Boosting

Trong phần tiếp theo, chúng ta sẽ đi qua các thuật toán liên quan đến lớp Boosting.

So với các thuật toán ensemble thuộc lớp bagging thì các thuật toán liên quan đến boosting thường:

- Các learner được training một các tuần tự, learner sau có gắng giảm giá trị loss so với learner trước
- Boosting có xu hướng giảm sử bias của model giữa truth label và predict label, khác với bagging tập trung giảm vairiance của model

1. Adaboost

Adaboost còn có tên gọi khác là Adaptive Boosting là một thuật toán phân loại thống kê được phát minh bởi Yoav Freund và Robert Schapire vào năm 1995. Đầu ra của các weak learner sẽ được tổng hợp theo một trọng số để biểu diễn cho kết quả cuối cùng của bài toán phân loại. Thông thường, Adaboost được sử dụng cho bài toán phân loại nhị phân, tuy nhiên có thể khái quát hóa thành bài toán multiple class

Mô hình thuật toán:

$H_m(x_i) = H_{m-1}(x_i) + \alpha_m * h_m(x_i)$ Trong đó $h_m(x_i)$ là weak learner tại thời điểm m

$\Rightarrow F_m(x_i) = \alpha_1 * h_1(x_i) + \alpha_2 * h_2(x_i) + ... + \alpha_m * h_m(x_i)$

a) Mã giả của Adaboost

Giả sử chúng ta có k cặp điểm dữ liệu $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ với $y_i = \{1, -1\}$

Bước 1: Khởi tạo trọng số cho từng điểm dữ liệu $w_i = 1/m$ với i = 1,2,3,…, m

Bước 2: For m=1 to M:

- Training model h_m bằng tập dữ liệu traning sử dụng trọng số w_m
- Tính toán lỗi

$$\varepsilon_m = \frac{\sum_{i=1 \,\&\, y_i \neq h_m(x_i)}^{N} w_i}{\sum_{i=1}^{m} w_i}$$

- Cập nhật giá trị $\alpha_m = ln(\frac{1-\varepsilon_m}{\varepsilon_m})$
- Cập nhật lại trọng số $w_i^{m+1} = w_i^m * e^{-y_i * \alpha_m * h_t(x_i)}$
- Chuẩn hóa lại giá trị w_i sao cho $\sum_{i=1}^{N} w_i = 1$

Step 3: Output

$F_m(x) = \alpha_1 * h_1(x) + \alpha_2 * h_2(x) + ... + \alpha_m * h_m(x)$

Giải thích mã giả:

+) Training model bằng tập dữ liệu training sử dụng trọng số w_i

Tại thời điểm t, mỗi điểm dữ liệu tương ứng với một trọng số w_i. Giá trị w_i biểu thị giá trị xác suất của điểm dữ liệu i được lựa chọn làm tập dữ liệu training tại thời điểm t, giá trị w_i càng cao thì có khả năng lựa chọn càng cao.

+) Chứng minh công thức α_j:

Loss function: $L_{exp}(x, y) = e^{-yf(x)}$

Nên tổng giá trị loss của training data tại thời interator t là:

$$E = \sum_{i=1}^{N} e^{-y_i * H_m(x_i)} = \sum_{i=1}^{N} e^{-y_i * [H_{m-1}(x_i) + \alpha_m * h_m(x_i)}$$

$$= \sum_{i=1}^{N} e^{-y_i * H_{m-1}(x_i)} * e^{-y_i * \alpha_m * h_m(x_i)})$$

Đặt $w_i^1 = 1$, $w_i^m = e^{y_i * H_{m-1}(x_i)}$. Suy ra:

$$E = \sum_{i=1}^{N} w_i^m * e^{-y_i * \alpha_m * h_m(x_i)} = \sum_{y_i = h_m(x_i)} w_i^m * e^{-y_i * \alpha_m * h_m(x_i)} + \sum_{y_i \neq h_m(x_i)} w_i^m * e^{-y_i * \alpha_m * h_m(x_i)}$$

$$= \sum_{y_i = h_m(x_i)} w_i^m * e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^m * e^{\alpha_m}$$

Do w_i^t cố định

$$\Rightarrow \frac{dE}{d\alpha_m} = -\alpha_m * \sum_{y_i = h_m(x_i)} w_i^m * e^{-\alpha_m} + \alpha_m * \sum_{y_i \neq h_m(x_i)} w_i^m * e^{\alpha_m} = 0$$

$$<==> e^{-\alpha_m} * \sum_{y_i = h_m(x_i)} w_i^m = e^{\alpha_m} * \sum_{y_i \neq h_m(x_i)} w_i^m$$

$$<==> -\alpha_m + \log(\sum_{y_i = h_m(x_i)} w_i^m) = \alpha_m + \log(\sum_{y_i \neq h_m(x_i)} w_i^m)$$

$$<==> \alpha_m = \frac{1}{2} * \log(\frac{\sum_{y_i = h_m(x_i)} w_i^m}{\sum_{y_i \neq h_m(x_i)} w_i^m}) = \frac{1}{2} * \log(\frac{1 - \epsilon_m}{\epsilon_m})$$

2. Gradient Boosting

Gradient boosting là một thuật toán trong machine learning được sử dụng trong 2 bài toán lớn là regression và classification. Ý tưởng của gradient boosting bắt nguồn từ quan sát

của Leo Breiman rằng boosting có thể được diễn giải như là một thuật toán tối ưu trên một hàm chi phí phù hợp nào đó.

2.1 Giới thiệu

Giống như các phương pháp boosting khác, gradient boosting tổng hợp các weak learner thành một strong learner duy nhất theo kiểu lặp đi lặp lại. Vì dụ như trong bài toán regression, mục tiêu của thuật toán là "dạy" cho model F dự đoán giá trị $y^{'} = F(x)$ sao cho giá trị MSE = $\frac{1}{n}\sum(y_{truth} - (y_{predict})\text{^}2$

Giả sử, xem xét thuật toán gradient boosting tại thời điểm m. Model F(x) dự đoán output là y_{pred}. Để cài thiện F(x), thuật toán nên thêm một ước lượng hợp lý $h_m(x)$ sao cho.

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i)$$

Tương được với:

$$h_m(x_i) = F_{m+1}(x_i) - F_m(x_i)$$

Do đó, thuật toán gradient boosting sẽ học $h_m(x)$ tại mỗi stage t, ($h_m(x)$ được xem là residual của $y_i - F_m(x)$

2.2 Thuật toán

Tương tự như nhiều bài toán học giám sát khác, mục tiêu là tìm kiếm một hàm số $F(x)$ xấp xỉ với output đầu ra. Gradient boosting tìm kiếm hàm xấp xỉ F(x) là tổng có trọng số của M function $h_m(x)$:

$$F(x) = \sum_{m=1}^{M} \gamma_m * h_m(x) + const$$

Trong đó $h_m(x)$ được gọi là weak learner mà chúng ta sẽ xây dựng

Đầu vào của thuật toán:

- Tập dữ liệu $\{(x_i, y_i)\}$ với 1 <= i <= n
- Hàm loss L(y, F(x)) khả vi
- Số iterator

Thuật toán:

Bước 1: Khởi tạo model $F_0(x)$ thỏa mãn:

$$F_0(x) = \arg\min_{\gamma} \sum_{i-1}^{n} L(y_i, \gamma).$$

Bước 2: For m=1 to M:

- Tính toán giá trị residual

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

- Fit weak learner, training model thông qua tập dữ liệu $\{(x_i, r_{im})\}$ với 1 <= i <=n
- Tính toán giá trị γ_m sao cho:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

- Cập nhật model:
  $F_m(x) = F_{m-1}(x) + \alpha * \gamma_m h_m(x)$

Bước 3: Output $F_M(x)$

2.3 Chi tiết Gradient boosting trong bài toán Regression

a) Chọn $F_0(x)$

Đối với bài toán regression, hàm loss thông thường được chọn là mean square error (MSE). Nên giá trị khởi tạo $F_0(x) = argmin_{\gamma}$

$$\sum_{i=1}^{n} L(y_i, \gamma) = argmin_{\gamma} \sum_{i=1}^{n} \frac{1}{2} * (y_i - \gamma)^2$$

Suy ra giá trị γ để $F_0(x)$ min khi và chỉ khi $\gamma = \frac{y_1 + y_2 + \ldots + y_n}{n}$

b) Tính r_{im}

$$r_{im} = -\left[\frac{dL(y_i, F(x_i))}{dF(x_i)}\right]_{F(x)=F_{m-1}(x)} = y_{i_{predict}}^{(m)} - y_i$$

2.4 Chi tiết thuật toán Gradient boosting trong bài toán phân loại

Dưới đây, chúng ta sẽ tìm hiểu chi tiết các thành phần của thuật toán gradient boosting trong bài toán nhị phân.

a) Chọn $F_0(x)$

Đối với bài toán phân loại nhị phân, hàm loss thông thường được chọn là binary cross entropy

L = - y * log(p) – (1-y) * log(1-p) với p là giá trị xác suất dự đoán đầu ra

= -y * [log(p) – log(1-p)] – log(1-p)

= -y * log($\frac{p}{1-p}$) – log(1-p)

Đặt log(odd) = log($\frac{p}{1-p}$) p = $\frac{e^{log(odd)}}{1+e^{log(odd)}}$

Suy ra L = -y * log(odd) + log(1 + $e^{log(odd)}$)

b) Tính toán r_{im}

⇨ $\frac{dL}{d\,log(odd)} = -y + \frac{e^{\log log\,(odd)}}{1+e^{\log log\,(odd)}} = p - y$

c) Tính giá trị γ_m

$$\gamma_m = argmin_\gamma \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma * h_m(x)\right)$$

$$= argmin_\gamma - y_i * \left(F_{m-1}(x) + \gamma * h_m(x)\right) + log(1 + e^{F_{m-1}\left(x_i\right)+\gamma * h_m(x)}) \quad (1)$$

Do biểu thức (1) khó cho việc tối ưu bằng cách tính đạo hàm thông thường nên biểu thức (1) sẽ được xấp xỉ theo công thức Taylor đến đạo hàm bậc 2

Đặt $G(\gamma) = \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma * h_m(x)\right)$

Đặt $\gamma' = \gamma * h_m(x)$ $G(\gamma') = \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma'\right)$

Suy ra $G(\gamma') \sim L(y_i,\ F_{m-1}(x_i)) \ + \ \frac{d}{d\,F_{m-1}(x)} L(y_i,\ F_{m-1}(x)) * \ \gamma'$

$$+ \ \frac{d}{d\,F_{m-1}(x)\text{^}2} L(y_i,\ F_{m-1}(x_i)) * \ (\gamma')\text{^}2$$

$$\Rightarrow \ \frac{d\gamma_m}{d\gamma'} = \ \sum_{i=1}^{N} \frac{d}{d\,F_{m-1}(x)} L(y_i,\ F_{m-1}(x_i)) + \sum_{i=1}^{N} \frac{d}{d\,F_{m-1}(x)\text{^}2} L(y_i,\ F_{m-1}(x_i)) * \ \gamma' = 0$$

$$\gamma' = \ \frac{-\sum_{i=1}^{N} \frac{d}{d\,F_{m-1}(x)} L(y_i, F_{m-1}(x_i))}{\sum_{i=1}^{N} \frac{d}{d\,F_{m-1}(x)\text{^}2} L(y_i, F_{m-1}(x_i))} = \ \frac{-\sum_{i=1}^{N} p_i - y_i}{\sum_{i=1}^{N} p_i * (1-p_i)}$$