JBRT: Java Boosting Regression Tree

July 7, 2015

1 Usage

The project can be imported into Eclipse directly. The main Program is in src/boostingRegressionTree/Model.java. A test Dataset in data/testDataset.csv is used as training data. The output model file is stored in ui/model.json. It has been visualized in ui/gbrt.html(Firefox required), as Fig 1 shows:

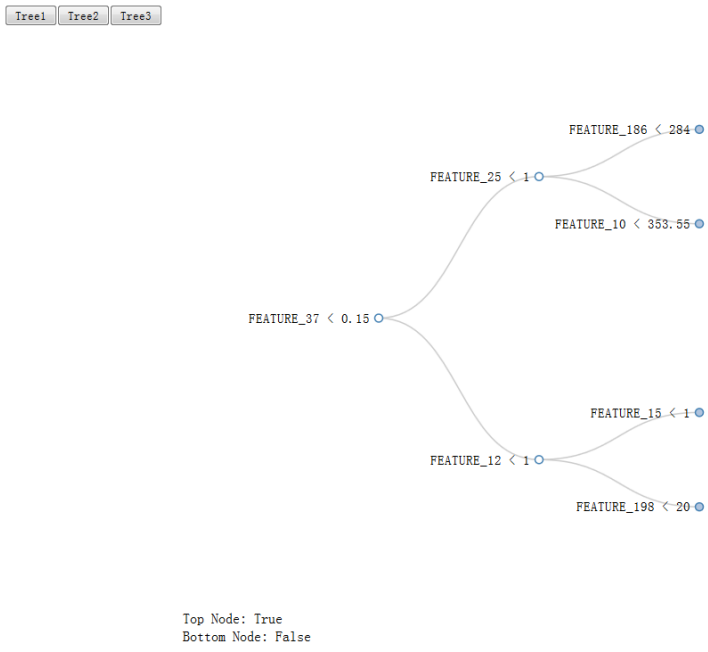

Figure 1: Boosting Regression Tree model visualization

2 Basic theory

The basic idea of Boosting: based on previous prediction $F(\boldsymbol{X})$, iteratively build a new $f(\boldsymbol{X})$ to improve the performance:

$$\hat{f} = \arg\min_{f} \sum_{i} L(y_i, F(\boldsymbol{X}_i) + f(\boldsymbol{X}_i)) \tag{1}$$

$f(\boldsymbol{X})$ can be a tree(Figure 2):

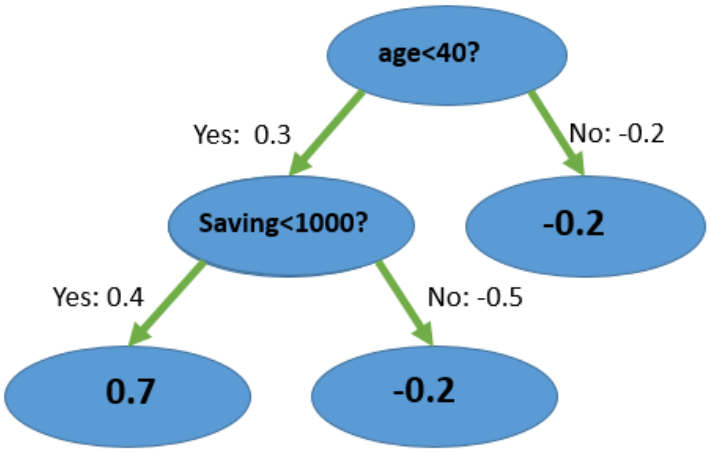

Figure 2: Regression Tree

The tree divides the feature space into several regions. For each region r, a value θ_r is assigned. Thus $f(\boldsymbol{X})$ can be written as:

$$\begin{aligned} f(\boldsymbol{X}) &= \sum_{r} \theta_r I_r(\boldsymbol{X}) \\ I_r(\boldsymbol{X}) &= \begin{cases} 1 & \boldsymbol{X} \in r \\ 0 & \text{else} \end{cases} \end{aligned} \tag{2}$$

Plug it into Eq 1, now the target is to find the best parameters $\boldsymbol{\theta} = \{\theta_{r_1}, \theta_{r_2}, ...\}$:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i} L(y_i, F(\boldsymbol{X}_i) + \sum_{r} \theta_r I_r(\boldsymbol{X_i})) \tag{3}$$

Note that each θ_r in $\boldsymbol{\theta}$ can be calculated independently:

$$\hat{\theta_r} = \arg\min_{\theta_r} \sum_{i \in r} L(y_i, F(\boldsymbol{X}_i) + \theta_r) \tag{4}$$

If Least Square Loss is adopted, then

$$\begin{aligned} \hat{\theta_r} &= \arg\min_{\theta_r} \sum_{i \in r} (y_i - F(\boldsymbol{X}_i) - \theta_r)^2 \\ &= \frac{\sum_{i \in r}(y_i - F(\boldsymbol{X}_i))}{|r|} \end{aligned} \tag{5}$$

From the above inference we can see, with the partition of the feature space $R = \{r_1, r_2, ..., r_n\}$ fixed, $f(\boldsymbol{X})$ can be calculated. Thus the problem remained is to find a best partition to minimize Eq 1. In JBRT, we simply adopt sample approach in each feature dimension.