

Gradient Boosting: Regression + Classification

Gradient Boosting: Regression, Classification

Regression 1.

Input: Data $\{(x_i, y_i)\}_{i=1}^n$, and a differentiable loss function $L(y_i, F(x))$
 for regression: $L(y_i, F(x)) = \frac{1}{2}(y_i - F(x))^2$

Step 1. Initialize a model with a constant value $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$

Step 2. For $m=1, \dots, M$:

(A) Compute $r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right] \quad F(x) = F_{m-1}(x)$ for $i=1, \dots, n$
 pseudo residuals

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}
 for $j=1, \dots, J_m$
 we build a regression tree to predict the residuals instead of the weights.

(C) For $j=1, \dots, J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \gamma \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}\{x \in R_{jm}\}$
 learning rate $\gamma \in (0, 1)$

Step 3. Output $F_M(x)$

Classification 2.

Input: Data $\{(x_i, y_i)\}_{i=1}^n$, differentiable loss function $L(y_i, F(x))$

$-ll(\text{Data}) = - \sum_{i=1}^n y_i \cdot \log(p) + (1-y_i) \log(1-p)$
 log-likelihood
 negative
 $L(y_i, F(x)) \uparrow$

J_m - leafs

Step 1. Initialize a model with a const. value: $\underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$

Step 2. For $m=1, \dots, M$:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm} , for $j=1, \dots, J_m$

(C) For $j=1, \dots, J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$

Solving loss for an optimal γ is a mess. Thus, we can approximate with a Taylor expansion of L over γ .

$$L(y_1, F_{m-1}(x_1) + \gamma) \approx L(y_1, F_{m-1}(x_1)) + \frac{dL}{dF}(y_1, F_{m-1}(x_1)) \gamma + \frac{1}{2} \frac{d^2L}{dF^2}(y_1, F_{m-1}(x_1)) \gamma^2$$

Basically $\frac{dL(y_i, F_{m-1}(x_i) + \gamma)}{dF} \leftarrow$ 2nd-order Taylor polynomial
 $\frac{d^2L(y_i, F_{m-1}(x_i) + \gamma)}{dF^2} \leftarrow$ grad of the old loss + current prediction
 $\frac{dL(y_i, F_{m-1}(x_i) + \gamma)}{dF^2} \leftarrow$ hessian of old loss + $\gamma \uparrow$

Especially XGBoost does it like that

$$\frac{d}{d\gamma} L(y_1, f_{m-1}(x_1) + \gamma) \approx \frac{d}{dF(\cdot)} L(y_1, f_{m-1}(x_1)) + \frac{d^2 L}{dF^2}(y_1, f_{m-1}(x_1)) \gamma$$

Set it to 0 → solve for γ : $\gamma = -\frac{\frac{d}{dF(\cdot)} L(y_1, f_{m-1}(x_1))}{\frac{d^2 L}{dF^2}(y_1, f_{m-1}(x_1))}$ Can be solved analytically

$$(D) \text{ Update } f_m(x) = f_{m-1}(x) + \gamma \sum_{j=1}^{J_m} \gamma_j m \mathbb{1}\{x \in R_j\}$$

Step 3. Output $f_m(x)$

L_1, L_2 Regularization (Lasso, Ridge)

1. Ridge Regression

Ridge regression = penalty for least squares fit = $\lambda \cdot \text{slope}^2$ It adds some bias, but λ = smaller variance
Given $D = \{(x_i, y_i)\}_{i=1}^n$

$$L(y, \hat{y}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \beta^2 \quad \text{features}$$

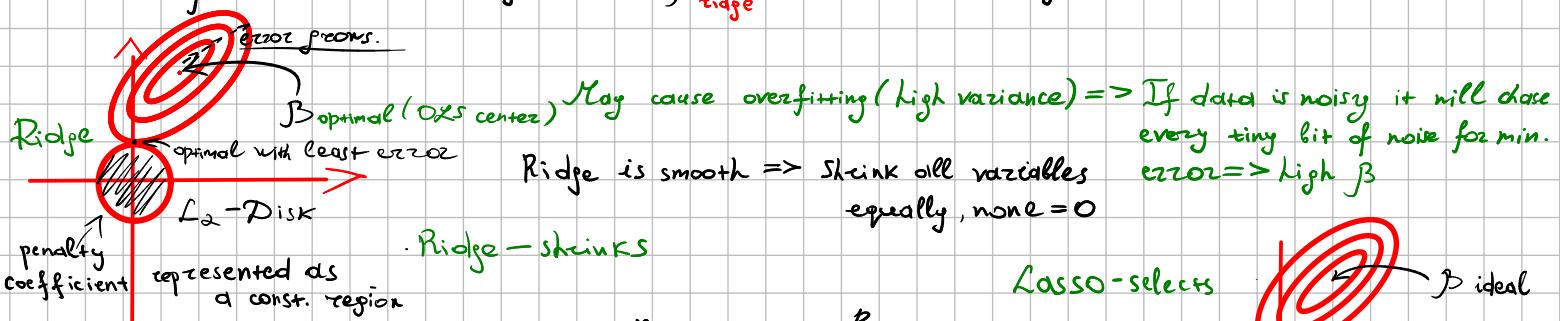
Matrix notation: $\text{RSS}(\beta) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$ regularizer \Rightarrow will penalize large β 's.
For an optimal solution: $\min_{\beta} \text{RSS}(\beta)$

$$\text{Expanding loss } y^T y - y^T X \beta - X^T \beta^T y + X^T \beta^T X \beta + \lambda \beta^T \beta$$

$$\text{Now for min: } \frac{\partial \text{RSS}}{\partial \beta} \Big|_{\beta} = 0 \Rightarrow 0 - \underbrace{y^T X - X^T y}_{= -2 y^T X} + X^T X \cdot 2\beta + 2\lambda \beta = 0 \Rightarrow -y^T X + X^T X \hat{\beta} + \lambda \beta = 0$$

$$\hat{\beta} = \min_{\beta} \underbrace{\beta^T X^T X \beta + 2\lambda \beta}_{\text{error grows.}}$$

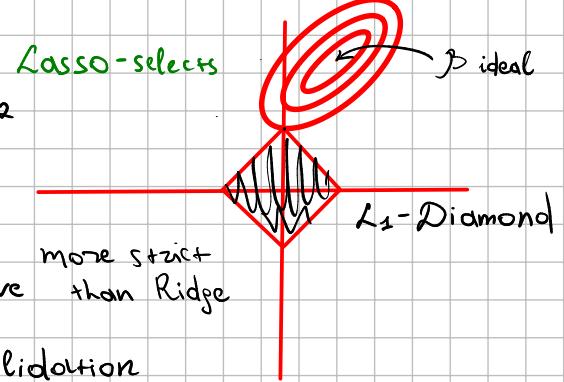
$$\hat{\beta} = (X^T X + 2I)^{-1} y^T X$$



$$\text{Actually, the ridge problem is } \hat{\beta}_{\text{ridge}} = \arg \min_{\beta} \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2$$

$$\text{s.t. } \sum_{j=1}^p \beta_j^2 \leq t, \text{ solving it by}$$

1. Lagrangian
2. KKT \rightarrow objective more strict than Ridge



2) Lasso Regression

Our penalty term: $\lambda \cdot |\text{slope}|$ L_1 -norm ; λ - use Cross-Validation

Ridge and Lasso do the same thing, they make our model less sensitive.

Lasso is more strict, some features = 0

$RSS(\beta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|\beta\|_1$ We don't have a closed-form analytical solution like in Ridge.

$\|\beta\|_1$ is not differentiable at 0. We want to minimize $L(y, \hat{y}) = \|X\beta - \hat{y}\|_1 = \sum_{i=1}^n |y_i - \sum_{j=1}^p X_{ij}\beta_j| + \lambda \|\beta\|_1$

Then, we do: 1. Compute the partial residual $r_i = y_i - \sum_{k \neq j} X_{ik}\beta_k$; 2. Compute the "strength":

\rightarrow of feature j $\rho_j = \sum_{i=1}^n X_{ij}(y_i - \hat{y}_i)$; 3. apply soft thresholding: $\beta_j = S(\rho_j, \lambda) = \rightarrow \rightarrow$

$\rightarrow \begin{cases} (\rho_j - \lambda)/\sum X_{ij}^2, & \text{if } \rho_j > \lambda \\ (\rho_j + \lambda)/\sum X_{ij}^2, & \text{if } \rho_j < -\lambda \\ 0, & \text{if } |\rho_j| \leq \lambda \end{cases}$ We can transform the objective into a more convenient form:

$$\min_{\beta} L(y, \hat{y}) = \frac{1}{2n} \sum_{i=1}^n \sum_{j=1}^p (y_i - \sum_{k=1}^p X_{kj}\beta_k)^2 + \lambda \|\beta\|_1$$

for lasso: $\hat{\beta}_{\text{lasso}} = \arg \min_{\beta} \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \alpha_{ij}\beta_j)^2$ for simpler differentiation, possible to do cause min is the same
st. $\sum_{j=1}^p |\beta_j| \leq \epsilon$

XGBoost: Regression, Classification A -дискретн., d_N -континуальна на $\alpha_{N-1}(x_i)$

На каждой итерации шаг будем сдвигать вектор сглажив s , - показывает как корректиру x_i для α_{N-1} чтобы как можно меньше изменить α_{N-1} :

После этого нового базового алгоритма $b_N(x)$ обуславливает градиент минимизации MSE or вектора сглажив s :

вектор сглажив $\vec{s} = \nabla_{\vec{x}} \alpha_{N-1}(x_i)$ просто шаг. некое обозначение

$$s = \left(-\frac{\partial L}{\partial z} \Big|_{z=\alpha_{N-1}(x_i)} \right)_{i=1}^l = -\nabla_{\vec{x}} \sum_{i=1}^l L(y_i, z_i) \quad z = \alpha_{N-1}(x_i)$$

Форма задачи: $\sum_{i=1}^l L(y_i, \alpha_{N-1}(x_i) + b(x_i)) \rightarrow \min_b$

Разложение в шаг Тейлора: $\sum_{i=1}^l L(y_i, \alpha_{N-1}(x_i) + b(x_i)) \approx \sum_{i=1}^l (L(y_i, \alpha_{N-1}(x_i)) - s_i b(x_i) + \frac{1}{2} h_i b^2(x_i))$

$h_i = \frac{\partial^2 L(y_i, z)}{\partial z^2} \Big|_{z=\alpha_{N-1}(x_i)}$ В контексте задачи оптимизации, h -и нулю не равно -> убираем

Задачу можно теперь представить так, $\sum_{i=1}^l (-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)) \rightarrow \min_b$

Он похож на MSE, $\sum_{i=1}^l (b(x_i) - s_i)^2 = \sum_{i=1}^l (b^2(x_i) - 2b(x_i)s_i + s_i^2) \Rightarrow \sum_{i=1}^l (b^2(x_i) - 2b(x_i)s_i)$ можно преобразовать еще раз, $= 2 \sum_{i=1}^l (-s_i b(x_i) + \frac{1}{2} b^2(x_i)) \rightarrow \min_b$ не зависит от b можно убрать

То есть, мы увидели что разложение в шаг Тейлора - хороший аппроксиматор. Здесь $h_i = 1$. Мы отображаем форму о второй производной, используя аппрокс. 2-го порядка, то есть считаем что функция имеет одинаковую кривизну во всем направлении

Регуляризация

Будем работать с ординационами $\sum_{i=1}^l (-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)) \rightarrow \min_b$

Сложность зависит от:

1: Число членов $J \rightarrow$ больше параметров

$\left\{ \begin{array}{l} \text{overfitting} \\ \text{если } J > n \end{array} \right.$

2: Норма коэффиц. b членов

$\|\beta\|_2^2 = \sum_{j=1}^p b_j^2$, чем меньше $> 0 \Rightarrow$ лучше беда у алгоритма

Добавим регуляризаторы: $\sum_{i=1}^l (-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)) + \gamma J + \frac{1}{2} \sum_{j=1}^p b_j^2 \rightarrow \min_b$ L_2 -регуляризация, убирает однозначно

или какого-то R_j и общий b норм - ограничение предсказаний (если $b(x_i) \rightarrow b_j$)

изменяет L после добавления нового $b(x_i)$

НО! не штрафует

В классическом град. будим мы при этом перебор (max-depth), но можно поработать иначе.

где мат. угодства

Мы можем упростить оружийную и суммировать по всем листьям (J), а не по всей выборке i -тым деревом

Получим: $\sum_{j=1}^J \left\{ \left(-\sum_{i=1}^l s_i \right) \cdot b_j + \frac{1}{2} \left(\lambda + \sum_{i=1}^l h_i \right) b_j^2 + \gamma \right\} \rightarrow \min_b$

Kinda like $ax^2 + bx + c = \text{fix}$

Каждый терм можно минимизовать независимо по b_j . Весь этот оружийный - параллельно с $a > 0$ - ведь b_j ноль

Значит оптимальный $b_j = -\frac{b}{2a} \Rightarrow b = -\sum_{i=1}^l s_i = S_j$, $a = \frac{1}{2} \left(\lambda + \sum_{i=1}^l h_i \right) = \frac{1}{2} (\lambda + H_j) \Rightarrow$

$\Rightarrow b_j = \frac{S_j}{2 \cdot \frac{1}{2} (\lambda + H_j)} = \frac{S_j}{\lambda + H_j}$ Или оружийная формула: $\sum_{j=1}^J \left\{ -S_j b_j + \frac{1}{2} (\lambda + H_j) b_j^2 + \gamma \right\} \rightarrow \min_b$

Упрощаем получившуюся оптимальную b_j : $\sum_{j=1}^J \left\{ l - S_j \cdot \frac{S_j}{\lambda + H_j} + \frac{1}{2} (\lambda + H_j) \cdot \frac{S_j^2}{(\lambda + H_j)^2} + \gamma \right\} \rightarrow \min_b$

Получаем, $\sum_{j=1}^J \left\{ \frac{-S_j^2}{\lambda + H_j} + \frac{1}{2} \frac{S_j^2}{\lambda + H_j} + \gamma \right\} \Rightarrow \sum_{j=1}^J \left(\frac{-2S_j^2 + S_j^2}{2(\lambda + H_j)} + \gamma \right) = \frac{1}{2} \sum_{j=1}^J \frac{-S_j^2}{H_j + \lambda} + \gamma J = H(J)$

Объяснение $b(x_i)$ - Decision Tree

Ошибки дерева $b(x_i)$ с оптимальным b_j в матрице

$H(b) = \min L(y_i, \alpha_{n+1}(x_i) + b(x_i))$

С его помощью можно строить дерево $b(x_i)$

То есть можно сказать снижение оружийных потерь в узле R = выигрыш информации

Значит наша задача максимизировать Gain - прирост информации = G

$G = H(R) - H(R_l) - H(R_r) \rightarrow \max$ Выбираем такое разбиение чтобы прирост информации (снижение потерь) $\rightarrow \max$

$H(R) = \frac{1}{2} \left[\left(\sum_{(h_i, s_i) \in R} s_i \right)^2 / \left(\lambda + \sum_{(h_i, s_i) \in R} h_i \right) + \gamma \right]$

Особенности XGBoost: 1. Базовый алгоритм приносит направление, посчитанное с учетом вторых производных оружийных потерь; 2. Регуляризация оружийной, L_2 для весов + λ , и γ для конст. листьев J ; 3. При построении дерева используется критерий информативности, зависящий от оптимального веса (антиградиента);

Стекинг.

Построение ансамбля $\xrightarrow{\text{Базинг}} \text{Бустинг} \xrightarrow{\text{Стекинг}}$

из выборки X делаем K моделей

Независимо обучим K базовых алгоритмов $b_1(x), \dots, b_K(x)$ на выборках X_1, \dots, X_K . Теперь хотим обучить мета-алгоритм $\alpha(x)$ на прогнозах базовых моделей.

$b_j^{-K}(x)$ - базовый алгоритм, обученный по всем блокам кроме K -го

Функционал будет таким: $\sum_{k=1}^K \sum_{(x_i, y_i) \in X_k} \lambda [y_i, \alpha(b_1^{-K}(x_i), \dots, b_K^{-K}(x_i))] \rightarrow \min_{\alpha}$

В таком случае мы избежим переобучения, мы тестим ошибку на объекте x_i который лежит в k -ом блоке, то есть b_j^{-K} - это объект при обучении не видел

Категориальные и Текстовые признаки!

Для ансамблей под деревьями - полезно! - TF-IDF + Binary Encoding = Оригинальная разнородность

Гешеши: (текущий, где $\alpha(x)$ обучается зря). Бустинг, а какими кат./текст. признаки = одно число базовых алгоритмов
Например: для кат. признаков = count - b_j , или линейные модели.