Gradient Boosting

Recapping Decision Trees

$$T(x, \theta) = \sum_{j=1}^{J} \gamma_j \, I(x, R_j)$$

$$\theta = \{R_j , \gamma_j\}_{j=1}^{J}$$

$$\theta = \arg\max_{\theta}() \sum_{j=1}^{J} \sum_{x_i \epsilon R_j} L(y_i, \gamma_j)$$

Boosted Trees

$$f_m = \textstyle\sum_{m=1}^{M} \; T(x, \theta_m)$$

For Regression Trees

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} L(\, y_i \,, \, f_{m-1}(x_i) + T(x_i \,, \, \theta_m) \,)$$

Squared Error Loss : Pick the tree that best predict the residual.

$$y_i - f_{m-1}(x_i) \; : \; \text{Target Function}$$

$$\gamma_{jm} \; : \; \text{average residul error in the } \mathrm{R}_{jm} \text{ region}$$

For **2 - class classification problems** and **Exponential Loss Function**, it becomes exactly same as **AdaBoost solution on decision trees**

Differential Loss Functions

$$L(f) = \sum_{i=1}^{N} L(y_i\,,\ f(x_i))$$

$$\hat{f} = \arg\min_{f} L(f)$$

$$f = (f(x_1),\ f(x_2),\ \ldots\ldots,\ f(x_N))$$

$$f_0 = h_0$$

$$f_M = \sum_{m=0}^{M} h_m \qquad h_m \,\epsilon\, R^N$$

– Steepest Descent

$$h_m = -\rho_m g_m$$

$$g_m = \left[\frac{\nabla L(y_i\,, f(x_i))}{\nabla f(x_i)} \right]_{f(x_i) = f_{m-1}(x_i)}$$

$$\rho_m = \arg\min_{\rho} L(f_{m-1} - \rho_{g_m})$$

$$f_m = f_{m-1} - \rho_m g_m$$

$$g_m \approx \text{unconstrained maximal descent direction}$$

$$\hat{\theta}_m = \arg\min_{\theta} \sum_{i=1}^{N} (-g_{im} - T(x_i\,;\,\theta))^2$$

Fit a tree "as close as" possible to gradient direction.

Regression:

$$\frac{1}{2}(y_i - f(x_i))^2 \qquad y_i - f(x_i)$$

$$|y_i - f(x_i)| \qquad \text{sign}(y_i - f(x_i))$$

Classification:

Deviance $\quad \mathrm{I}(y_i = \mathrm{C}_k) - P_k(x_i) \quad : (i^{th}\,\text{component})$

$$\hat{\gamma}_{jm} = \arg\min_{\gamma} \sum_{x_i \,\epsilon\, R_{jm}} L(y_i\,, f_{m-1}(x_i) + \gamma)$$

XG BOOST - it's Xtreme..

- XGBoost works as Newton-Raphson in function space unlike gradient boosting that works as gradient descent in function space.
- A second order Taylor approximation is used in the loss function to make the connection to Newton Raphson method.

A generic unregularized XGBoost algorithm is:

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α.

Algorithm:

1. Initialize model with a constant value:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$$

2. For m = 1 to M:

    1. Compute the 'gradients' and 'hessians':

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

$$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

XG BOOST continued..

2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$ by solving the optimization problem below:

$$\hat{\phi}_m = \underset{\phi \in \Phi}{\arg\min} \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i)\left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right]^2.$$

$$\hat{f}_m(x) = \alpha\hat{\phi}_m(x).$$

3. Update the model:

$$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$