Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 12: Boosting (II) Gradient Boosting (Draft: version 0.5.1)

Topics to be covered:

- xyz

1 Forward stagewise additive model

Recall AdaBoost successively combines weak learners $h_t(x)$ to create a strong learner

$$g(x) = \sum_{t=1}^{T} \omega_t h_t(x), \tag{1}$$

from which the final predictor (classifier) is obtained as $H(x) = \text{sgn}\left(g(x)\right)$. The way of combining weak learners successively as in (1) is called a **forward stagewise additive model**, or in short, an **additive model**. The following is its basic framework [2].

Forward stagewise additive model
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ for $i = 1, \cdots, N$
- Total number of rounds: T

- Loss function $L(y, f(x))$
- Base learner model $h(x, \theta)$, where θ is the model parameter

Initialize: Fix $f_0(x) = constant$.
Do for $t = 1, \cdots, T$:

(1) Compute

$$(\lambda_t, \theta_t) = \underset{\lambda,\theta}{\operatorname{argmin}} \sum_i L(y^{(i)}, f_{t-1}(x^{(i)}) + \lambda h(x^{(i)}, \theta))$$

(2) Define

$$f_t(x) = f_{t-1}(x) + \lambda_t h(x, \theta_t)$$

Output: $F(x) = f_T(x)$

This kind of additive model is in contrast with random forests. Recall that random forest aggregates fully grown trees, each of which has low bias but high variance. Thus the principal aim of random forest is to reduce the variance of the final predictor by aggregating such trees all at once. The trick is to make such individual trees as independent (small correlation) as possible, and that is the reason random forest construction is the way it is. On the other hand, AdaBoost does not require an individual classifier (learner) to be as accurate. It only requires each individual learner to be slightly better than a random guess–hence the name weak learner. The gist of AdaBoost is the way it combines such weak learners to make improvement successively.

Historically AdaBoost is the first successful example of additive models. Since then there has been a proliferation of similar ideas, all of which are dubbed boosting. In this lecture, we cover some of the most salient and most widely used boosting methods, in particular, the **gradient boosting**.

2 Exponential loss and AdaBoost

As a way of introduction, let us see how AdaBoost can be cast in the framework forward stagewise additive model presented in Section 1. First, define the loss function

$$L(y, f(x)) = \exp(-yf(x)). \tag{2}$$

For AdaBoost the base learner is $h(x) \in \{+1, -1\}$ and $y \in \{+1, -1\}$. The Step (1) of forward stagewise additive model can be written in the form

$$\begin{aligned}(\lambda_t, h_t) &= \underset{\lambda,h}{\operatorname{argmin}} \sum_i \exp\left(-y^{(i)}[f_{t-1}(x^{(i)}) + \lambda h(x^{(i)})]\right) \\ &= \underset{\lambda,h}{\operatorname{argmin}} \sum_i w_{t,i} \exp(-\lambda y^{(i)} h(x^{(i)})), \end{aligned} \tag{3}$$

where

$$w_{t,i} = \exp(-y^{(i)} f_{t-1}(x^{(i)})).$$

Now (3) can be replaced with the following two stage arguments: First, note that

$$\begin{aligned}\sum_i \exp(-\lambda y^{(i)} h(x^{(i)})) &= e^{-\lambda} \sum_{y^{(i)}=h(x^{(i)})} w_{t,i} + e^{\lambda} \sum_{y^{(i)}\neq h(x^{(i)})} w_{t,i} \\ &= e^{-\lambda} \sum_i w_{t,i} + (e^{\lambda} - e^{-\lambda}) \sum_i w_{t,i} \mathbb{I}(y^{(i)} \neq h(x^{(i)})). \end{aligned} \tag{4}$$

Since the only term depending on h in the line above is the last summation term, h_t of the solution to (3) is gotten by solving

$$h_t = \underset{h}{\operatorname{argmin}} \sum_i w_{t,i} \mathbb{I}(y^{(i)} \neq h(x^{(i)})),$$

regardless of the value of λ as long as $\lambda > 0$. To get λ_t of the solution to (3), we have only to find λ_t that minimizes (4) in which h is replaced with h_t. Since

$$\underset{\lambda}{\operatorname{argmin}}(e^{-\lambda} A + e^{\lambda} B) = \frac{1}{2} \log(A/B),$$

from (4), we get

$$\lambda_t = \frac{1}{2} \log \left(\frac{\sum_{y^{(i)}=h_t(x^{(i)})} w_{t,i}}{\sum_{y^{(i)}\neq h_t(x^{(i)})} w_{t,i}} \right).$$

Define the probability distribution

$$D_t(i) = \frac{w_{t,i}}{\sum_i w_{t,i}}.$$

Then it is easy to see that λ_t can be written as

$$\lambda_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right),$$

where ϵ_t is the error rate as used in AdaBoost as defined by

$$\epsilon_t = \sum_i D_t(i)\mathbb{I}(y^{(i)} \neq h_t(x^{(i)})).$$

Now Step (2) of forward stagewise additive model tells us that

$$f_t(x) = f_{t-1} + \lambda_t h_t(x).$$

At the next stage, the new weight $w_{t+1,i}$ is given as

$$\begin{aligned} w_{t+1,i} &= \exp(-y^{(i)} f_t(x^{(i)})) \\ &= \exp\left(-y^{(i)}[f_{t-1}(x^{(i)}) + \lambda_t h_t(x^{(i)})]\right) \\ &= w_{t,i}\exp\left(-\lambda_t y^{(i)} h_t(x^{(i)})\right) \\ &= w_{t,i} \cdot \begin{cases} e^{-\lambda_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\lambda_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}). \end{cases} \end{aligned}$$

Therefore we have

$$D_{t+1}(i) \propto D_t(i) \cdot \begin{cases} e^{-\lambda_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\lambda_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}), \end{cases}$$

which is exactly the same formula for the change of probability distribution in AdaBoost.

3 Gradient boosting machine

3.1 Residual and gradient

Let $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ be a given dataset for a regression problem, and let $f(x)$ be a regressor we found by some regression algorithm. Its error is

$$E = \sum_i L(y^{(i)}, f(x^{(i)})),$$

where $L(y, f(x))$ is an error (loss) of the regressor f at the input point x. For now, let us assume the L^2 error

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2.$$

Then the error $y^{(i)} - f(x^{(i)})$ is called the **residual** of f at $x^{(i)}$. If we can find a new regressor $h(x)$ such that $h(x^{(i)}) = y^{(i)} - f(x^{(i)})$ for all i, then the new regression function $F(x) = f(x) + h(x)$ would have no error as $F(x^{(i)}) = y^{(i)}$ for all i. It is very rare to find such perfect $h(x)$, so we can try to find $h(x)$ such that $h(x^{(i)}) \approx y^{(i)} - f(x^{(i)})$. If so, the error of $F(x)$ would be smaller than that of $f(x)$.

Note that if

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2,$$

the residual $y - f(x)$ can be written as

$$\widetilde{y} = y - f(x) = -\frac{\partial}{\partial f(x)} L(y, f(x)) \tag{5}$$

Thus these residuals can define a new dataset

$$\widetilde{\mathfrak{D}} = \{x^{(i)}, \widetilde{y}^{(i)}\}_{i=1}^{N},$$

where

$$\widetilde{y}^{(i)} = y^{(i)} - f(x^{(i)}) = -\frac{\partial}{\partial f(x^{(i)})} L(y^{(i)}, f(x^{(i)})) = -\frac{\partial}{\partial f(x)} L(y^{(i)}, f(x)\,)\Big|_{f(x)=f(x^{(i)})} \tag{6}$$

Note $h(x)$ above is the new regressor that fits $\widetilde{\mathfrak{D}}$, i.e. $h(x)$ is the one gotten by using $\widetilde{\mathfrak{D}}$ as a dataset.

3.2 Gradient boosting

The idea in the above section of rewriting the residuals as gradients can be generalized to the following gradient boosting algorithm, which is basically due to Friedman [1].

Gradient Boosting
Input:

- Data: $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ for $i = 1, \cdots, N$
- Total number of rounds: T
- Loss function $L(y, f(x))$
- Base learner model $h(x, \theta)$, where θ is the model parameter

Initialize: Fix $f_0(x)$ by letting $f_0(x) = h(x, \theta_0)$, where

$$\theta_0 = \underset{\theta}{\operatorname{argmin}} \sum_i L(y^{(i)}, h(x^{(i)}, \theta)),$$

or one can simply define $f_0(x) = constant$.
Do for $t = 1, \cdots, T$:

- Define

$$\widetilde{y}^{(i)} = -\frac{\partial}{\partial f_{t-1}(x^{(i)})} L(y^{(i)}, f_{t-1}(x^{(i)}))$$

  and a new dataset

$$\widetilde{\mathfrak{D}} = \{(x^{(i)}, \widetilde{y}^{(i)})\}_{i=1}^{N}$$

- Fit a new base learner $h(x, \theta_t)$ to the dataset $\widetilde{\mathfrak{D}}$
- Compute

$$\lambda_t = \underset{\lambda}{\operatorname{argmin}} \sum_i L(y^{(i)}, f_{t-1}(x^{(i)}) + \lambda h(x^{(i)}, \theta_t))$$

- Define

$$f_t(x) = f_{t-1}(x) + \lambda_t h(x, \theta_t)$$

Output: $F(x) = f_T(x)$

3.3 Various loss functions

Absolute loss

$$L(y, f(x)) = |y - f(x)|$$

Huber loss

$$L(y, f(x)) = \begin{cases} |y - f(x)|^2 / 2 & \text{if } |y - f(x)| \le \delta \\ \delta(|y - f(x)| - \delta/2) & \text{if } |y - f(x)| \ge \delta. \end{cases}$$

3.4 Gradient boosting of classifiers

Gradient boosting is not just for regression. It can be applied equally well to classification problems. To motivate, let us look at AdaBoost again. In a nutshell, AdaBoost tries to find a classifier $f^*(x) \in \{+1, -1\}$ such that

$$\begin{aligned} f^*(x) &= \underset{f(x)}{\operatorname{argmin}} E[e^{-Yf(X)} \,|\, X = x] \\ &= \underset{f(x)}{\operatorname{argmin}} \left[e^{-f(x)} P(Y = 1 \,|\, X = x) + e^{f(x)} P(Y = -1 \,|\, X = x)\right]. \end{aligned}$$

Since

$$\underset{\alpha}{\operatorname{argmin}} \left[e^{-\alpha} P(Y = 1 \,|\, X = x) + e^{\alpha} P(Y = -1 \,|\, X = x)\right] = \frac{1}{2} \log \left(\frac{P(Y = 1 \,|\, X = x)}{P(Y = -1 \,|\, X = x)} \right),$$

we have

$$f^*(x) = \frac{1}{2} \log \left(\frac{P(Y = 1 \,|\, X = x)}{P(Y = -1 \,|\, X = x)} \right).$$

From this, it is easy to see that

$$P(Y = 1 \,|\, X = x) = \frac{1}{1 + e^{-2f^*(x)}} = \frac{e^{f^*(x)}}{e^{-f^*(x)} + e^{f^*(x)}}.$$

To cast this in the Bernoulli framework, define $Y' = (Y + 1)/2 \in \{0, 1\}$. Let $f(x) \in \{0, 1\}$ be a binary classifier and let $p(x) = P(Y = 1 \,|\, X = x)$. Then the log-likelihood of Y' given $p(x)$ is easily seen to be

$$l(Y', f(x)) = Y' \log p(x) + (1 - Y') \log(1 - p(x)) = -\log(1 + e^{-2f(x)}).$$

Equivalently the cross entropy, which is equal to the negative log-likelihood, is

$$H(Y', f(x)) = \log(1 + e^{-2f(x)}).$$

As we have seen in Lecture 3, the minimizer of this cross entropy also leads to the same solution as the one gotten by AdaBoost. For gradient boost, we use this cross entropy as a loss function for binary gradient boost.

3.5 Gradient boosting of multiclass classifier

4 Boosting of trees

References

[1] Friedman,J. *Greedy boosting approximation: a gradient boosting machine*, Ann.Stat. 29, 11891232

[2] Hastie, T., Tibshirani, R, and Friedman, J., *The Elements of Statistical Learning, 2nd edition*, Springer-Verlag (2009)