1 Regression tree

Tree-based methods partition the feature space into a set of rectanggles and fit a simple model in each one. They are conceptually simple yet powerful tools.

Given a region R which is a subset of domain D, define function I_R on D

$$I_R(p) = \begin{cases} 1 & \text{if } p \in R, \\ 0 & \text{if } p \notin R \end{cases}$$

Consider a regression problem with a collection of responses y and features $x \in \mathbb{R}^p$. A regression tree consists of a partition of $\mathbb{R}^p$ $R_1, R_2, \cdots, R_m$ and a prediction of y for each region in the partition: $c_1, c_2, \cdots, c_m$. Formally

$$\hat{y} = \sum_{i=1}^{m} c_i I_{R_i}(x)$$

We will recursively define the process of growing a tree. Suppose we already have a partition $R_1, \cdots, R_{m-1}$. On each of the region R_i, we want to further split the region into two parts: $R_{i1}^{js} = \{x | x_j < s, x \in R_i\}$ and $R_{i2}^{js} = \{x | x_j >= s, x \in R_i\}$. Such a split has two unfixed parameters: which feature we are going to split over (j) and where we are goint to split over (s). We select this value by achieving the best local result in the target function.

$$(j, s) = \arg\min_{j,s}[\min_{c_1} \sum_{x_k \in R_{i1}^{js}} (y_k - c_1)^2 + \min_{c_2} \sum_{x_k \in R_{i2}^{js}} (y_k - c_2)^2]$$

And c_1, c_2 will be the prediction for R_{i1}^{js} and R_{i2}^{js} respectively.

In the case when we use the least square error as cost function, the above simplies to

$$\arg\min_{j,s}[\sum_{x_k \in R_{i1}^{js}} (y_k - \bar{y}_{i1}^{js})^2 + \sum_{x_k \in R_{i2}^{js}} (y_k - \bar{y}_{i2}^{js})^2]$$

Here $\bar{y}$ is the average of y which belong to the region indicated by the subscripts of $\bar{y}$

In the case when cost function is the absolute error. $\bar{y}$ is replaced by the median of y in the corresponding region.

Obviously a tree can overfit the data. We introduce a one regularization method. The idea is to first grow a relativelly large tree using primitive method, then prune it.

We grow the tree until the number of nodes reach a fixed number. Call this tree T_0. Consider any subtree $T \subset T_0$ which can be realized by collapsing some of T_0's nodes. Define cost function

$$C(T) = \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_m)^2 + \alpha |T|$$

$|T|$ is the number of leaves of T and $R_1, R_2, \cdots, R_m$ are the partition of $\mathbb{R}^p$ of the prediction model corresponding to T (the regions represented by the leaves of T). α is a tunning parameter of model.

For any α, we can find $T_\alpha = \text{arg min } C(T)$ through weakest link pruning: That is we succesively close the node of T_0 which produces the smallest increase pernode increase in $C(T) - \alpha|T|$ until we are left with one node. It can be shown that this sequence contains T_α.

2 Random forest for regression tree

Suppose we want to fit a certain model on a training data. Bagging or bootstrap aggregation fits the model on a collection of bootstrap samples and average their prediction result. Bagging reduces the variance and maintain the same bias as a single model would have.

Random forests is a modification of bagging that builds a large collection of de-correlated trees and average over them. The procedure can be described as:

1. Parameters: B, N, J, d, training data $\{y, x\}$
2. For b in $1 \ldots B$
   (a) Draw a bootstrap sample of size N from the training data $\{y, x\}$
   (b) Grow a tree T_b with J nodes with standard procedure on boostrapped data, with one modification: each time we split the node, we randomly pick d direction in feature space $\mathbb{R}^p$ and use them rather than using the whole feature space.
3. Make prediction by $\frac{1}{B}\sum_{b=1}^{B} T_b(x)$

3 Gradient boosting for regression tree

3.1 Boosting for additive model

Boosting is a way of fitting an additive expansion in a set of elementary basis functions. The prediction is given by

$$f(x) = \sum_{i=1}^{M} T(x; \gamma_i)$$

where $T(;\gamma_i)$ are a basis functions with different tunning parameters. If applied to regression trees, T are single tree and γ_i are the partition R_j and prediction for each region in the partition (for least square cost function, it's just average of y for that region).

In general, we can fit such model by "forward stagewise modeling". We sequentially add new basis functions to the expansion and achieve the minimal cost function by adjusting the parameters of the newly added model:

$$\gamma_k = \text{arg min}_{\gamma_k^*} C(y - \sum_{i=1}^{k-1} T(x;\gamma_i), T(x;\gamma_k^*))$$

3.2 Gradient boosting

Forward stagewise boosting doesn the same thing in "space of functions" as the convex optimization does in "space of coefficients", indeed, various techniques for the latter can be applied to the former. The mirror of gradient descent and line search in convex optimization here is gradient boosting.

Consider the derivative

$$g_k = \frac{\partial C(y,f)}{\partial f}|_{f=\sum_{i=1}^{k-1} T(x;\gamma_i)}$$

The difference between this and gradient descent, is that we cannot find a new model with exactly the same direction as g_k, therefore we solve the optimal problem first:

$$\gamma_k = \text{arg min}_{\gamma_k^*} || - g_k - T(x;\gamma_k^*)||_2$$

and then do line search to find the optimal step size on the approximal direction of gradient:

$$\rho_k = \text{arg min}_{\rho_k} C(y, \sum_{i=1}^{k-1} \rho_i T(x;\gamma_i) + \rho_k^* T(x;\gamma_k))$$

If C is the least square function, then trivially $-g_k = y - \sum_{i=1}^{k-1}$ is just the residue of current model. For absolute error cost function $g_k = -\text{sign}(y - \sum_{i=1}^{k-1})$, which is the signs of residue. Gradient boosting make it convenient to generalize to classification problem or regression problem with other cost function.

As with any model, boosting needs to be regularized. Typical method includes restricting the number of additive models used to be less than M. Another is adding a shrinkage factor when adding new models ν:

$$f \to f + \nu \rho_k T(;\gamma_k)$$

3.3 Applying to trees

The general procedure of gradient boosting almost directly applies to the regression tree. Due to the piecewise constant nature of regression tree as a function, we see that picking the step size and picking the prediction for each region of tree actually overlap each other and therefore we can combine them togather. To specify, the whole procedure is

1. Parameters M, J, training data y, x
2. For k in $1 \dots M$:

(a) $g_k = \frac{\partial C(y,f)}{\partial f}|_{f=\sum_{i=1}^{k-1} T_i(x)}$

(b) Fit a regression tree with J leaves T'_k to the target g_k, take its leaves: $R_{1k}, \ldots, R_{Jk}$ (but not coefficients)

(c) For j in $1, \ldots, J$: compute $c_l = \arg\min_c C(y, \sum_{i=1}^{k-1} T_k(x_m) + c)|_{R_{jk}}$

(d) $T_k = \sum_{j=1}^{J} c_j I(R_{jk})$

3. Predict by $\sum_{i=1}^{M} T_i(x)$