Ensemble methods

Fraida Fund

Contents

Ensemble methods

Recap: decision trees

- Let trees grow deep - low bias, high variance
- Don't let trees get deep: low variance, high bias

Ensemble methods - the idea

Combine multiple **weak learners** - having either high bias or high variance - to create an **ensemble** with better prediction

Ensemble methods - types (1)

- Combine multiple learners with high **variance** in a way that reduces their variance
- Combine multiple learners with high **bias** in a way that reduces their bias

Ensemble methods - types (2)

- **Averaging methods**: build base estimators *independently* and then average their predictions. Combined estimator is usually better than any single base estimator because its *variance* is reduced.
- **Boosting methods**: build base estimators *sequentially* and each one tries to reduce the *bias* of the combined estimator.

Bagging

Bagging - background

- Designed for, and most often applied to, decision trees
- Name comes from **bootstrap aggregation** (remember bootstrap from resampling lecture?)

Bootstrapping

- Basic idea: Sampling **with replacement**
- Each "bootstrap training set" is *same size* as full training set, and is created by sampling with replacement
- Some samples will appear more than once, some samples not at all

Bootstrap aggregation

- Create multiple versions $1, \ldots, B$ of training set with bootstrap
- Independently train a model on each bootstrap training set: calculate $\hat{f}^1(x) \ldots, \hat{f}^B(x)$
- Combine output of models by voting (classification) or averaging (regression):

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

Bagging trees

- Construct B trees using B bootstrapped training sets.
- Let the trees grow deep, no pruning.
- Each individual tree has low bias, high variance.
- Average the prediction of the trees to reduce variance.

Bagging - example

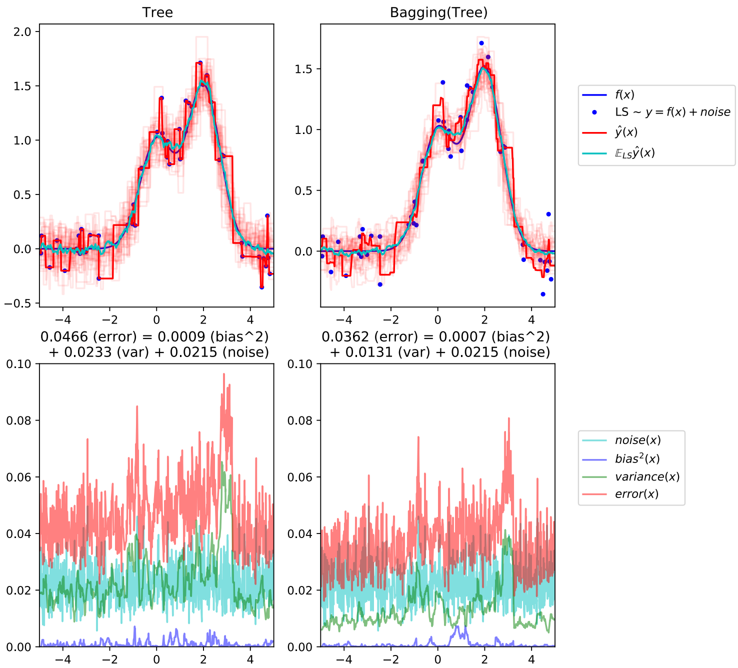


Figure 1: Bagging: average the output of many deep trees, reduce variance of estimate

Out-of-bag error estimation

- One average, each bag contains about 2/3 of training samples. Remaining samples are called out-of-bag (OOB) observations.
- For each observation i: predict response for the $\frac{B}{3}$ trees where i was OOB, and average (or take majority vote).
- Compute OOB MSE or OOB classification error as mean of OOB errors for all samples.
- Use OOB error to find the number of trees needed to stablize error rate.

Correlated trees

Problem: trees produced by bagging are highly correlated.

- Imagine there is one feature that is strong predictor, several moderate predictors
- Most/all trees will split on this feature
- Averaging correlated quantities does not reduce variance as much.

Random forests

Grow many decorrelated trees:

- **Bootstrap**: grow each tree with bootstrap resampled data set.
- **Split-variable randomization**: Force each split to consider *only* a subset of m of the p predictors.

Typically $m = \frac{p}{3}$ but this should be considered a tuning parameter (tune using OOB error).

A note on computation

- Bagged trees and random forests can be fitted in parallel on many cores!
- Each tree is built independently of the others

Boosting

Boosting - training

Iteratively build a succession of models:

- Train a weak model. Typically a very shallow tree (1-6 splits).
- In training set for bth model, focus on errors made by $b-1$th model.
- Use (weighted) model output
- Reduces bias *and* variance!

AdaBoost (Adaptive Boosting)

Adjust *weights* so that each successive model focuses on more "difficult" samples.

Consider classification problem, where sign of model output gives estimated class label and magnitude gives confidence in label.

AdaBoost algorithm

1. Let $w_i = \frac{1}{N}$ for all i in training set.
2. For $m = 1, \dots, M$, repeat:

AdaBoost algorithm (inner loop)

- Fit a tree $\hat{f}^m$ and compute weighted error err_m (e.g. classification error) using weights on training samples w_i:

$$err_m = \frac{\sum_{i=1}^N w_i 1(y_i \neq \hat{f}^m(x_i))}{\sum_{i=1}^N w_i}$$

- Compute coefficient $\alpha_m = \text{log}\left(\frac{1-err_m}{err_m}\right)$
- Update weights:

$$w_i \leftarrow w_i e^{\alpha_m 1(y_i \neq \hat{f}^m(x_i))}$$

AdaBoost algorithm (final step)

3. Output boosted model:

$$\hat{f}(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m \hat{f}^m(x)\right]$$

AdaBoost example

AdaBoost example (2)

Boosting - algorithm for regression tree (1)

1. Let $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in training set.

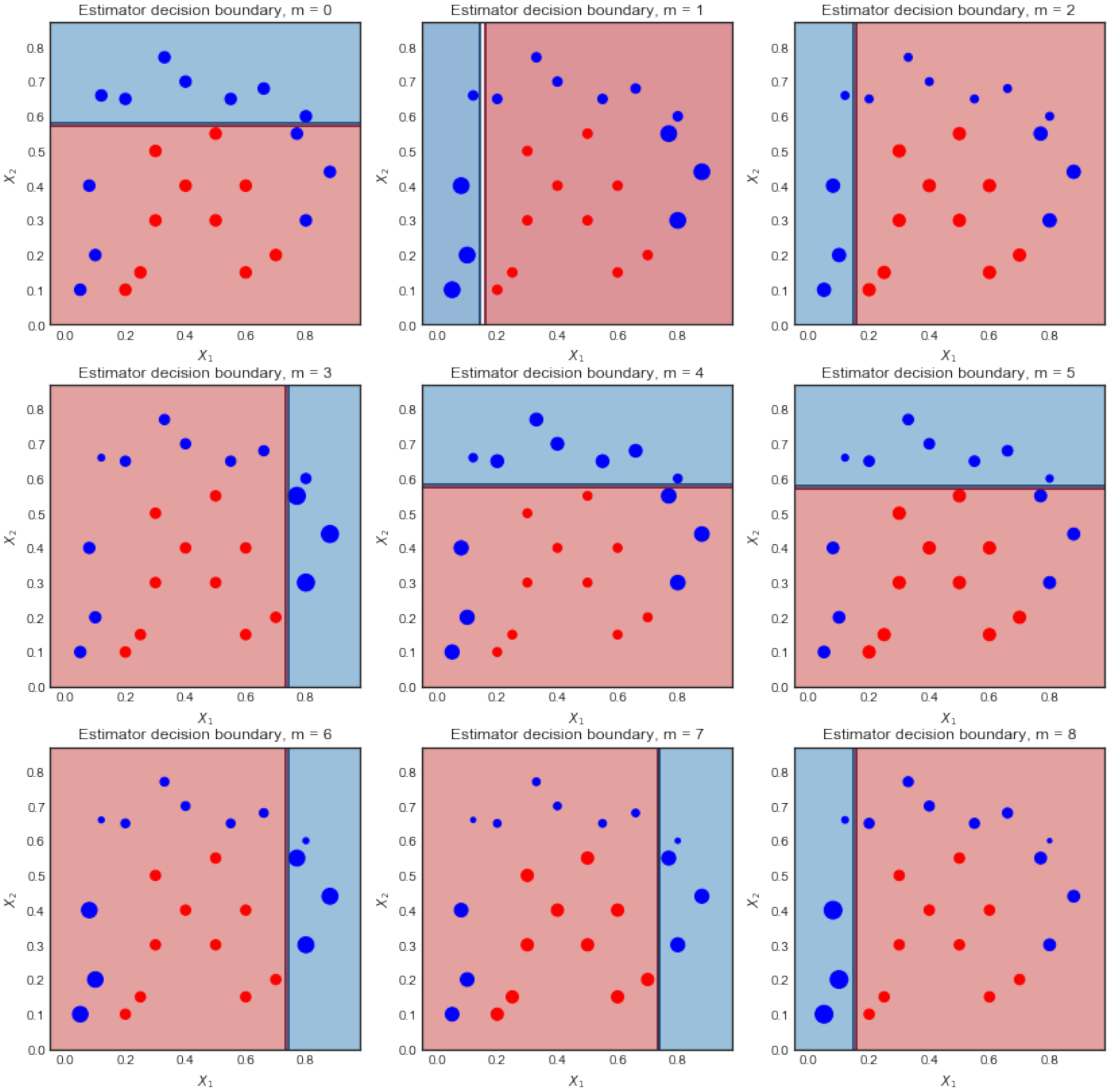


Figure 2: Example via https://xavierbourretsicotte.github.io/AdaBoost.html showing weak learners.

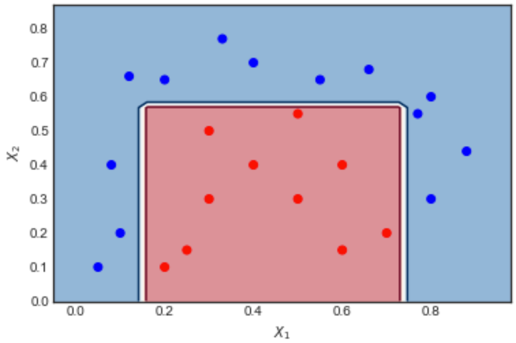


Figure 3: Average output after 10 iterations.

2. For $b = 1, \dots, B$, repeat:

Boosting - algorithm for regression tree (inner loop)

- Fit a tree $\hat{f}^b$ with d splits ($d+1$ leaf nodes) on training data (X, r).
- Update $\hat{f}$ with a *shrunken* version of new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

- Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x)$$

Boosting - algorithm for regression tree (final step)

3. Output boosted model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting - algorithm for regression tree (tuning)

Tuning parameters to select by CV:

- Number of trees B - can overfit if too large.
- Shrinkage parameter λ, controls *learning rate*. Typically $0.001 - 0.01$. Very small λ may require large B for good performance.
- d, number of splits in each tree. ($d = 1 \rightarrow$ tree is called a *stump*)

Boosting - example

Gradient descent intuition

Gradient Boosting

- General goal of boosting: find the model at each stage that minimizes loss function on ensemble (computationally difficult!)
- AdaBoost interpretation (discovered years after classification): Gradient descent algorithm that minimizes exponential loss function.
- Gradient boosting: works for any differentiable loss function. At each stage, find the local gradient of loss function, and take steps in direction of steepest descent.

Gradient descent: learning rate

Demo

- Demo on digits dataset

Stacking

Stacking - basic idea

- Learn base classifiers (not necessarily same type)
- Construct new dataset:

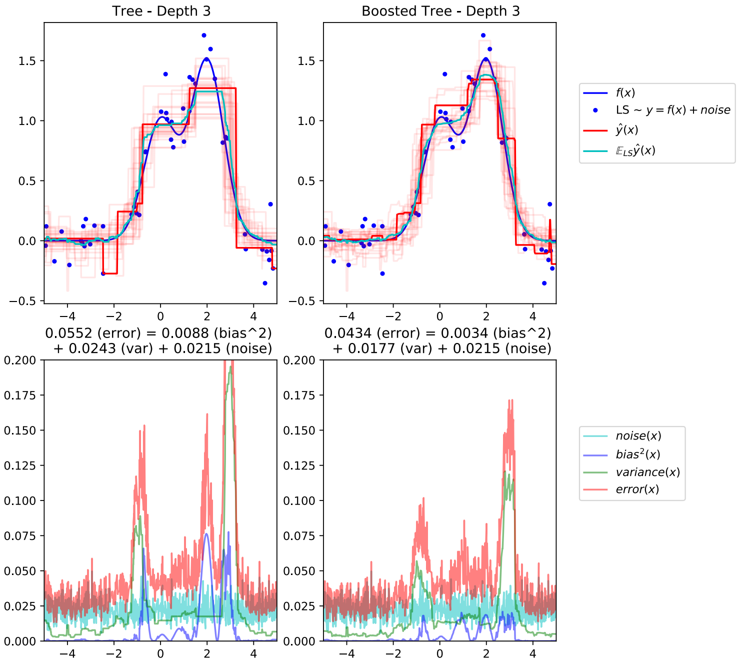


Figure 4: Boosting: reduce bias and variance!

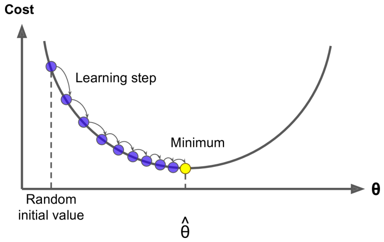
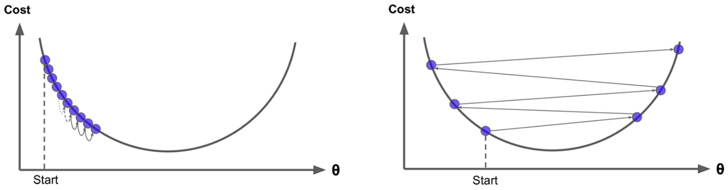


Figure 5: Intuition: gradient descent.



a) too small

a) too big

Figure 6: Intuition: effect of learning rate.

  - Features: output of base classifiers
  - Label: original label
- Learn a classifier on the “new” dataset

Summary of ensemble methods

- Can use a single estimator that has poor performance
- Combining the output of multiple estimators into a single prediction: better predictive accuracy, less interpretability