29 Gradient Boosting

Lab Objective: *Understand how to build an ensemble of regression trees using gradient boosting and use it to predict survival of Titanic passengers.*

In the previous lab we saw that building an ensemble of classification trees could outperform one tree (by reducing the variance). In this lab, we will present another method for building an ensemble of trees, this time with the intent of decreasing the bias (hopefully without increasing the variance too much).

NOTE: A very good tutorial on gradient boosting in this context can be found at `http://xgboost.readthedocs.io/en/latest/model.html`

First of all, for this lab it will be more convenient to consider *regression trees*, where values of the output variable are permitted to take any value in $\mathbb{R}$. Instead of the Gini impurity measure, we instead use the mean-squared error:

$$MSE(D) = \sum_{i=1}^{n} (y_i - f(x_i))^2. \tag{29.1}$$

Here D is again some subset of the data.

Previously, we used designed stopping criteria by putting a threshold on the Gini impurity, or on the tree depth. In this setting we will take a slightly different (but very similar) approach, and define the complexity of a particular $f(x) = \sum_{k=1}^{T} \chi_{D_k} w_k$ by

$$\Omega(f) = \gamma T + \frac{\lambda}{2} \sum_{k=1}^{T} w_k^2$$

Here γ and λ are regularization parameters, and T is the total number of leaves in the tree. χ_{D_k} is an indicator function, meaning that χ_{D_k} takes value 1 on the leaf D_k and 0 elsewhere.

Our overall goal will be to train K trees which minimize the following:

$$J(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \sum_{j=1}^{K} \Omega(f_k), \quad f = \sum_{j=1}^{K} f_k$$

We acheive this by training the trees iteratively, namely we first train f_1, then f_2 and so on. More concretely, if we let $\sum_{j=1}^{t} f_j = f^t$ then training f_{t+1} becomes a problem of minimizing

$$J(f_{t+1}) = \sum_{i=1}^{n} (y_i - (f^t(x_i) + f_{t+1}(x_i))^2 + \Omega(f_{t+1}).$$

After some simple manipulations this is equivalent to minimizing

$$\tilde{J}(f_{t+1}) = \sum_{i=1}^{n} (2(y_i - f^t(x_i))f_{t+1}(x_i) + f_{t+1}(x_i)^2) + \Omega(f_{t+1}) =: \sum_{i=1}^{n} g_i f_{t+1}(x_i) + h_i f_{t+1}(x_i).$$

Given a particular split D_L and D_R we can compute the objective value and optimal function values on the split via the formulas

$$w_i = -\frac{G_i}{H_i + \lambda}, \quad i = L, R, \quad J = -\frac{G_L^2}{2(H_L + \lambda)} - \frac{G_R^2}{2(H_R + \lambda)} + 2\gamma. \tag{29.2}$$

where here G_i is the sum of the g_i and H_i is the sum of h_i (all within the set of interest, namely on one side of the split).

Problem 1. For a given split, write a function which computes the optimal function and objective values using equation (29.2).

Equation (29.2) can naturally be generalized to give the optimal function values and objective value for the whole tree. However, this isn't so helpful, since we will always train trees in a greedy way, instead of the whole tree at once. Thus whenever we consider a particular split, we will seek for the split which maximizes

$$Gain = \frac{1}{2}\left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right) - \gamma \tag{29.3}$$

If this value is always negative, then we are not benefited by branching: namely the γT term will always be greater than the gain in the training error.

Problem 2. Modify your previous code to construct regression trees using the mean-squared error as the training objective (as opposed to the Gini impurity). You will need to optimize both over the split and over the value which the tree takes on each split.

In summary, we have the following algorithm:

```
For t = 1 to K:
    compute g,h
    recursively build tree
    (by finding the split which maximizes (29.3), and stop branching if the max↩
        gain < 0)
```

Problem 3. Implement the previously algorithm, reusing your code from the classification tree lab as appropriate.

Problem 4. Use your gradient boosting code to train a classifier for the titanic data. Let $K = 100$ and find appropriate values for γ and λ. Note that you can just encode the categorical survival data as continuous variables $0, 1$. The output of the regression tree will be values in $[0, 1]$, which can be turned into labels by thresholding at .5. How does the performance compare with random forests? Oftentimes boosted trees work a little better.