS 355 / 555

Introductory Machine Learning

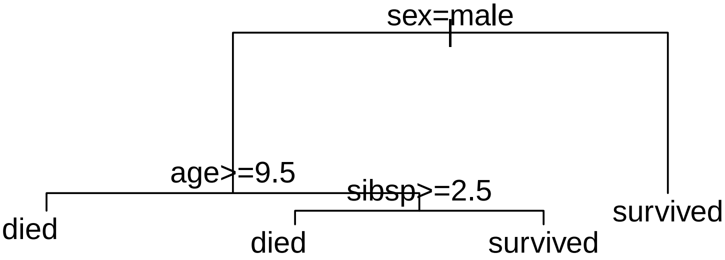
Ensemble Methods (with Trees)

Tuesday, September 24th

Prof. Elisa Celis

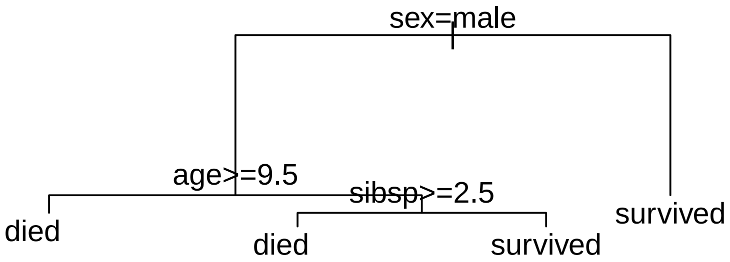
Yale

Recap: Trees



The CART (Classification And Regression Tree) algorithm works by Binary Recursive Partitioning.

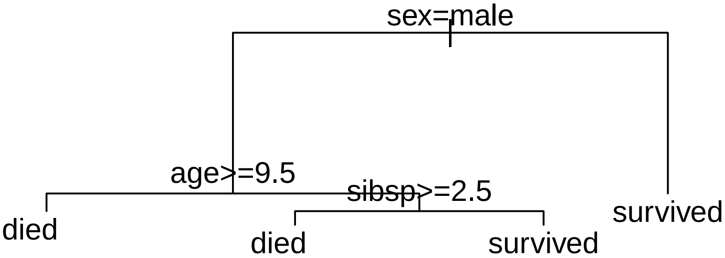
Recap: Trees



The CART (Classification And Regression Tree) algorithm works by Binary Recursive Partitioning.

- Binary (split)

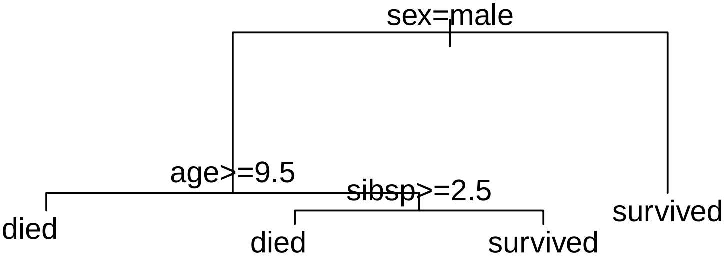
Recap: Trees



The CART (Classification And Regression Tree) algorithm works by Binary Recursive Partitioning.

- Binary (split)
- Partitioning (via RSS or node impurity)

Recap: Trees



The CART (Classification And Regression Tree) algorithm works by Binary Recursive Partitioning.

- Binary (split)
- Partitioning (via RSS or node impurity)
- Recursive (greedy)

Trees vs. kNN

Decision trees are similar in spirit to k-nearest neighbors.

Trees vs. kNN

Decision trees are similar in spirit to k-nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.

Trees vs. kNN

Decision trees are similar in spirit to k-nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.
- However, decision trees use adaptive neighborhoods.

Trees vs. Linear Regression

Linear regression fits models of the form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Trees vs. Linear Regression

Linear regression fits models of the form:

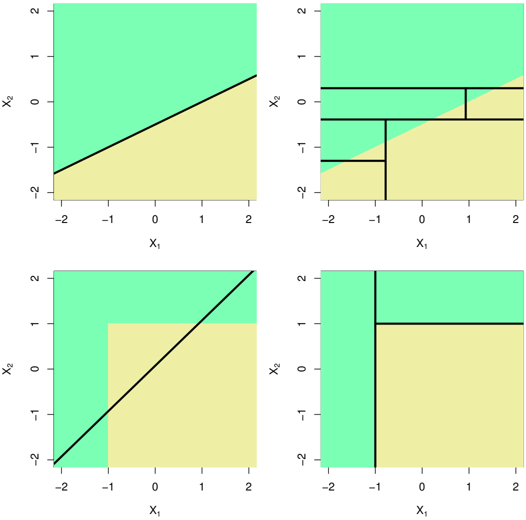
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Regression trees are like fitting linear regression models with a bunch of indicators!

$$f(X) = \sum_{j=1}^{J} \beta_j \mathbb{1}\left\{X \in R_j\right\}$$

Trees vs. Linear Regression

Are trees better?



Trees Summary

- prediction rules easy to interpret
- trees are sensitive to underlying data
- prediction accuracy can be so-so

Trees Summary

- prediction rules easy to interpret
- trees are sensitive to underlying data
- prediction accuracy can be so-so
  - Deep trees have low bias, but suffer from high variance.
  - Shallow trees have low variance, but suffer from high bias.
  - One remedy is to adjust stopping criteria and/or prune a deep tree.

Trees Summary

- prediction rules easy to interpret
- trees are sensitive to underlying data
- prediction accuracy can be so-so
  - Deep trees have low bias, but suffer from high variance.
  - Shallow trees have low variance, but suffer from high bias.
  - One remedy is to adjust stopping criteria and/or prune a deep tree.

Today, we consider a different approach, through the use of ensembles.

Trees Summary

- prediction rules easy to interpret
- trees are sensitive to underlying data
- prediction accuracy can be so-so
  - Deep trees have low bias, but suffer from high variance.
  - Shallow trees have low variance, but suffer from high bias.
  - One remedy is to adjust stopping criteria and/or prune a deep tree.

Today, we consider a different approach, through the use of ensembles.

This approach will improve accuracy and sensitivity, but will lose with respect to interpretability.

Ensemble: Intuition

An analogy is the "**Wisdom of the Crowds**".

Ensemble: Intuition

An analogy is the "**Wisdom of the Crowds**".

- Tree — One Person
- Forest — Crowd

Ensemble: Intuition

An analogy is the “**Wisdom of the Crowds**”.

- Tree — One Person
- Forest — Crowd

Goal: by combining multiple models the right way we can obtain more accurate and/or robust models.

Ensemble: Intuition

An analogy is the "**Wisdom of the Crowds**".

- Tree — One Person
- Forest — Crowd

Goal: by combining multiple models the right way we can obtain more accurate and/or robust models.

Approach: Train many trees to build a "forest" and use them to collectively come to a decision.

Ensemble: Intuition

More precisely, create B trees to get B predictors $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$.

Ensemble: Intuition

More precisely, create B trees to get B predictors $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$.

Each tree is called a **base model** (or weak learner).

Ensemble: Intuition

More precisely, create B trees to get B predictors $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$.

Each tree is called a **base model** (or weak learner).

Regression: For datapoint x, we take an average $\widehat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}^{*b}(x)$.

Ensemble: Intuition

More precisely, create B trees to get B predictors $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$.

Each tree is called a **base model** (or weak learner).

Regression: For datapoint x, we take an average $\widehat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}^{*b}(x)$.

Classification: For datapoint x, take the majority vote $\widehat{f}_{bag}(x) = maj_{b=1}^{B}\{f^{*b}(x)\}$.

Ensemble: Intuition

More precisely, create B trees to get B predictors $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$.

Each tree is called a **base model** (or weak learner).

Regression: For datapoint x, we take an average $\widehat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}^{*b}(x)$.

Classification: For datapoint x, take the majority vote $\widehat{f}_{bag}(x) = maj_{b=1}^{B}\{f^{*b}(x)\}$.

Why does this help?

Ensemble: Intuition

Regression: Intuitively, given $Y_1, Y_2, \ldots,$ *iid* with mean μ and variance σ^2:

- suppose we want to estimate μ

Ensemble: Intuition

Regression: Intuitively, given $Y_1, Y_2, \ldots,$ *iid* with mean μ and variance σ^2:

- suppose we want to estimate μ
- consider estimators Y_1 and $\bar{Y}$ (both unbiased)

Ensemble: Intuition

Regression: Intuitively, given $Y_1, Y_2, \ldots,$ *iid* with mean μ and variance σ^2:

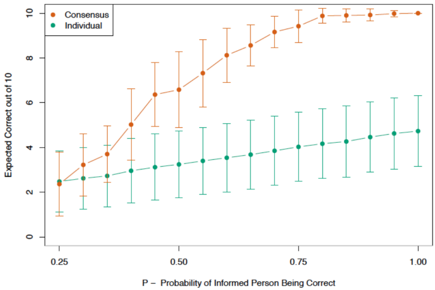
- suppose we want to estimate μ
- consider estimators Y_1 and $\bar{Y}$ (both unbiased)
- then $Var(\bar{Y}) = \sigma^2/n < Var(Y_1)$

Ensemble: Intuition

Regression: Intuitively, given $Y_1, Y_2, \ldots,$ *iid* with mean μ and variance σ^2:

- suppose we want to estimate μ
- consider estimators Y_1 and $\bar{Y}$ (both unbiased)
- then $Var(\bar{Y}) = \sigma^2/n < Var(Y_1)$

Classification: Similarly, we can improve predictions by aggregating multiple iid predictors:



Ensemble: Intuition

It is not enough to just have a crowd (forest).

Ensemble: Intuition

It is not enough to just have a crowd (forest).

If everyone thinks the same, then you get groupthink and the value of a crowd is the same as one person.

Ensemble: Intuition

It is not enough to just have a crowd (forest).

If everyone thinks the same, then you get groupthink and the value of a crowd is the same as one person.

The people's opinions in the crowd should be **uncorrelated.**

Ensemble: Intuition

It is not enough to just have a crowd (forest).

If everyone thinks the same, then you get groupthink and the value of a crowd is the same as one person.

The people's opinions in the crowd should be **uncorrelated.**

In particular, if we train many trees the way we know how, all end up the same.

Ensemble methods

Ensemble methods pool together multiple different models to arrive at more reliable predictions.

Ensemble methods

Ensemble methods pool together multiple different models to arrive at more reliable predictions.

To ensure the models are different, we will train each one slightly differently:

Ensemble methods

Ensemble methods pool together multiple different models to arrive at more reliable predictions.

To ensure the models are different, we will train each one slightly differently:

- **b**ootstrap **agg**regation (bagging): randomizes training data
- random forests (feature bagging): randomizes training data + randomizes features
- boosting: changes/weights training data

Ensemble methods

Ensemble methods pool together multiple different models to arrive at more reliable predictions.

To ensure the models are different, we will train each one slightly differently:

- **b**ootstrap **agg**regation (bagging): randomizes training data
- random forests (feature bagging): randomizes training data + randomizes features
- boosting: changes/weights training data

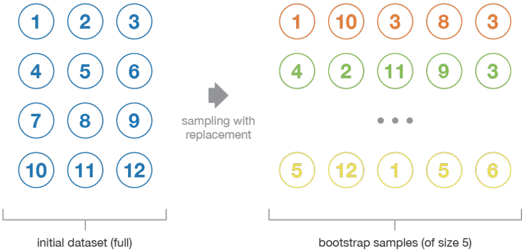
These techniques are **general** and can be applied to other models – today we focus on trees.

Ensemble Method 1: Bagging

To start, we first create different training data sets via **random sampling with replacement.**

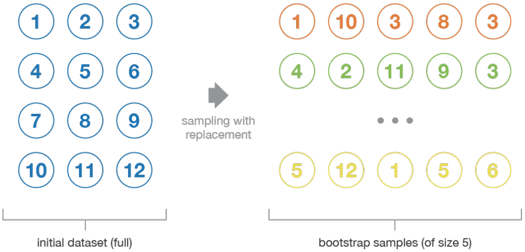
Ensemble Method 1: Bagging

To start, we first create different training data sets via **random sampling with replacement.**



Ensemble Method 1: Bagging

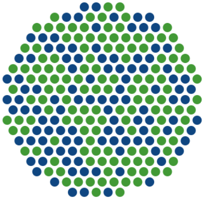
To start, we first create different training data sets via **random sampling with replacement**.



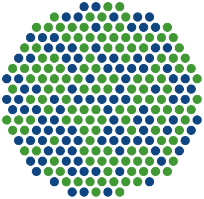
This is called **bootstrap sampling**, and each sample is sometimes called a **bag**.

Ensemble Method 1: Bagging

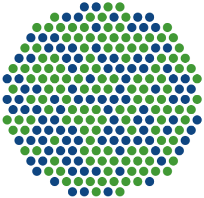
We can now train a decision tree on each sample.



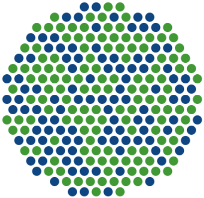
Training Set



Training Set



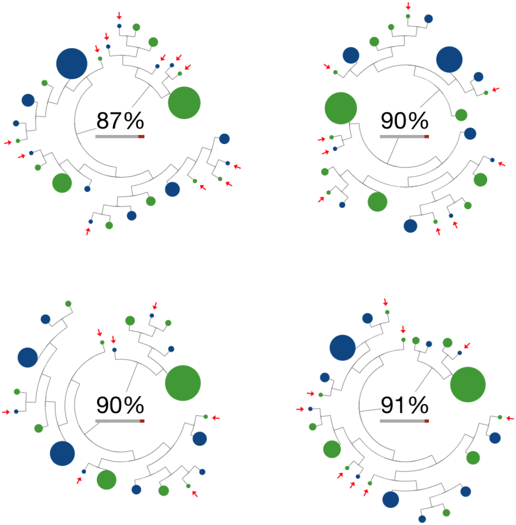
Training Set



Training Set

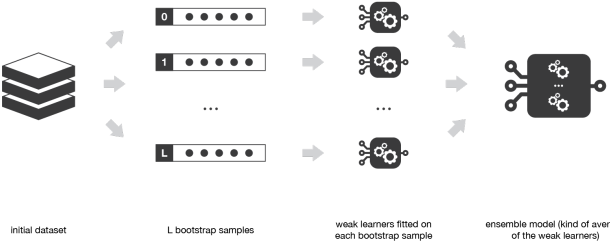
Ensemble Method 1: Bagging

We can now train a decision tree on each sample.



Ensemble Method 1: Bagging

Overall, the process looks like this:



Ensemble Method 1: Bagging

In general, if our original training set is of size n, we sample n points (with replacement).

Ensemble Method 1: Bagging

In general, if our original training set is of size n, we sample n points (with replacement).

Each bagged tree uses about 2/3 of all observations:

Ensemble Method 1: Bagging

In general, if our original training set is of size n, we sample n points (with replacement).

Each bagged tree uses about 2/3 of all observations:

$$1 - \left(1 - \frac{1}{n}\right)^n \to 1 - \frac{1}{e} \approx 0.6321$$

Ensemble Method 1: Bagging

In general, if our original training set is of size n, we sample n points (with replacement).

Each bagged tree uses about 2/3 of all observations:

$$1 - \left(1 - \frac{1}{n}\right)^n \to 1 - \frac{1}{e} \approx 0.6321$$

The remaining data (*out-of-bag* (OOB) observations) can be put to good use.

Out-of-Bag error estimation

	Bagging iteration					OOB Est.
Obs	1	2	3	...	B	
1	OOB	train	train	...	train	$\widehat{y}_1$
2	train	OOB	train	...	train	$\widehat{y}_2$
3	train	train	OOB	...	train	$\widehat{y}_3$
4	OOB	train	train	...	OOB	$\widehat{y}_4$
...	...	...	...	...	...	...
n	train	train	OOB	...	train	$\widehat{y}_n$

Out-of-Bag error estimation

	Bagging iteration					OOB Est.
Obs	1	2	3	...	B	
1	OOB	train	train	...	train	$\widehat{y}_1$
2	train	OOB	train	...	train	$\widehat{y}_2$
3	train	train	OOB	...	train	$\widehat{y}_3$
4	OOB	train	train	...	OOB	$\widehat{y}_4$
...	...	...	...	...	...	...
n	train	train	OOB	...	train	$\widehat{y}_n$

- For each training point, make predictions using each model for which it was OOB.
- Aggregate over all models arrive at an OOB prediction $\widehat{y}_i$.
- Compute prediction error for ensemble by taking the average of the individual OOB prediction errors $\widehat{y}_1, \ldots, \widehat{y}_n$.

Out-of-Bag error estimation

For large B, OOB error is essentially LOOCV error.

In effect, cross-validation can be performed "along the way".

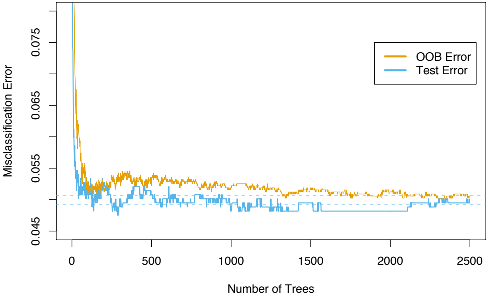


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

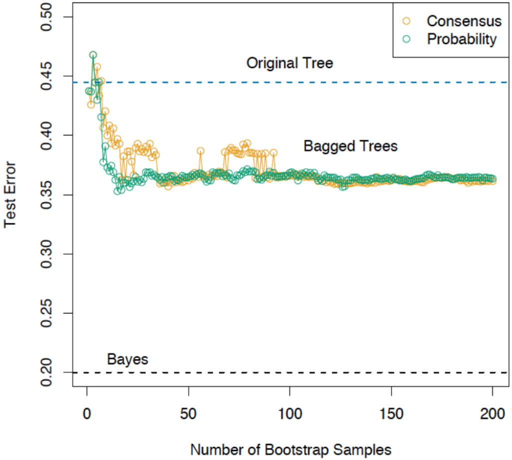
Bagging Accuracy

Given our initial intuition, the more trees we get, the more accurate we should get, until we reach perfect prediction.

Bagging Accuracy

Given our initial intuition, the more trees we get, the more accurate we should get, until we reach perfect prediction.

In reality:



Ensemble Method 2: Feature Bagging

Problem: Training data is not iid.

Ensemble Method 2: Feature Bagging

Problem: Training data is not iid.

Feature bagging attempts to de-correlate base models further by randomizing available features.

Ensemble Method 2: Feature Bagging

Problem: Training data is not iid.

Feature bagging attempts to de-correlate base models further by randomizing available features.

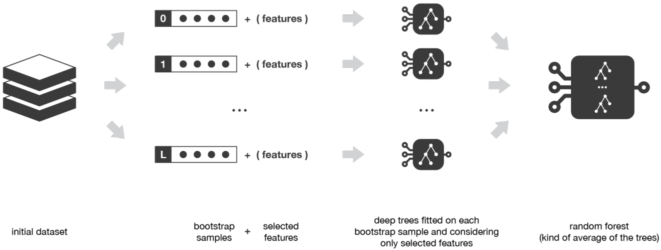
A random subset of m features is used to build each predictor (used in conjunction with "usual" bagging).

Ensemble Method 2: Feature Bagging

Problem: Training data is not iid.

Feature bagging attempts to de-correlate base models further by randomizing available features.

A random subset of *m* features is used to build each predictor (used in conjunction with “usual” bagging).

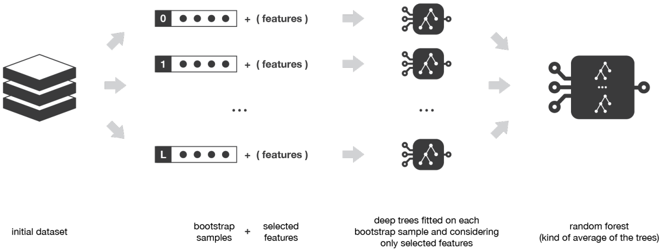


Ensemble Method 2: Feature Bagging

Problem: Training data is not iid.

Feature bagging attempts to de-correlate base models further by randomizing available features.

A random subset of *m* features is used to build each predictor (used in conjunction with “usual” bagging).



Particularily useful if a small set of features is highly predictive.

Random Forest

Takes bagging + feature bagging even further in a tree-specific manner!

Random Forest

Takes bagging + feature bagging even further in a tree-specific manner!

Randomly select m out of p predictors to use **at each split.**

Random Forest

Takes bagging + feature bagging even further in a tree-specific manner!

Randomly select m out of p predictors to use **at each split.**

Rules of thumb for p features:

- Classification: sample $m = \sqrt{p}$ features at each split, with min node size 1.
- Regression: sample $m = p/3$ features at each split with a min node size of 5.

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data
   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data

   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:

      i. Select m variables at random from the p variables

      ii. Pick the best variable/split-point among the m

      iii. Split the node into two children nodes

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data

   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:

   i. Select m variables at random from the p variables

   ii. Pick the best variable/split-point among the m

   iii. Split the node into two children nodes

2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data

   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:

      i. Select m variables at random from the p variables

      ii. Pick the best variable/split-point among the m

      iii. Split the node into two children nodes

2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

To make a prediction at a new point x:
Regression: Average $\widehat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
Classification: Majority vote of the individual trees

Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data

   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:

      i. Select m variables at random from the p variables

      ii. Pick the best variable/split-point among the m

      iii. Split the node into two children nodes

2. Output the ensemble of trees $\{T_b\}_{b=1}^B$.

To make a prediction at a new point x:
Regression: Average $\widehat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$
Classification: Majority vote of the individual trees

As before, can use out-of-bag (OOB) samples for error estimation.

Bagging and Random Forests Recap

- Grow many trees and average their predictions
- Trees are grown deep, to have low bias, but high variance
- To “decorrelate” the predictors, each tree is
  - bagging: grown on a bootstrap sample of the data
  - random forests: also grown with random subsets of the predictors at each split
- Note: tree growing can be done in parallel!

Bagging and Random Forests Recap

- Grow many trees and average their predictions
- Trees are grown deep, to have low bias, but high variance
- To "decorrelate" the predictors, each tree is
  - bagging: grown on a bootstrap sample of the data
  - random forests: also grown with random subsets of the predictors at each split
- Note: tree growing can be done in parallel!

What if training deep trees is very costly?

Ensemble Method 3: Boosting

Ensemble Method 3: Boosting

- Works with shallow trees (even stumps) which instead have high bias / low variance.

Ensemble Method 3: Boosting

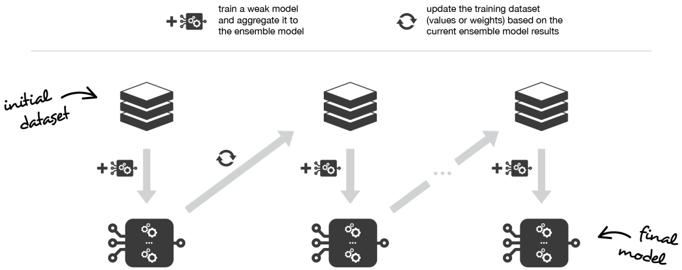
- Works with shallow trees (even stumps) which instead have high bias / low variance.
- Attempts to combat bias by reweighting training data and/or adjusting the training goal to learn from “hard” examples.

Ensemble Method 3: Boosting

- Works with shallow trees (even stumps) which instead have high bias / low variance.
- Attempts to combat bias by reweighting training data and/or adjusting the training goal to learn from “hard” examples.
- Base models have to be trained sequentially.

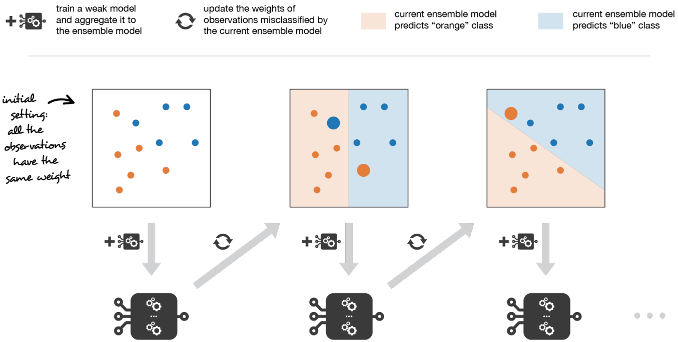
Ensemble Method 3: Boosting

- Works with shallow trees (even stumps) which instead have high bias / low variance.
- Attempts to combat bias by reweighting training data and/or adjusting the training goal to learn from “hard” examples.
- Base models have to be trained sequentially.

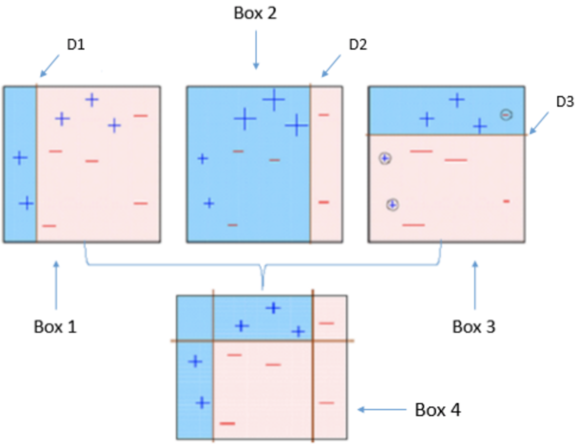


Ensemble Method 3: Boosting

- Works with shallow trees (even stumps) which instead have high bias / low variance.
- Attempts to combat bias by reweighting training data and/or adjusting the training goal to learn from "hard" examples.
- Base models have to be trained sequentially.



Ensemble Method 3: Boosting



Ensemble Method 3: Boosting

There are many ways to boost... details beyond the scope of this class.

Ensemble Method 3: Boosting

There are many ways to boost... details beyond the scope of this class.

Two common methods:

- adaBoost
- gradient boosting

Ensemble methods: Recap

- Bagging reduces variance without increasing bias by averaging base model predictions trained on random subsets of the training data.

Ensemble methods: Recap

- Bagging reduces variance without increasing bias by averaging base model predictions trained on random subsets of the training data.
- Feature bagging (e.g., for random forests) decorrelates base models by randomly sampling features.

Ensemble methods: Recap

- Bagging reduces variance without increasing bias by averaging base model predictions trained on random subsets of the training data.
- Feature bagging (e.g., for random forests) decorrelates base models by randomly sampling features.
- Boosting methods work to reduce bias by chaning the impotance placed on different training datapoints.

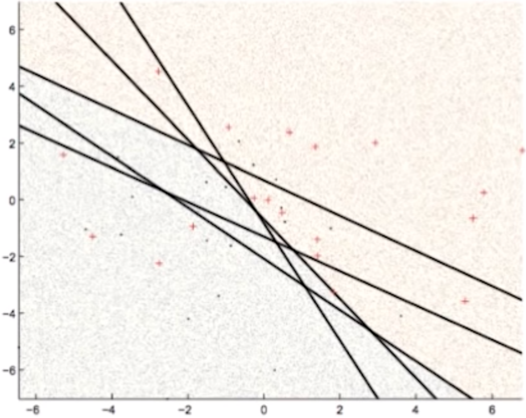
Ensemble methods: Recap

- Bagging reduces variance without increasing bias by averaging base model predictions trained on random subsets of the training data.
- Feature bagging (e.g., for random forests) decorrelates base models by randomly sampling features.
- Boosting methods work to reduce bias by chaning the impotance placed on different training datapoints.

Ensemble learning is a **general** machine learning paradigm, can be used with non-tree base models.

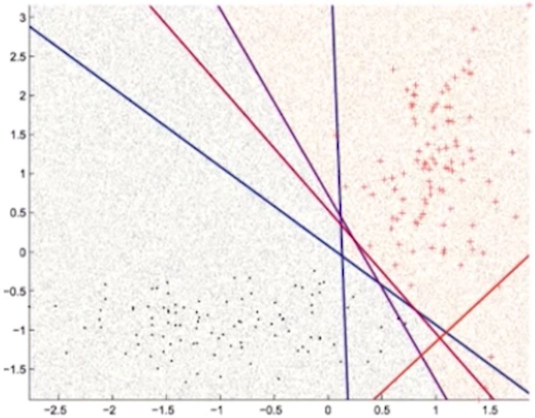
Ensemble methods: Linear Regression

Ensemble learning is a **general** machine learning paradigm, can be used with non-tree base models.



Ensemble methods: Linear Regression

Ensemble learning is a **general** machine learning paradigm, can be used with non-tree base models.



Ensemble methods: Pros & Cons

1. Bagging:
   - Advantages: Reduces variance / avoids overfitting. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive. Not be desirable when base models still highly correlated.

Ensemble methods: Pros & Cons

1. Bagging:
   - Advantages: Reduces variance / avoids overfitting. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive. Not be desirable when base models still highly correlated.
2. Feature bagging:
   - Advantages: Reduces variance / avoids overfitting. Better at de-correlating base models. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive.

Ensemble methods: Pros & Cons

1. Bagging:
   - Advantages: Reduces variance / avoids overfitting. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive. Not be desirable when base models still highly correlated.
2. Feature bagging:
   - Advantages: Reduces variance / avoids overfitting. Better at de-correlating base models. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive.
3. Boosting:
   - Advantages: Reduces bias. Training base models is fast.
   - Disadvantages: Not as effective against overfitting. Has to be done sequentially (may be more costly overall).