Lec 21: Boosting II

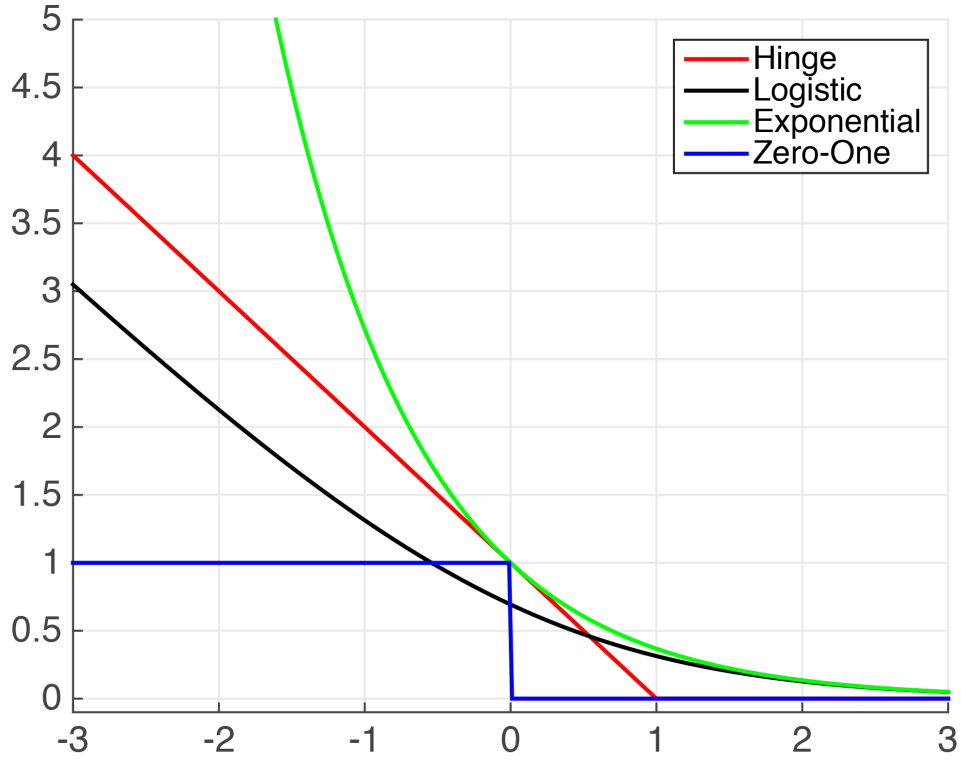
Ailin Zhang

Agenda

- Basic idea
- Weak learner example: classification tree
- L2 boosting
- AdaBoost
- Gradient Boosting
- Extreme Gradient Boosting (XGB)
- Connection to other models

Gradient Boosting and XGB are more flexible and admit any type of loss.

Exponential loss is the upper bound for 0/1 loss



Loss functions for classification. The horizontal axis is $m_i = y_i X_i^\top \beta$. The vertical axis is $L(y_i, X_i^\top \beta)$.

Adaboost Convergence Analysis

$\epsilon = b/s = \frac{\sum_{y_i \neq h(x_i)} w_i}{\sum_i w_i}$, and $Z_t = (1-\epsilon_t)e^{-\beta} + \epsilon_t e^{\beta} \geq 2\sqrt{\epsilon_t(1-\epsilon_t)}$

If we perform variable change $\gamma_t = \frac{1}{2} - \epsilon_t$

$$Z_t = \sqrt{1-4\gamma_t^2} \leq \exp(-2\gamma_t^2)$$

Therefore, after t steps, the error rate of the strong classifier is bounded by:

$Error(H) = \frac{1}{n}\sum_{i=1}^{n} 1(f(x_i) \neq y_i)) \leq \frac{1}{n}\sum_{i=1}^{n} \exp(-f(x_i)y_i) \leq$
$\frac{1}{n}\prod_{t=1}^{T} Z_t \leq \frac{1}{n}\exp(-2\sum \gamma_t^2)$

It is clear that each step the upper bound of the error decrease exponentially. A weak classifier with small error rate will lead to faster descent.

Adaboost Summary

- The objective of Adaboost is to minimize an upper bound of the classification error
- It takes a stepwise minimization scheme and it may not be optimal due to the greedy pursuit. When we calculate the parameter for the t-th weak classifier, we do not change the weights of the previous weak classifiers.
- We should stop AdaBoost if all the weak classifiers have error rate 0.5. This will eventually happen as we update the weights.

Adaboost to RealBoost

- In Adaboost, each weak classifier $h(x)$ is defined as a 1D binary function.
- We may extend $h(x)$ to a more general form$h(x;\theta)$; with θ being a vector of B parameter(B is the number of bins that we choose to approximate an arbitrary 1D function)

Relation between Adaboost and probabilities

Gradient Boosting

- Consider a more general minimization problem: $min_f \{\sum_{i=1}^{n} L(y_i, f(x_i))\}$, for any loss function L.
- The gradient boosting algorithm solves this problem by iteratively changing $f(x)$.
  - Take any given prediction (candidate for minimization) and call it $\hat{f}(x)$.
  - In $L2$ boosting, for example, the idea is to iterate over $\hat{f}(x)$, letting your new guess for what the minimizer is be $\hat{f}(x) + \beta h(x)$, where $h(x)$ is a weak-learner
  - With gradient boosting, we start by asking the question "what if we could change $\hat{f}(x)$ by any amount $\Delta f(x)$?". What would be the ideal amount we should use?
  - To minimize the objective function, we would like to let $(\Delta f_i)_{i=1\ldots n} \propto -\left(\frac{\partial L}{\partial f_i}\right)_{i=1\ldots n}$

Gradient Boosting

- with the gradient being evaluated at $\hat{f}(x_i)$, so that at each iteration we could get closer to solving our problem. We can let $\tilde{y}_i = -\frac{\partial L}{\partial f_i}|_{\hat{f}(x_i)}$.
- The gradient boosting algorithm relies on weak-learners, just as adaboost and $L2$ boosting do. We would like to use the gradient, but we are restricted to weak learners here (for example, tree stumps).
- Therefore, we need to find at every iteration weak learners h, so that $h(x_i) \propto \tilde{y}_i$.This is done using a simple heuristic. We choose β and h by minimizing a squared-loss problem:

$$min \left\{ \sum_{i=1}^{n} (\tilde{y}_i - \beta h(x_i))^2 \right\}$$

  - Not provably best or mathematically optimal. Instead, they are just ideas that somebody thought sounded good, implemented, then found to work in practice acceptably well

Gradient Boosting

- By solving the problem above, we then obtain h that is aligned to the negative of the gradient at the current $\hat{f}(x)$.
- Note that the loss we want to minimize is actually L, which may not be squared loss. Then, to choose the actuall β we will use, we look for $\beta = argmin_\beta \sum_{i=1}^{n} L(y_i, \hat{f}_i + \beta h(x_i))$.
- β here represents the step-size at the current iteration. By repeating this procedure, using the heuristic above to find the weak-learner best aligned with the gradient, and then solving for the step-size β, we can get closer and closer to finding the solution to the minimization problem posed above.

Question: How to initialize?

Gradient Boosting

- By solving the problem above, we then obtain h that is aligned to the negative of the gradient at the current $\hat{f}(x)$.
- Note that the loss we want to minimize is actually L, which may not be squared loss. Then, to choose the actuall β we will use, we look for $\beta = argmin_\beta \sum_{i=1}^n L(y_i, \hat{f}_i + \beta h(x_i))$.
- β here represents the step-size at the current iteration. By repeating this procedure, using the heuristic above to find the weak-learner best aligned with the gradient, and then solving for the step-size β, we can get closer and closer to finding the solution to the minimization problem posed above.

Question: How to initialize?

A typical choice for the initial $\hat{f}(x)$ would be a constant value γ, such that $\gamma = argmin_\gamma \{\sum_{i=1}^n L(y_i, \gamma)\}$.

Algorithm

Suppose we want to minimize a general loss:

$$\min \sum_{i=1}^{n} L(y_i, f_i).$$

Then gradient boosting consists of the following steps

- *Step 1*: Set $f_0(x_i) = \frac{1}{n}\sum_1^n y_i$, $m = 1$.
- *Step 2*: Compute residuals $y_i^m = y_i - f_{m-1}(x_i)$.
- *Step 3*: Choose h^* by minimizing $\sum_{i=1}^n (y_i^m - \beta h(x_i))^2$, and set $h_m = h^*$.
- *Step 4*: Reestimate β by $\beta^* = \arg\min_\beta \sum_{i=1}^n L(y_i, f)$
- *Step 5*: $m \leftarrow m + 1$; repeat Step 2.

XGB - Extreme Gradient Boosting

- XGB is a boosting algorithm that relies on the Newton-Rhapson method.

XGB - Extreme Gradient Boosting

- XGB is a boosting algorithm that relies on the Newton-Rhapson method.
- Consider a second-order aproximation for any loss function at a given point $\hat{\theta}$. We can write it as
$L(\theta) = L(\hat{\theta}) + L^{'}(\hat{\theta})(\theta - \hat{\theta}) + \frac{1}{2}L^{''}(\hat{\theta})(\theta - \hat{\theta})^2$.
- Then, an approximation for the total loss, around the current $\hat{f}(x)$ is:

$$\sum_{i=1}^{n} \left\{ L(y_i, \hat{f}(x_i)) + L^{'}(y_i, \hat{f}(x_i))\Delta f(x_i) + \frac{1}{2}L^{''}(y_i, \hat{f}(x_i))\Delta f(x_i)^2 \right\}.$$

- The central question XGB tries to answer is what are the $\Delta f(x_i)$ that we need to pick at every step.

XGB - Extreme Gradient Boosting

- Let $\hat{f}(x_i) = \hat{f}_i$, $g_i = L^{'}(y_i, \hat{f}_i)$ and $a_i = L^{''}(y_i, \hat{f}_i)$. And assume we want to use weak learners (trees) as the Δf_i. Then, at every iteration, we want to find $T(x) = \sum_{m=1}^{M} c_m 1(x \in R_m)$ that minimizes $\sum_{i=1}^{n} g_i T(x_i) + \frac{1}{2} a_i T(x_i)^2$.
- This is equal to $\sum_{m=1}^{M} c_m \sum_{i:x_i \in R_m} g_i + \frac{1}{2} c_m^2 \sum_{i:x_i \in R_m} a_i$. Let $\sum_{i:x_i \in R_m} g_i = G_m$, and $\sum_{i:x_i \in R_m} a_i = A_m$.
- To find the c_m for $m = 1...M$, M, and the regions R_m that minimize $\sum_{i=1}^{M} c_m G_m + \frac{1}{2} c_m^2 A_m$.

XGB - Extreme Gradient Boosting (Regularization)

- Also, in XGB, we explicitly penalize excess complexity, adding a penalty of the type $\gamma M + \frac{1}{2}\lambda \sum_{m=1}^{M} c_m^2$ to this problem (at each iteration). For any given M, the solution for each c_m is independent than that of the others. So we can find c_m by fixing M and solving:

$$c_m = argmin\{c_m G_m + \frac{1}{2} c_m^2 (A_m + \lambda)\}$$

XGB - Extreme Gradient Boosting (Computation)

- Taking the derivative and setting it to 0, we get:

$$G_m + c_m(A_m + \lambda) = 0$$

$$\Rightarrow c_m = -\frac{G_m}{A_m + \lambda}$$

- Plugging this into the last objective function mentioned above, we get that at the optimal, this function takes the value $-\frac{G_m^2}{A_m+\lambda} + \frac{G_m^2}{2(A_m+\lambda)} = \frac{-G_m^2}{2(A_m+\lambda)}$.
- At each iteration, we grow a tree to minimize $\sum_{m=1}^{M} -\frac{1}{2}\frac{G_m^2}{A_m+\lambda} + \gamma M$.