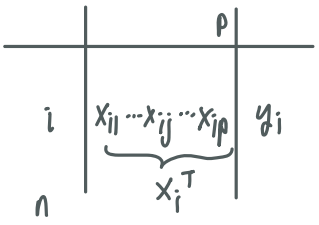
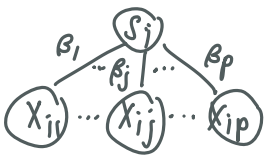
Last week

linear regression

touched logistic





$$\beta = \begin{pmatrix} \beta_1 \\ \beta_j \\ \beta_p \end{pmatrix}$$

$$s_i = \sum_{j=1}^{p} x_{ij}\beta_j = \langle x_i, \beta \rangle = x_i^T \beta$$

linear regression

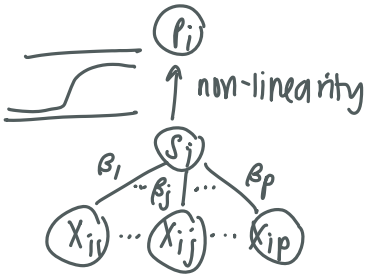
$$y_i \sim N(s_i, \sigma^2)$$

logistic regression

$$y_i \sim \text{Bernoulli}(p_i)$$

$$P(y_i = 1 | s_i) = p_i$$

$$P(y_i = 0 | s_i) = 1 - p_i$$



$$p_i = \text{sigmoid}(s_i) = \frac{e^{s_i}}{1 + e^{s_i}}$$

learning objective function

maximum likelihood

$$\text{Likelihood}(\beta) = \prod_{i=1}^{n} P(y_i | s_i)$$

$$P(y_i | p_i) = p_i^{y_i} (1 - p_i)^{(1-y_i)}$$

$$p_i \begin{cases} p_i & y_i = 1 \\ 1 - p_i & y_i = 0 \end{cases}$$

$$\log p_i = s_i - \log(1 + e^{s_i})$$

$$\log(1 - p_i) = -\log(1 + e^{s_i})$$

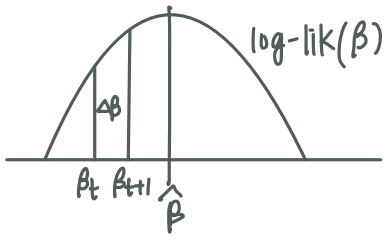
$$\text{log-likelihood}(\beta) = \sum_{i=1}^{n} \log P(y_i | s_i)$$

$$= \sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

$$= \sum_{i=1}^{n} \underbrace{\left[y_i s_i - \log(1 + e^{s_i}) \right]}_{L(s_i)}$$

→ to approximate function to quadratic form, apply 2nd order Taylor expansion

$$\text{least squares} = \sum_{i=1}^{n} (y_i - s_i)^2$$



iterated reweighted least squares (IRLS)

$$L(s_i) \doteq L(\hat{s}_i) + L'(\hat{s}_i)\Delta s_i + \frac{1}{2}L''(\hat{s}_i)\Delta s_i^2$$

2nd order Taylor expansion
surrogate quadratic function that we want to optimize

$s_i = \hat{s}_i + \Delta s_i$ ← constant

↓ fixed

start at $\beta_0 = 0$

at iteration t β_t

$\hat{s}_i = x_i^T \beta_t$

when β is changed by $\Delta\beta$, you change s_i by Δs_i

$\beta_{t+1} = \beta_t + \Delta\beta$

$s_i = x_i^T \beta_{t+1} = \hat{s}_i + \Delta s_i \qquad \Delta s_i = x_i^T \Delta\beta$

$$L'(s_i) = y_i - \frac{e^{s_i}}{1+e^{s_i}}$$

$$= \boxed{y_i - p_i = e_i} \quad \leftarrow \text{residual}$$

$$L''(s_i) = \frac{\partial}{\partial s_i}\left(-\frac{e^{s_i}}{1+e^{s_i}} + 1\right)$$

$$= \frac{\partial}{\partial s_i}\left(\frac{1}{1+e^{s_i}}\right)$$

$$= -\frac{e^{s_i}}{(1+e^{s_i})^2} = \boxed{-p_i(1-p_i) = -w_i}$$

$L(s_i) = L(\hat{s}_i) + \hat{e}_i \Delta s_i - \frac{1}{2}\hat{w}_i \Delta s_i^2$ all e_i & w_i are calculated from $\hat{s}_i$

$$= -\frac{1}{2}\hat{w}_i\left[\Delta s_i^2 - 2\frac{\hat{e}_i}{\hat{w}_i}\Delta s_i\right] + \text{constant}_1$$

$$= -\frac{1}{2}\hat{w}_i\left[\frac{\hat{e}_i}{\hat{w}_i} - \Delta s_i\right]^2 + \text{constant}_2$$

$\frac{\hat{e}_i}{\hat{w}_i} \rightarrow \hat{y}_i$ (working response)

$\Delta s_i \rightarrow x_i^T \Delta\beta$

$$\log\text{-lik}(\beta) = \underbrace{-\sum_{i=1}^{n} \hat{w}_i \left(\hat{y}_i - x_i^T \Delta\beta\right)^2}_{\text{Loss}}$$

← iteratively reduce error to get better prediction.

w_i gets changed at each iteration

note: this is the univariate scenario (2nd order Taylor expansion)

(Recall linear reg $\sum_{i=1}^{n} (y_i - x_i^T \beta)^2$)

minimize this (equiv to maximizing surrogate log lik due to the negative sign)

$$\frac{d\text{loss}}{d\Delta\beta} = -\sum_{i=1}^{n} x_i \hat{w}_i (\hat{y}_i - x_i^T \Delta\beta) = 0$$

↑ derivative of $[-x_i^T \Delta\beta]$ term

$$= \sum_{i=1}^{n} \hat{w}_i x_i \hat{y}_i - \sum_{i=1}^{n} \hat{w}_i x_i x_i^T \Delta\beta$$

$$\Delta\beta = \left(\sum_{i=1}^{n} \hat{w}_i x_i x_i^T\right)^{-1} \left(\sum_{i=1}^{n} \hat{w}_i x_i \hat{y}_i\right)$$

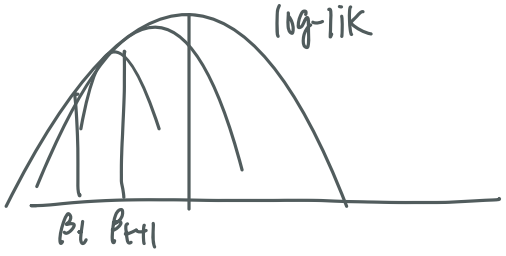
stop when $\Delta\beta \approx 0$

use converged β to make prediction

$\beta_0 \cdots \beta_t \rightarrow \hat{s}_i \rightarrow \hat{e}_i, \hat{w}_i \rightarrow \hat{y}_i \rightarrow \Delta\beta$

$\beta_{t+1} = \beta_t + \Delta\beta$

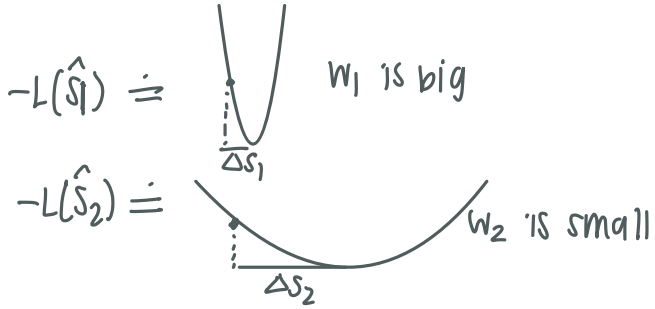
at each iteration, you make up for the created error to eventually have $\Delta\beta \approx 0$



XG boost in boosting tree (powerful classification method)

add new trees to $B = 1000$ to fit the data $\hat{Y}$

to make more accurate predictions.



if w_i is big, only a small change is necessary in Δs_i (large curvature = small dist)

w_i is large when $p_i = \frac{1}{2}$ (very uncertain)

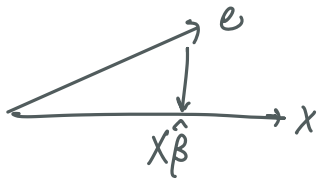
overfitting

simplest linear model

i	x_i	y_i

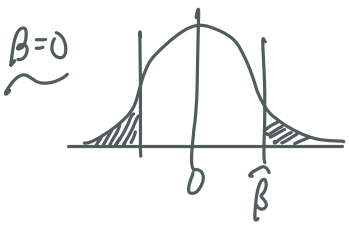
$y_i = x_i\beta + e_i \quad e_i \sim N(0, \sigma^2)$

$\hat{\beta} = \frac{\langle x, y \rangle}{|x|^2}$

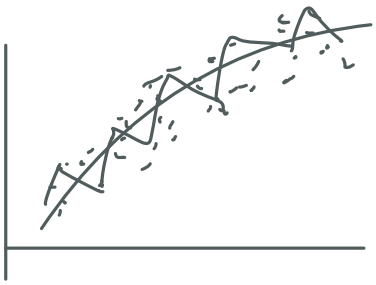


if true value of $\beta = 0$, $\hat{\beta} = \frac{\langle x, e \rangle}{|x|^2} \neq 0$

model absorbs the noise



hypothesis testing for $\hat{\beta}$ helps avoid overfitting.



need regularization to avoid overfitting in ML.