Review of Linear Regression.

$y = X\beta + \varepsilon$. $\quad \varepsilon \sim N_n(0, \sigma^2 I)$.

$y = (y_1, \cdots, y_n) \quad \in \mathbb{R}^n$ response

$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \quad \in \mathbb{R}^{n \times (p+1)}$ covariates.

$\beta = (\beta_0, \beta_1, \cdots, \beta_p) \quad \in \mathbb{R}^{(p+1)}$ parameter

$\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n) \quad \in \mathbb{R}^n$. error

Data: $\{(y_i, x_i)_{i=1}^n\}$

Estimation (Training):

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 = \arg\min_{\beta} (y - X\beta)^T (y - X\beta).$$

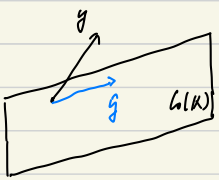
$L(\beta) = (y - X\beta)^T (y - X\beta)$. $\quad \frac{\partial L}{\partial \beta} = 0 - 2X^T y + X^T X \beta = 0 \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$.

Prediction:

$$\hat{y} = X\hat{\beta} = \underbrace{X(X^T X)^{-1} X^T}_{H} y = Hy.$$

Interpretation:

$\hat{y}$ is orthogonal projection of y onto $\mathcal{C}(X)$. i.e. $\hat{y} = \text{proj}_{\mathcal{C}(X)}(y)$.



Why? $\mathcal{C}(X) = \{w = X\beta : \beta \in \mathbb{R}^{p+1}\}$.

since $\hat{\beta}$ is minimizer of $L(\beta) = (y - X\beta)^T (y - X\beta)$.

For any $\tilde{\beta} \in \mathbb{R}^{p+1}$, $L(\tilde{\beta}) \geq L(\hat{\beta}) \Rightarrow X\hat{\beta}$ is indeed orthogonal proj.

Logistic Regression.

Data: $\{(y_i, x_i)_{i=1}^n\}$. $y_i \in \{0, 1\}$. $x_i \in \mathbb{R}^p$

Optimal Bayes Classifier.

$$g_{bayes}(x) = \arg\max_k P(y=k \mid x) \qquad k = \{0, 1\}$$

Optimal in the sense of posterior probability.

Goal: Model $P(y=1 \mid x)$

$y \in \{0, 1\} \Rightarrow y \mid X \sim Bin(1, \pi)$. $\pi \in (0, 1)$.

We will model π:

$$logit(\pi) = \log\left(\frac{\pi}{1-\pi}\right) = x^T\beta. \qquad (\text{In MLR: } y = x^T\beta).$$

Estimation can be done via MLE:

$$l(\pi) = \sum_{i=1}^n y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i) = \sum_{i=1}^n y_i \log\left(\frac{\pi_i}{1-\pi_i}\right) + \log(1-\pi_i).$$

and $\log\frac{\pi}{1-\pi} = x_i^T\beta$.

$$\Rightarrow \pi = \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}$$

$$\Rightarrow \log 1-\pi = -\log\{1+\exp(x_i^T\beta)\}.$$

$$= \sum_{i=1}^n y_i \cdot x_i^T\beta - \log\{1+\exp(x_i^T\beta)\}.$$

$$l(\beta) = \sum_{i=1}^n y_i \cdot x_i^T\beta - \log\{1+\exp(x_i^T\beta)\}.$$

$$\hat{\beta} = \arg\max_\beta l(\beta).$$

$$// \quad expit(x_i^T\beta)$$

Prediction: $logit(\pi_i) = x_i^T\beta \Rightarrow \hat{\pi}_i = \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)} = P(y_i=1 \mid x_i)$

For a new unit with covariate x_{new}, we can do classification by:

$$g_{bayes}(x_{new}) = \mathbb{I}\left(expit(x_{new}^T\hat{\beta}) \geq 1/2\right)$$

Numerical calculation:

In R, we implement logistic regression by "glm" function.

y - response.

X - covariate.

Model fitting:

fit = glm(y ~ X , family = "binomial").

coef = fit $ coef .

How do we know if our model is good?

Suppose $y \in \mathcal{Y}$. $X \in \mathcal{X}$.

def. loss function $L: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ s.t. $L(y, y') \geq 0$. $L(y, y') = 0 \iff y = y'$.

Commonly adopted loss function:

1. l_2 loss: $L(y, y') = (y - y')^2$

2. 0-1 loss: $L(y, y') = I(y \neq y')$.

def. Training error: suppose we have data D, and we trained a classifier $\hat{f}: \mathcal{X} \to \{0, 1\}$.
Then, the training err is:

$$\overline{err} = \frac{1}{|D|} \sum_{i: x_i \in D} L(y_i, \hat{f}(x_i)).$$

def. Generalization Error: suppose we have data D, and we trained a classifier $\hat{f}: \mathcal{X} \to \{0, 1\}$.
Then, the generalization err is:

$$err = E(L(y, \hat{f}(x)))$$

Suppose further, we have a test set T, then, empirically, this is $\frac{1}{|T|} \sum_{i: x_i \in T} L(y_i, \hat{f}(x_i))$

Remark: Danger of using training err to determine model performance:
Suppose we have data $(x_i, y_i)_{i=1}^n$. Define $f(x) = \begin{cases} y_i & \text{if } x = x_i \\ 0 & \text{o.w.} \end{cases}$

Then, training error = 0. but obviously, this is very bad classifier. It gives no guarantee on test error.

It is important to do "Training-Testing" split. to evaluate the performance of a model.

Suppose we have model $D = (x_i, y_i)_{i=1}^{n}$.

Let $D = D_{train} \cup D_{test}$.

⇒ Train a classify using D_{train}.

⇒ Evaluate the performance by: $\frac{1}{|D_{test}|} \sum_{i: x_i \in D_{test}} L(y_i, \hat{f}(x_i))$.

- Avoid overfitting.