Gradient Tree Boosting

正田 備也 @ Rikkyo University

2022 年 7 月 1 日

[1] を参考にこの資料を作成した。

1 テイラー展開

$f(x)$ を x_0 のまわりでテイラー展開する。

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots \tag{1}$$

ここで、x を $x + \Delta x$ に、x_0 を x に、それぞれ置き換える。

$$f(x + \Delta x) \approx f(x) + f'(x)((x + \Delta x) - x) + \frac{f''(x)}{2!}((x + \Delta x) - x)^2 + \cdots \tag{2}$$

つまり、

$$f(x + \Delta x) \approx f(x) + \Delta x f'(x) + \frac{1}{2}(\Delta x)^2 f''(x) + \cdots \tag{3}$$

2 Regression trees

- 回帰木の葉の数を T とする。葉とその添字を同一視して、葉の集合を $\{1, \ldots, T\}$ と表すことにする。
- 回帰木の葉には、重みが付与されている。すべての葉の重みをまとめて $w \in \mathbb{R}^T$ と書くことにする。
- インスタンス $\boldsymbol{x} \in \mathbb{R}^d$ から回帰木の葉への写像を $q : \mathbb{R}^d \to T$ とする。q は $\mathbb{R}^d$ の分割を引き起こす。
- 回帰木は、インスタンス $\boldsymbol{x}$ について、葉 $q(\boldsymbol{x})$ の重み $w_{q(\boldsymbol{x})}$ を予測値として出力する。

3 Tree ensemble model

Tree ensemble model $\phi(\boldsymbol{x})$ は、複数の回帰木による予測値の和でもって予測を行う。つまり、i 番目のインスタンス $\boldsymbol{x}_i$ についての予測値 $\hat{y}_i$ は、次のように表される。

$$\hat{y}_i = \phi(\boldsymbol{x}_i) = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i) \tag{4}$$

ただし、K は回帰木の個数であり、個々の回帰木は $f_k(\boldsymbol{x})$ と表記している。

4 Learning objective

目的関数には、木が複雑になりすぎないように、正則化を用いる。損失関数を $l(y, \hat{y})$ とすると、

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \quad \text{where} \quad \Omega(f) = \gamma T + \tfrac{1}{2}\lambda \|w\|^2. \tag{5}$$

5 Gradient tree boosting

Gradient tree boosting では、additive に回帰木を training していく。t 番目の回帰木の与える予測値 $f_t(\boldsymbol{x}_i)$ は、以下のように、$t-1$ 番目の回帰木による予測値 $\hat{y}_i^{(t-1)}$ に加算される。

$$\mathcal{L}^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)) + \Omega(f_t) \tag{6}$$

ここで、テイラー展開を使って損失関数 $\mathcal{L}^{(t)}$ を近似する。

$$\tilde{\mathcal{L}}^{(t)} \simeq \sum_i \Big[l(y_i, \hat{y}_i^{(t-1)}) + f_t(\boldsymbol{x}_i) g_i + \frac{1}{2} f_t^2(\boldsymbol{x}_i) h_i \Big] + \Omega(f_t) \tag{7}$$

ただし、$g_i \equiv \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$、また、$h_i \equiv \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)^2}}$ と定義した。例えば、$l(y, \hat{y}) = (y - \hat{y})^2$ というように損失関数を選ぶ場合、$g_i = -2(y - \hat{y}_i^{(t-1)})$ であり、$h_i = 2$ である。

t 番目の木を求めるための最適化計算に関係しない定数 $l(y_i, \hat{y}_i^{(t-1)})$ を削除し、$\Omega(f_t)$ と $f_t(\boldsymbol{x})$ とを展開して書き直すと、

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &\simeq \sum_{i=1}^{n} \Big[f_t(\boldsymbol{x}_i) g_i + \frac{1}{2} f_t^2(\boldsymbol{x}_i) h_i \Big] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \Big[\big(\sum_{i \in I_j} g_i \big) w_j + \frac{1}{2} \big(\sum_{i \in I_j} h_i + \lambda \big) w_j^2 \Big] + \gamma T \end{aligned} \tag{8}$$

ただし、I_j は、葉 j に属するインスタンスの集合 $I_j \equiv \{i : q(\boldsymbol{x}_i) = j\}$ と定義した。

式 (8) より、$\mathcal{L}^{(t)}$ は w_j の 2 次関数になっている。よって、$\mathcal{L}^{(t)}$ を最大にする w_j は、以下のとおり。

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{9}$$

このときの目的関数の値は

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{10}$$

ここで q が登場しているのは、この値を、特定の木構造 q のスコアと見做すことができるからである。

Gradient tree boosting では、このスコアが大きくなるように、木を枝分かれさせていくことになる。

木を枝分かれさせるアルゴリズムは様々考えられる。論文 [1] を参照。

参考文献

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. New York, NY, USA, 2016. Association for Computing Machinery.