NYU Center for Data Science: DS-GA 1003

Machine Learning and Computational Statistics (Spring 2018)

Brett Bernstein

April 5, 2018

Instructions: Following most lab and lecture sections, we will be providing concept checks for review. Each concept check will:

- List the lab/lecture learning objectives. You will be responsible for mastering these objectives, and demonstrating mastery through homework assignments, exams (midterm and final), and on the final course project.
- Include concept check questions. These questions are intended to reinforce the lab/lectures, and help you master the learning objectives.

You are strongly encourage to complete all concept check questions, and to discuss these (and related) problems on Piazza and at office hours. However, problems marked with a ($\star$) are considered optional.

Trees, Bootstrap, Bagging, and RFs

Trees

Trees Learning Objectives

- Be able to describe the structure of a binary tree (ex: put bounds on number of leaves given height; describe the geometry of the resulting prediction function; etc.).
- Give pseudocode for finding the optimal split for (a) a continuous feature, and (b) a categorical feature for a binary classification problem.
- Describe some reasonable strategies for controlling the complexity of a tree.
- In particular, describe the regularization approach used in CART (pruning and use of number of leaves as complexity measure), recognize the cost complexity criterion as our standard regularized ERM.
- Recall the entropy, Gini, and misclassification error splitting criteria. Give some intuition around preference for Gini/entropy (i.e. purity measures) over misclassification.

Trees Concept Check Questions

1. (a) How many regions (leaves) will a tree with k node splits have?

   (b) What is the maximum number of regions a tree of height k can have? Recall that the height of a tree is the number of edges in the longest path from the root to any leaf.

   (c) Give an upper bound on the depth needed to exactly classify n distinct points in $\mathbb{R}^d$. [Hint: In the worst case each leaf will have a single training point.]

   Solution.

   (a) Given a fixed tree, if we split a leaf node we add a single leaf to the tree. Thus k splits corresponds to $k+1$ leaves.

   (b) A tree of height k can have at most 2^k regions (leaves).

   (c) A tree of height $\lceil \log_2(n) \rceil$ is sufficient to distinguish all possible values for the first feature. At each leaf we can then put another tree of this height that distinguishes the second feature, and so forth. These give an upper bound of $d\lceil \log_2(n) \rceil$.

2. This question involves fitting a regression tree using the square loss. Assume the n data points for the current node are sorted by the first feature. Give pseudocode with $O(n)$ runtime for optimally splitting the current node with respect to the first feature.

   Solution.

```
import numpy as np
def bestSplit(y) :
    '''
    Greedily computes the best splitting point for the current node.
    Assumes there is at least 2 values. There are more numerically
    stable ways of doing this
    [see The Art of Computer Programming p. 232, Vol 2, 3rd Edition]
    @param y : array-like, shape = [n_samples,], contains the
    output values for each sample. Assumes the
    values are sorted by the corresponding first
    feature values.
    @return the value i such that inputs 0,...,i belong in the
    left subtree.
    '''
    sums = np.cumsum(y) #partial sums of y-values
    sumsq = np.cumsum([a**2 for a in y]) #partial sums of squared y-values
    S = sums[-1] #sum of all y-values
    SS = sumsq[-1] # sum of all squared y-values
    bestIdx = -1
    bestLoss = None
    N = len(y)
```

```
for idx in range(N−1) :
    leftLoss = sumsq[idx] − sums[idx]**2/(idx+1.0)
    rightLoss = (SS−sumsq[idx])−(S−sums[idx])**2/(N−(idx+1.0))
    loss = leftLoss+rightLoss
    if bestIdx == −1 or loss < bestLoss :
        bestIdx = idx
        bestLoss = loss
return bestIdx,bestLoss
```

3. Suppose we are looking at a fixed node of a classification tree, and the class labels are, sorted by the first feature values,

$$4, 1, 0, 0, 1, 0, 2, 3, 3.$$

We are currently testing splitting the node into a left node containing $4, 1, 0, 0, 1, 0$ and a right node containing $2, 3, 3$. For each of the following impurity measures, give the value for the left and right parts, along with the total score for the split.

(a) Misclassification error.

(b) Gini index.

(c) Entropy.

Solution.

(a) Left: 3/6, Right: 1/3, Total: $6(3/6) + 3(1/3) = 4$

(b) Left: $3/6(3/6) + 2/6(4/6) + 1/6(5/6) = 22/36$, Right: $1/3(2/3) + 2/3(1/3) = 4/9$, Total: $6(22/36) + 3(4/9) = 30/6 = 5$

(c) Left: $-3/6 \log(3/6) - 2/6 \log(2/6) - 1/6 \log(1/6)$, Right: $-1/3 \log(1/3) - 2/3 \log(2/3)$, Total:

$$6[-3/6 \log(3/6) - 2/6 \log(2/6) - 1/6 \log(1/6)] + 3[-1/3 \log(1/3) - 2/3 \log(2/3)].$$

Bootstrap and Bagging

Bootstrap and Bagging Learning Objectives

- Recall from basic statics concepts related to an estimator (e.g. bias) and its variance.
- Describe (outside the context of bagging/RFs) how the bootstrap is a useful method for estimating the variance of an estimator, and have some intuition on how it can be applied across many problems.
- Again recalling basic statistics, understand why bagging (averaging predictions) reduces variance.

- Recalling that the bootstrap ignores an expected 37% of data in each bootstrap sample, explain how we can use out-of-bag observations to approximate test performance.
- Describe how RF reduces correlation between trees using column sampling while training on bootstrap samples.

Bootstrap and Bagging Concept Check Questions

1. Let $X_1, \ldots, X_n$ be an i.i.d. sample from a distribution with mean μ and variance σ^2. How large must n be so that the sample mean has standard error smaller than .01?

   Solution. Recall that the sample mean has variance

   $$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\text{Var}(X_1)}{n} = \frac{\sigma^2}{n},$$

   with standard error $\sigma/\sqrt{n}$. Thus we have

   $$\sigma/\sqrt{n} < .01 \iff n > 10000\sigma^2.$$

2. Let $X_1, \ldots, X_{2n+1}$ be an i.i.d. sample from a distribution. To estimate the median of the distribution, you can compute the sample median of the data.

   (a) Give pseudocode that computes an estimate of the variance of the sample median.

   (b) Give pseudocode that computes an estimate of a 95% confidence interval for the median.

   Solution.

   (a)
   i. Draw B bootstrap samples $D^1, \ldots, D^B$ each of size $2n+1$. The samples are formed by drawing uniformly with replacement from the original data set $X_1, \ldots, X_{2n+1}$. We will make a total of $B(2n+1)$ draws.
   ii. For each D^i compute the corresponding median $\hat{m}_i$.
   iii. Compute the sample variance of the B medians $m_1, \ldots, m_B$.

   (b)
   i. Draw B bootstrap samples $D^1, \ldots, D^B$ each of size $2n+1$. The samples are formed by drawing uniformly with replacement from the original data set $X_1, \ldots, X_{2n+1}$. We will make a total of $B(2n+1)$ draws.
   ii. For each D^i compute the corresponding median $\hat{m}_i$.
   iii. Compute the 2.5% and 97.5% sample quantiles of the list $\hat{m}_1, \ldots, \hat{m}_B$. Use these as the estimates of the left and right endpoints of the confidence interval, respectively.