Ensembles: Bagging and Random Forests

Objectives

Morning

- What is an **Ensemble Model**?
  - Why is it good?
- What is **Bagging**? What are bagged trees?
  - How do you do it?
- What are **Random Forests**?
  - How do you make them? Why might they be an improvement over bagging?
- Individual Assignment
  - Implement bagging
  - Implement random feature selection

What is an Ensemble Model?

An **Ensemble Model** combines many *"weak"* learners to form a strong learner

- Basic Idea: **Wisdom of the Crowd**

Wisdom of the Crowd

Suppose you are trying to predict an election

- You have 5 experts
  - Each expert was picked *independently* of the each other
  - Each expert has a 70% chance of being right
- How could you leverage expert picks to improve accuracy; how often would you be right?
- How many independent predictors would you need to have a 99.9% probability of being correct?

Wisdom of the Crowd

AAAAA $\binom{5}{5}(.7)^5(.3)^0 +$

AAAAD $\binom{5}{4}(.7)^4(.3)^1 +$

AAADD $\binom{5}{3}(.7)^3(.3)^2 =$

.837

Wisdom of the Crowd: Majority Vote

A majority vote of these 5 experts yields an overall accuracy of 83.7%:

$$\binom{5}{3}.7^3.3^2 + \binom{5}{4}.7^4.3^1 + \binom{5}{5}.7^5.3^0 \approx .837$$

With 101 such experts, we can acheive a 99.9% majority vote accuracy

It's clear that utilizing multiple weak classifiers can yield a very strong classifier

What is an Ensemble Model?

An **Ensemble Model** combines many *"weak"* learners to form a strong learner

- Basic Idea: **Wisdom of the Crowd**
  - Run multiple models on the data and aggregate the predictive results to produce an overall prediction that is better than any of the individual models could do on their own
  - The overall prediction is
    - ★ the **average prediction for a regressor** or
    - ★ the **plurality choice for a classifier** (or the average of the percentages of each class)

Wisdom of the Crowd: Majority Vote

A majority vote of these 5 experts yields an overall accuracy of 83.7%:

$$\binom{5}{3}.7^3.3^2 + \binom{5}{4}.7^4.3^1 + \binom{5}{5}.7^5.3^0 \approx .837$$

With 101 such experts, we can acheive a 99.9% majority vote accuracy

It's clear that utilizing multiple weak classifiers can yield a very strong classifier

What's the catch? Why wouldn't this work?

Ensembles: Independence

We could repeatedly sample from the population, building decision tree models and averaging the results. Unfortunately, we only have only sample and if all the learners are the same, creating an ensemble model won't help

- A solution to this is to train each learner on a **different subset of the data**
  - But how do we do this when we only have one set of data to work with?

Bootstrapping to the rescue!

Bootstrapping Review

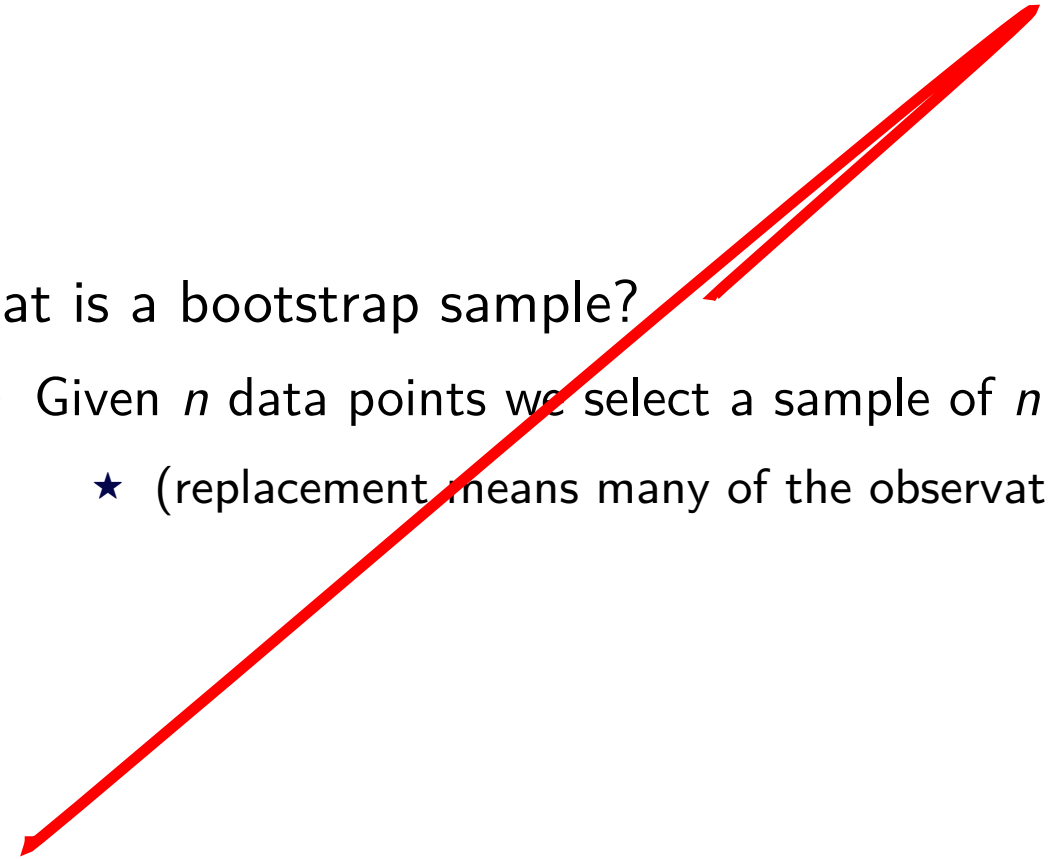
Questions

1. What is a bootstrap sample?
2. What have we learned that bootstrap samples are good for so far?

$\hat{f} = \hat{f}(x_0)$

$E((\hat{f} - f)^2) = \underbrace{E(\hat{f} - f)^2}_{Bias^2} + Var(\hat{f}) + Var(\varepsilon)$

Bootstrapping Review



1. What is a bootstrap sample?
   - Given n data points we select a sample of n points with replacement
     - ★ (replacement means many of the observations will be repeated)

Bootstrapping Review

2. What have we learned that bootstrap samples are good for so far?
   - We used bootstrap samples to construct confidence intervals around sample statistics. E.g., to get a 95% confidence interval around a sample median:
     - ★ take 1,000 bootstrap samples
     - ★ take the mean of each sample
     - ★ The 95% confidence interval for the median is between the 25th and the 975th largest samples

Bootstrapping Review

- Method
  - Start with your dataset of size n
  - Sample from your dataset with replacement to create one bootstrap sample of size n
  - Repeat B times
  - Each bootstrap sample can then be used as a separate dataset for estimation or model fitting
- Advantages
  - Completely automatic
  - Requires no theoretical calculations
  - Not based on asymptotic results
  - Available regardless of how complicated the estimator might be

Bias and Variance Review

Let $\hat{f}^{(b)}(x_0)$ be the b^{th} predictor of outcome $f(x_0)$ constructed for a training set with features $x^{(b)}$ and outcomes $Y^{(b)}$

- Bias
  - Error from failure to match the training set
  - Caused by choosing a model that is too simple

$$Bias(\hat{f}^{(b)}(x_0)) = E[\hat{f}^{(b)}(x_0) - f(x_0)] = E[\hat{f}^{(b)}(x_0)] - f(x_0)$$

- Variance
  - Error due to random noise specific to our training set
  - Decrease as we get more data

$Var[X] = E((X - E(X))^2)$

$$Var(\hat{f}^{(b)}(x_0)) = E\left[\left(\hat{f}^{(b)}(x_0) - E\left[\hat{f}^{(b)}(x_0)\right]\right)^2\right]$$

Very Large "Bushy" Decision Trees

- What's great about them?
  - **Low bias**
- What's not so great about them?
  - **High variance**
- **Very large "bushy" decision trees overfit**
- Bias
  - $\hat{f}^{(b)}(x_0)$ can be unbiased by being highly complex
  - $E\left[\hat{f}^{(b)}(x_0)\right] = f(x_0)$
- Variance
  - We will assume that $Var\left(\hat{f}_{bag}(x_0)\right) = \sigma^2$

Averaging Multiple Predictors

Let $\hat{f}_{bag}(x_0)$ be the averaged predictor of $\hat{f}^{(0)}(x_0), ..., \hat{f}^{(B)}(x_0)$:

$$\hat{f}^{(bag)}(x_0) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)$$

Bias( ) = 0

Var( ) = σ^2

Questions

1. What's its bias $Bias(\hat{f}_{bag}(x_0))$?
2. What's its variance $Var(\hat{f}_{bag}(x_0))$?

Averaging Multiple Predictors

$$\text{Bias}(\hat{f}_{bag}) = E[\hat{f}_{bag}] - f$$

$$E\left[\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}\right] - f$$

$$= E\left[\frac{1}{B}\sum_{b=1}^{B}\left(\hat{f}^{(b)} - f\right)\right]$$

$$= \frac{1}{B}\sum_{b=1}^{B} \underbrace{E\left[\hat{f}^{(b)} - f\right]}_{\text{Bias}(\hat{f}^{(b)}) = 0}$$

Averaging Multiple Predictors

$$Bias(\hat{f}_{bag}(x_0)) = E\left[\hat{f}_{bag}(x_0) - f(x_0)\right]$$

$$= E\left[\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)\right) - f(x_0)\right] = E\left[\frac{1}{B}\sum_{b=1}^{B}\left(\hat{f}^{(b)}(x_0) - f(x_0)\right)\right]$$

$$= \frac{1}{B}\sum_{b=1}^{B} E\left[\hat{f}^{(b)}(x_0) - f(x_0)\right]^{(*)} = \frac{1}{B}\sum_{b=1}^{B} Bias(\hat{f}^{(b)}(x_0)) = 0$$

Therefore $\hat{f}_{bag}(x_0)$ is also an unbiased predictor

$^{(*)} E[aX + bY] = aE[X] + bE[Y]$

Averaging Multiple Predictors

$$\mathrm{Var}\left(\hat{f}_{\text{bag}}\right) = \mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}\right)$$

$$\mathrm{Std}[\bar{x}] = \frac{\mathrm{Std}(x)}{\sqrt{n}} = \frac{1}{B^2}\mathrm{Var}\left(\sum_{b=1}^{B}\hat{f}^{(b)}\right)$$

$$= \frac{1}{B^2} B\sigma^2 = \frac{\sigma^2}{B}$$

$$\mathrm{Var}[aX + bY] = a^2\mathrm{Var}[X] + b^2\mathrm{Var}[Y] + 2ab\,\mathrm{Cov}(X,Y)$$

Averaging Multiple Predictors

$$Var(\hat{f}_{bag}(x_0)) = Var\left[\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)\right]$$

$$= \frac{1}{B^2}\sum_{b=1}^{B} Var(\hat{f}^{(b)}(x_0))^{(**)} = \frac{\sigma^2}{B}$$

So even if $\hat{f}^{(b)}(x_0)$ are high variance predictors, the variance of $\hat{f}_{bag}(x_0)$ decreases with B

$^{(**)}\, Var[aX + bY] = a^2 Var[X] + 2abCov(X, Y) + b^2 Var[Y]$; let's first asssume that the predictors $\hat{f}^{(b)}(x_0)$ are independent from each other, so no covariance terms

Bagging

Bagging$^{(*)}$, or bootstrap aggregation, is a general-purpose procedure for reducing the variance of a statistical learning method

Bagging simply predicts $f(x_0)$ with $\hat{f}_{bag}(x_0) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)$, an average of predictors made from bootstraped samples

1. Bootstrapped trees provides unbiased, high variance predictors
   - Trees are grown deep and are not pruned, hence each individual tree has high variance, but low bias
2. Averaged predictors are still unbiased
3. Averaged estimators are lower variance than single predictors
4. Bootrapping explores the "dataset"/"predictors" space; then uses the average of predictors we might see under sampling

$^{(*)}$ Bagging is a general ensemble procedure often used with trees

Bagging

- The number of trees B is not a critical parameter with bagging; using a very large B won't lead to overfitting
- In practice we want to use a large enough B such that our test error has settled down

Bias vs. Variance

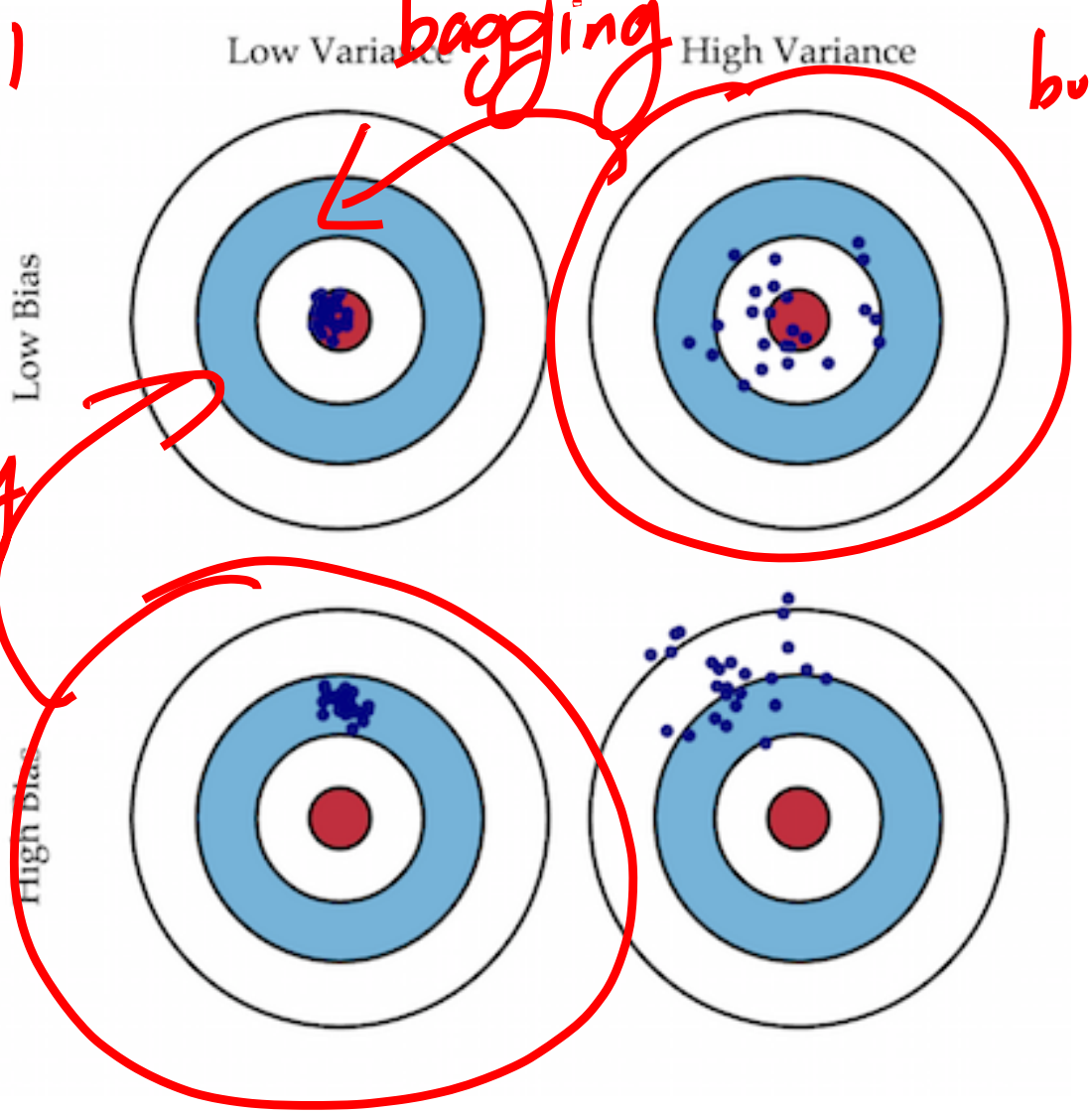


Figure 1:Bias vs. Variance

Bagging

Method

- Create bootstrap samples, fit a model from each bootstrap sample; can use a subset of possible features to build models
- Each model then gets a vote in the final results Final selection based on aggregate predictions
  - majority or weighted vote if classification (i.e., predicting a category)
  - average if regression (i.e., predicting a continuous value)

But... Wait!

Are $\hat{f}^{(i)}(x_0)$ and $\hat{f}^{(j)}(x_0)$ really uncorrelated?

- $\hat{f}^{(i)}$ is fitted on $x^{(i)}$ and $Y^{(i)}$
- $\hat{f}^{(j)}$ is fitted on $x^{(j)}$ and $Y^{(j)}$
- $x^{(i)}$ and $x^{(j)}$ are bootstrapped from the same x
- $Y^{(i)}$ and $Y^{(j)}$ are bootstrapped from the same Y
- So they are very similar samples (indeed, they overlap)
- **So $\hat{f}^{(i)}$ and $\hat{f}^{(j)}$ are expected to be correlated**

How good is bagging?

$$Cor(X,Y) = \frac{Covar(X,Y)}{\sqrt{Var(X)Var(Y)}}$$

Assuming $Cor(\hat{f}^{(i)}(x_0), \hat{f}^{(j)}(x_0)) = \rho \, \forall i, j$, then

$$Cov(\hat{f}^{(i)}(x_0), \hat{f}^{(j)}(x_0)) = \rho\sigma^2$$

How good is bagging?

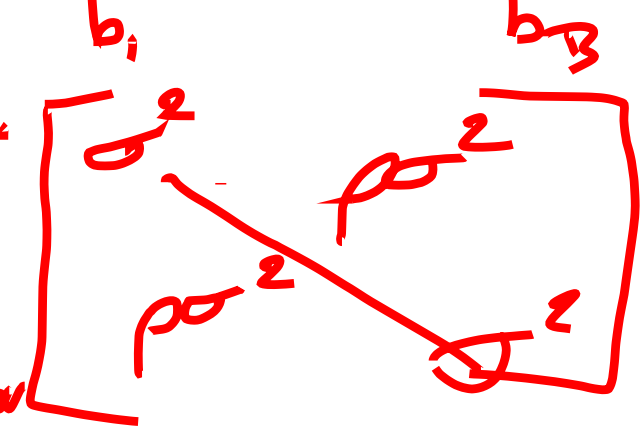
$$\mathrm{Var}\left(\hat{f}_{\text{bag}}\right) = \mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}\right]$$

$$= \frac{1}{B^2}\mathrm{Var}\left(\sum_{b=1}^{B}\hat{f}^{(b)}\right)$$

$$= \frac{1}{B^2}\left(B\sigma^2 + (B^2-B)\rho\sigma^2\right)$$

$$= \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

"fixed"



How good is bagging?

Assuming $Cor(\hat{f}^{(i)}(x_0), \hat{f}^{(j)}(x_0)) = \rho \, \forall i, j$, then

$$Cov(\hat{f}^{(i)}(x_0), \hat{f}^{(j)}(x_0)) = \rho\sigma^2$$

and

$$Var(\hat{f}_{bag}(x_0)) = Var\left[\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)\right]$$

$$= \frac{1}{B^2}Var\left[\sum_{b=1}^{B}\hat{f}^{(b)}(x_0)\right] = \frac{1}{B^2}(B\sigma^{2(*)} + (B^2 - B)\rho\sigma^{2(**)})$$

$$= \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

$^{(*)}$ B variance terms, $^{(**)}$ $B^2 - B$ covariance terms

How good is bagging?

$$Var(\hat{f}_{bag}(x_0)) = \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

- Only the uncorrelated parts of $\hat{f}^{(b)}(x_0)$ benefit from the "CLT" effect
- So, is there a way to get ρ close to 0?
  - We might try to decorrelate $\hat{f}_{(b)}(x_0)$. But how are $\hat{f}_{(b)}(x_0)$ correlated?

Decorrelating Trees

Trees are "correlated" because:

- Bootstrap samples are about the same
  - Nothing to do there
- Infuential features tend to be same
  - This we can infuence...

How can we make constructed tree not exactly the same?

Random Forests

- **Random Forests** provide an improvement over bagged trees by way of a small tweak that further decorrelates the trees
  - Both methods utilize bootstrap samples
  - However, random forests also randomly selects features
- Intuition
  - With bagged trees, a very strong predictor will often/always show up in the top split so all trees will look quite similar and highly correlated
  - With random forests, the very strong predictor will be included only a portion of the time, allowing moderately strong predictors to have a chance

Random Forests

- Trees are constructed by recursive best splits
- At each split, we only use a random subset of features
  - Random exclusion only per split (features are reincludable later)
  - This technique, called subset sampling, serves to prevent trees from always choosing the same (best) splits
- More features means stronger but also more correlated trees. . .

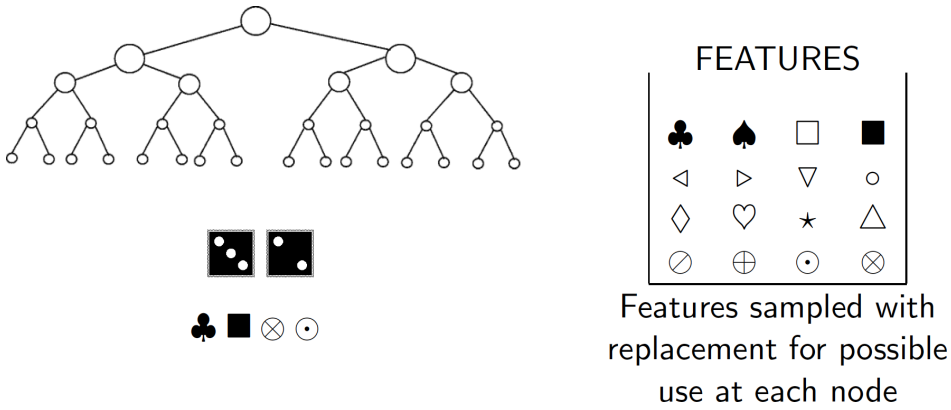


Figure 2:Random subset of features

Random Forest Algorithm

- For $b = 1, ..., B$:
  - Draw a bootstrap sample of size N from the training data
  - Grow a random-forest tree to the bootstrapped data by recursively repeating the following for each terminal node of the tree, until minimum node size is reached
    - ★ Picking a subset of features (m) from the p features
    - ★ Pick the best variable/split-point among the m variables
    - ★ Split the node into two child nodes
- Output the ensemble of trees

Tuning and Performance

Tuning

- Typically select $m = \sqrt{p}$ (classification) or $m = \frac{p}{3}$ features
- In reality, m is an hyperparameter that we can tune

Performance

- **Random forests are often robust to tree characteristics**
  - **Very large numbers of "bushy" trees are typically used and approach state of the art performance out of the box**

Random Forest Parameters

- Total number of trees (`n_estimators`)
- Number of features to consider at each split (`max_features`)
- Number of points for each bootstrap sample
- Individual decision tree parameters
  - E.g., tree depth (`max_depth`), pruning (`min_samples_split`, `min_samples_leaf`)

How many trees?

- Variance decreases with more trees
  - But with diminishing returns
- Runtime scales linearly with the number of trees
- More is still better, but wait until the end to fit forests with 100,000 trees

Pros and Cons of Random Forests

Pros

- Often give near state-of-the-art performance
- Good out-of-the-box performance
- No feature scaling needed
- Model nonlinear relationships

Cons

- Can be expensive to train (though can be done in parallel)
- Models can be quite large (the pickled version of a several hundred tree model can easily be several GBs)
- Not interpretable (although techniques such as predicted value plots can help)