Ensemble Learning

Dr. Mohamed Elshenawy
mmelshenawy@gmail.com

1

In Previous Lectures

- Linear Regression
- Polynomial Regression
- Training and test error
- Bias and Variance Tradeoff
- Classification
- KNN
- Logistic Regression
- SVM
- Decision trees

2

In this session

- Ensemble Learning
  - Bagging (Bootstrap Aggregation)
  - Random Forests
  - Boosting

3

References

For further readings, check:

Section 8.2 from the book : An Introduction to Statistical Learning. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani, 2013, ISBN: 978-1-461-47137-0

4

Ensemble Learning

- Aim: take a mediocre algorithm (base model) and transform it into a super classifier without requiring any fancy new algorithm.

5

How can we combine different models?

- Ensemble methods differ in training strategies and combination methods
- We will discuss two approaches
  1. Parallel training of different models using overlapping training sets (Bagging)
  2. Sequential training by iteratively re-weighting training examples such that each model focuses on a different subset of hard examples (boosting)

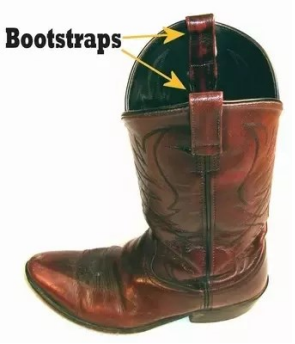
6

Bootstrap

7

Bootstrap

- The bootstrap is a widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- Allows us to emulate the process of obtaining new sample sets so that we can estimate the variability in our estimate.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set.



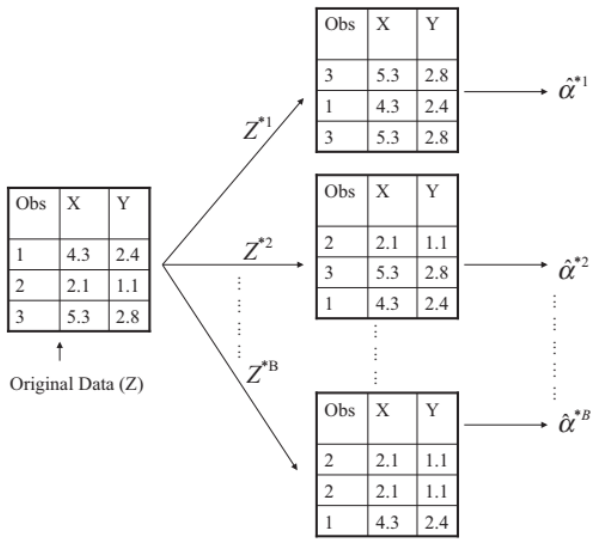
8

Procedure

- Suppose our original data set consists of n data points $X = \{x_1, x_2 \ldots, x_n\}$
- Multiple data sets are created as follows:
- Create a new data set X_B by drawing n points at random from X, with replacement, so that some points in X may be replicated in X_B, whereas other points in X may be absent from X_B.
- Repeat this process L times to generate L data sets each of size n and each obtained by sampling from the original data set X.

9

Obtain a different estimate of parameter α using a dataset Z



A graphical illustration of the bootstrap approach on a small sample containing n = 3 observations.

10

Bagging

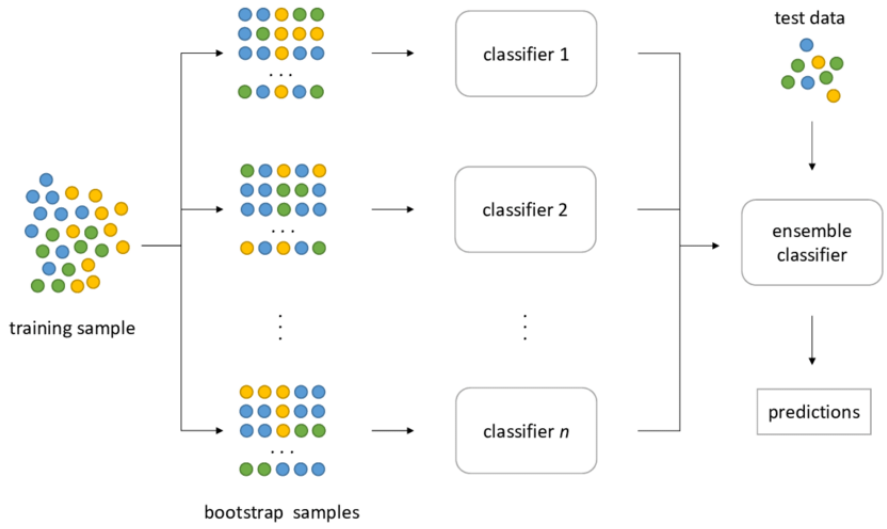
11

Bagging

- Bagging: a general-purpose procedure for reducing the variance (the sensitivity of the model to the individual data points) .
- Basic idea: create different models using different bootstrapped training sets and average their prediction to produce the final output of the model
- Bagging can improve predictions for many regression and classification methods,
- It is particularly useful for decision trees (suffer from high variance).

12

Illustration



13

Procedure

- We generate B different bootstrapped training data sets.
- We then train our method on the b^{th} bootstrapped training set in order to get $\hat{f}^{*b}(x)$
- finally average all the predictions, to obtain

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

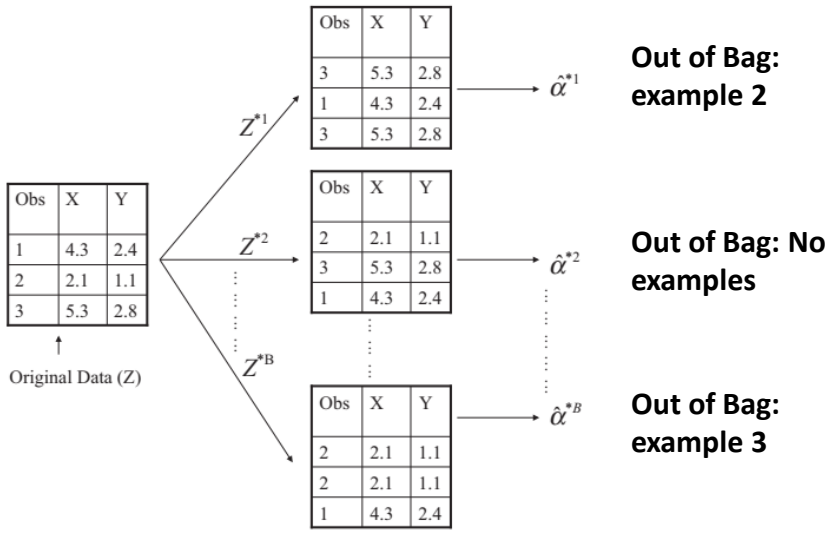
- For classification, you can average class probabilities (or take the majority vote)

14

Out-of-Bag (OOB) Error

- There is a very straightforward way to estimate the test error of a bagged model (without the need to perform cross-validation or the validation set approach)
- Out-of-Bag (OOB) error: OOB is the mean prediction error on each training sample x_i, using only the model that did not have x_i in their bootstrapped training set.
- That is, for each observation x_i in a dataset X, calculate the average prediction error for all models that do not have x_i in their bootstrap sample.
- The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

15



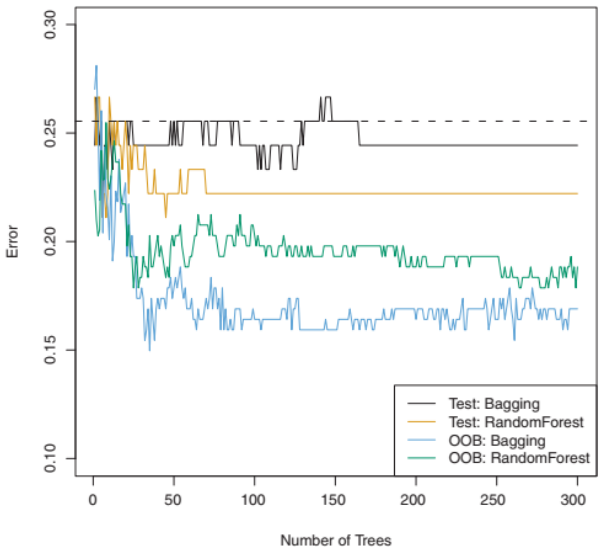
Out of Bag: example 2

Out of Bag: No examples

Out of Bag: example 3

16

Out-of-Bag Error (Cont.)



17

How can we estimate variable importance?

- Bagging typically results in improved accuracy over prediction using a single tree. Unfortunately, however, it can be difficult to interpret the resulting model.
- When we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure (bagging improves prediction accuracy at the expense of interpretability).

18

Variable Importance Measures

- You can calculate an overall summary of the importance of each predictor using *Variable Importance Measures* such as RSS (in regression models) and Gini Index (in classification models)

19

Variable Importance Measures- Regression

- For a given predictor, calculate the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees

20

Variable Importance Measures- Classification

- For a given predictor, calculate the total amount that the Gini index (measure of node *purity*—a small value indicates that a node contains predominant observations from a single class) is decreased by splits over a given predictor, averaged over all B trees.

21

Random Forest

22

Random Forests

- Random forests provide an improvement over bagged trees by way of a random small tweak that *decorrelates* the trees.
- You may consider Random forest as many, relatively uncorrelated, models that operate as a committee to make decisions

23

Random Forests - Random Variable Selection

- As in bagging, we build a number forest of decision trees on bootstrapped training samples.
- However, when building these decision trees, each time a split in a tree is considered, we select a random sample of m predictors (out of the available p predictors).
- The split is allowed to use only one of those m predictors and a fresh sample of m predictors is taken at each split.
- Typically we choose $m \approx \sqrt{p}$—that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.
- Why do we make random variable selection?

24

Random Forests - Random Variable Selection (Cont.)

- Why do we need random variable selection in random forests? the algorithm is not even allowed to consider a majority of the available predictors when preforming the splitting.
- The answer is: to build a set of relatively *uncorrelated models.*
- Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

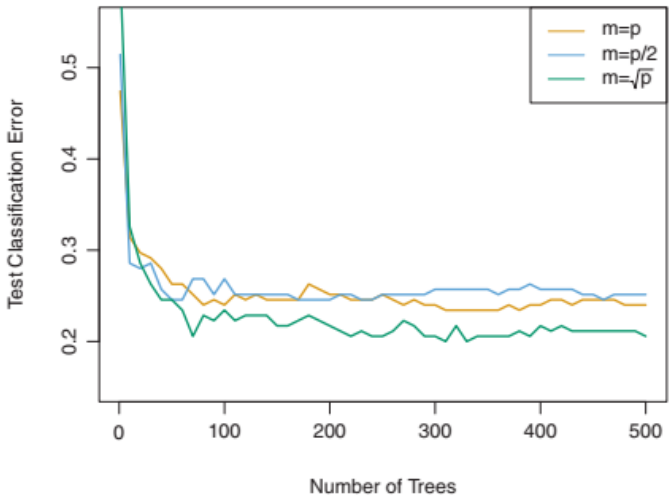
25

Uncorrelated trees

- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.
- Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split (will look quite similar to each other).
- Hence the predictions from the bagged trees will be highly correlated (does not lead to a sufficient reduction in variance).
- Random forests overcome this problem by forcing each split to consider only a subset of the predictors.

26

Impact of choosing different m values



27

Boosting

28

Boosting – Basic idea

- The principal difference between boosting and the bagging algorithm is is that the base models are trained in sequence.
- Each model is trained using information from previously trained models.
- Boosting does not involve bootstrap sampling; instead each tree is fit on a *modified version of the original data set.*

29

Boosting for Regression Trees - Algorithm

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Tuning Parameters:

- B, number of models
- d, depth of each model
- λ: shrinkage parameter

30

Basic idea

- Fit a decision tree to the residuals from the model (mistakes of this model), rather than the outcome Y , as the response.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well.
- The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals

31

- Suppose the value you are predicting is 12 and $\lambda = 0.1$
- First model: predicts 6 (i.e. residue: 6)
- Second model tries to predict 6, let's say it predicts 4.5
- Total prediction (6+0.1*4.5=6.45) – residue:5.55

…….

32

Tuning Parameters

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. (We use cross-validation to select B).
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable.

33