Tree Based Algorithm

F. Song

1 Tree Algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations. *When decide which node to split on, choose the one with the maximum information gain*

2. Use cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α. CostFunction $= \sum_{m=1}^{|T|} \sum_{i,x_i \in R_m} (errorterm) + \alpha|T|$. $|T|$ stands for the size or the number of nodes of the tree. α determines the size of the tree. The tree is trained to minimize the cost.

3. Use K-fold cross validation to choose α.(cross validation is an effective way to choose a model from a model set)

4. Having chosen α, we go back to the full tree and find the sub-tree that has the smallest error.

2 Error Measure

2.1 Mean square error

Commonly used for regression problem.

2.2 Gini Index

The Gini index is defined by $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
Gini index is referred to as a measure of purity - a small value indicates that a node contain predominantly observations from a single class.

2.3 Cross Entropy

Cross entropy is givne by $D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$
It turns out the Gini index and cross entropy are similar numerically.

3 Bootstrap aggregation or bagging

Bootstrap aggregation, or bagging, is a general procedure for reducing the variance of a statistical learning method.

Recall that given a set of n **independent** observations$Z_1, Z_2, ...Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n

Pruning is trying to reduce the variance, but leads to high bias, the idea of bagging is **donot prune back,have bushy trees, which has low bias, get rid of the variance by doing this averaging**

p.s. Bootstrapping is a resampling method, it's random sampling with replacement.

4 Random Forest

Ramdom forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduce the variance when we average the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, **a random selection of m predictors** is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh selection of m predictors is taken at each split, and typically we choose $m = \sqrt{q}$- that is, the number of predictors considered at each split is approximately equal to the squre root of the total number of predictors.

5 out of bag error estimation

It turns out that there si a very straightforward way to estimate the test error of a bagged model. Recall that the key to bagging is that trees are repeatedly fit to bootstrpped subsets of the observations. One can show that on average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are fererred to as teh **out the bag** observations. We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around B/3 predictions for the ith observation, which we average. This estimate is essentially leave one out cross validation error for bagging, if B is large.

6 Boosting

Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. Similar to bagging and random forest, it gives prediction models that are average of over trees. But there is a fundamental difference. Random forest and bagging, the trees that are averaged are all equivalent and the average is just used to reduce variance. With boosting it's a sequential method. And each of the trees that's added into the mix is added to improve on the performance of the previous collection of trees. Gradient boosting is an instantiating of boosting for regression.
Step1: Learn a regression predictor
Step2: Compute the error residual
Step3: Learn to predict the residual

Boosting algorithm for regression trees:

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

Boosting is based on weak learners(high bias, low variance). Boosting reduces error mainly by reducing bias(and also to some extent variance, by aggregating the ouput from many models)

On the other hand, Random forest uses fully grown decision trees(low bias, high variance). It tackles the error reduction task by reducing variance. The trees are made uncorrelated to maximize the decrease in variance, but the algorithm cannot reduce bias(which is slight higher than the bias of an individual tree in the forest). Hence the need for large, unpruned tress, so that the bias is initially as low as possible.

7 Greedy decision tree learning

1. Step1: start with an empty tree
2. Step2: Select a feature to split data
3. For each split of the tree:
   (a) Step 3: If nothing more to split, make predictions
   (b) Step4: Otherwise, go to Step 2 continue on this split

Stopping conditions:

1. All examples have the same target value
2. No more features to split on

How to choose which feature to split on first:

1. Given a subset of data
2. For each feature:
   (a) split data of M according to the feature
   (b) compute classification error
3. Choose feature with lowest classfication error

8 How to pick nodes/ choose which feature to split on

Entropy of distribution

$$H(\pi) = -\Sigma\pi log\pi \tag{3}$$

For a training set containing p positive examples and n negative examples, we have:

$$H(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}log_2\frac{p}{p+n} - \frac{n}{p+n}log_2\frac{n}{p+n} \tag{4}$$

maximize utility method: Chosen attribute A, with K distinct values, divides the training set E into subsets $E_1, ...E_k$,The **Expected Entropy(EH)** remaining after trying attribute A is:

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{p+n} H(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}) \tag{5}$$

where $p_i + n_i$ is the points in child i, $p + n$ is the child in parent node

The **information gain** or **reduction in entropy** for this attribute is :

$$I(A) = H(\frac{p}{p+n}, \frac{n}{p+n}) - EH(A) \tag{6}$$

Choose the attribute with biggest I

we bag trees because trees are very different from each other, and that's an guarantee of uncorrelateness. . When averaging them, we are close to the right answer.