

Classification: basics

Wei Li

Syracuse University

Spring 2024

OVERVIEW

Binary Classification

Multi-class Classification

LDA and QDA

Logistic Regression: binary case

Logistic Regression: multi-class case, sparsity

Others: nonparametric classifier, robust loss, viewpoint from perceptrons

Binary Classification

Sensitivity and Specificity

Imagine a scenario where people are tested for a disease:

- ▶ The test outcome: positive (sick) or negative (healthy)
- ▶ The actual status: positive (sick) or negative (healthy)

There are four possible scenarios:

- ▶ True positive (TP): sick people correctly identified as sick
- ▶ False positive (FP) : healthy people incorrectly identified as sick
- ▶ True negative (TN): healthy people correctly identified as healthy
- ▶ False negative (FN): sick people incorrectly identified as healthy

True outcome	Test Outcome		Total
	Positive	Negative	
Positive	True Pos. (TP)	False Neg. (FN)	P
Negative	False Pos. (FP)	True Neg. (TN)	N
	P^*	N^*	

True outcome	Test Outcome		Total
	Positive	Negative	
Positive	True Pos. (TP)	False Neg. (FN)	P
Negative	False Pos. (FP)	True Neg. (TN)	N
	P^*	N^*	

Accuracy (1-Error): $(TP + TN)/(TP + TN + FP + FN)$.

Sensitivity (true positive rate/power/recall): the proportions of positives that are correctly identified

$$\text{Sensitivity} = TP/P = TP/(TP + FN)$$

Specificity (true negative rate): the proportions of negative that are correctly identified

$$\text{Specificity} = TN/N = TN/(FP + TN)$$

- ▶ False positive rate (Type I error): $= 1 - \text{Specificity}$.
- ▶ False negative rate (Type II error): $= 1 - \text{Sensitivity}$
- ▶ False discovery rate (FDR/precision): the proportion of predicted positives that are in fact false positives $FDR = FP/P^*$

example 1

	Predicted	
True	email	spam
email	573	40
spam	53	334

spam = presence of disease, email = absence of disease

$$\begin{aligned}\text{specificity} &= 100 \times \frac{573}{573+40} = 93.4\% \\ \text{sensitivity} &= 100 \times \frac{334}{334+53} = 86.3\%\end{aligned}$$

example 2

Threshold A is chosen to balance sensitivity and specificity without leaning too heavily towards either.

True Positive (TP)	False Negative (FN)
40	5
False Positive (FP)	True Negative (TN)
10	45

With Threshold A, we have: - Sensitivity: 88.9% - Specificity: 81.8%

Threshold B (it has a lower criterion for a positive test result)–adjusted to make the test more sensitive to detecting the disease.

True Positive (TP)	False Negative (FN)
45	0
False Positive (FP)	True Negative (TN)
20	35

With Threshold B, we have: - Sensitivity: 100% - Specificity: 63.6%

Binary classification: problem

- ▶ input vector $X \in \mathcal{X} \subset \mathbb{R}^d$
- ▶ output $Y \in \{0, 1\}$
- ▶ “hard classifier” $h : \mathcal{X} \longrightarrow \{0, 1\}$

The rule is characterized as

$$h(X) = 1(b(X) > 0)$$

where b is the boundary function (or discriminant function) that gives the decision boundary $\{x : b(x) = 0\}$.

- ▶ If $b(X)$ is a linear in X , then the classifier has a linear boundary (in X -space).
 - ▶ With transformed X included, the classifier can have a nonlinear boundary (in X -space).

Examples of linear boundary

Linear logit model:

Assume that the **logit function** is linear in x , i.e.,

$$b(x) = \log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = 0 \mid X = x)} = \beta_0 + \beta_1^\top x$$

Thus the classification boundary is given by $\{x : \beta_0 + \beta_1^\top x = 0\}$

Examples: LDA, Logistic regression (see later)

The classification error rate, of h is defined as

$$R(h) = E_{X,Y}(1(Y \neq h(X))) = P(Y \neq h(X))$$

The rule h that minimizes $R(h)$ is

$$h^*(x) = \begin{cases} 1 & \text{if } m(x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

where $m(x) = P(Y = 1 \mid X = x) = E(Y \mid X = x)$.

- ▶ This optimal rule is called the **Bayes rule (classifier)** (under equal costs).
- ▶ The risk $R(h^*)$ is called the **Bayes risk**.
- ▶ The set $\{x : m(x) - \frac{1}{2} = 0\}$ is called the **Bayes decision boundary**.

Alternatively, the Bayes rule is h^* , is given by

$$h^*(x) = \begin{cases} 1 & \text{if } P(Y = 1 \mid X = x) > P(Y = 0 \mid X = x) \\ 0 & \text{if } P(Y = 1 \mid X = x) < P(Y = 0 \mid X = x) \end{cases}$$

The classification boundary of the Bayes rule is

$$\begin{aligned} & \{x : P(Y = 1 \mid X = x) = P(Y = 0 \mid X = x)\} \\ &= \{x : P(Y = 1 \mid X = x) - 0.5 = 0\} \end{aligned}$$

From Bayes' theorem

$$\begin{aligned} p(Y = 1 \mid X = x) &= \frac{p(x \mid Y = 1)p(Y = 1)}{p(x \mid Y = 1)p(Y = 1) + p(x \mid Y = 0)p(Y = 0)} \\ &= \frac{\pi_1 p_1(x)}{\pi_1 p_1(x) + (1 - \pi_1) p_0(x)} \end{aligned}$$

- ▶ $\pi_1 = p(Y = 1), \pi_0 = p(Y = 0)$: the marginal distribution of Y (prior class probabilities)
- ▶ $p_j(x) = p(x \mid Y = j)$: the conditional density of X given that $Y = j$.

Thus the Bayes rule becomes

$$h^*(x) = \begin{cases} 1 & \text{if } \frac{p_1(x)}{p_0(x)} > \frac{\pi_0}{\pi_1} \\ 0 & \text{otherwise.} \end{cases}$$

Unequal Losses

For any decision function, there are two possible errors:

- ▶ misclassifying a sample in class 0 to 1 (false positive)
- ▶ misclassifying a sample in class 1 to 0 (false negative)

Each type of error is associated with a cost (the price to pay for the consequence):

- ▶ $L(1, 0)$ is the cost of misclassifying a sample in class 1 to 0
- ▶ $L(0, 1)$ is the cost of misclassifying a sample in class 0 to 1.

We assume $L(j, j) = 0$ for $j = 0, 1$; but it may not be $L(0, 1) = L(1, 0)$.

The loss becomes

$$L(Y, h(X)) = L(1, 0)1(Y = 1, h(X) = 0) + L(0, 1)1(Y = 0, h(X) = 1)$$

For fixed x , the Bayes rule is given as

$$h^*(x) = \begin{cases} 1 & \text{if } L(1,0)P(Y = 1 \mid X = x) > L(0,1)P(Y = 0 \mid X = x) \\ 0 & \text{if } L(1,0)P(Y = 1 \mid X = x) < L(0,1)P(Y = 0 \mid X = x) \end{cases}$$

Equivalently,

$$h^*(x) = \begin{cases} 1 & \text{if } \frac{P(Y=1|X=x)}{P(Y=0|X=x)} > \frac{L(0,1)}{L(1,0)} \\ 0 & \text{if } \frac{P(Y=1|X=x)}{P(Y=0|X=x)} < \frac{L(0,1)}{L(1,0)} \end{cases}$$

the Bayes rule

$$h^*(x) = 1 \left\{ x : P(Y = 1 \mid X = x) > \frac{L(0,1)}{L(0,1) + L(1,0)} \right\}.$$

Using the Bayes' theorem,

- ▶ $\pi_1 = p(Y = 1), \pi_0 = p(Y = 0)$: the marginal distribution of Y (prior class probabilities)
- ▶ $p_j(x) = p(x \mid Y = j)$: the conditional density of X given that $Y = j$.

$$h^*(x) = \begin{cases} 1 & \text{if } \frac{p_1(x)}{p_0(x)} > \frac{\pi_0 L(0,1)}{\pi_1 L(1,0)} \\ 0 & \text{if } \frac{p_1(x)}{p_0(x)} < \frac{\pi_0 L(0,1)}{\pi_1 L(1,0)} \end{cases}$$

By changing the weights for L_{01} and L_{10} , we can effectively change the classification threshold.

- ▶ L_{01} = the loss of predicting a “non-disease” example to “disease”
- ▶ L_{10} = the loss of predicting a “disease” example to “non-disease”

$$h^*(x) = 1 \left\{ x : P(Y = 1 \mid X = x) > \frac{L(0, 1)}{L(0, 1) + L(1, 0)} \right\}.$$

How to increase the sensitivity and decrease the specificity of the rule?

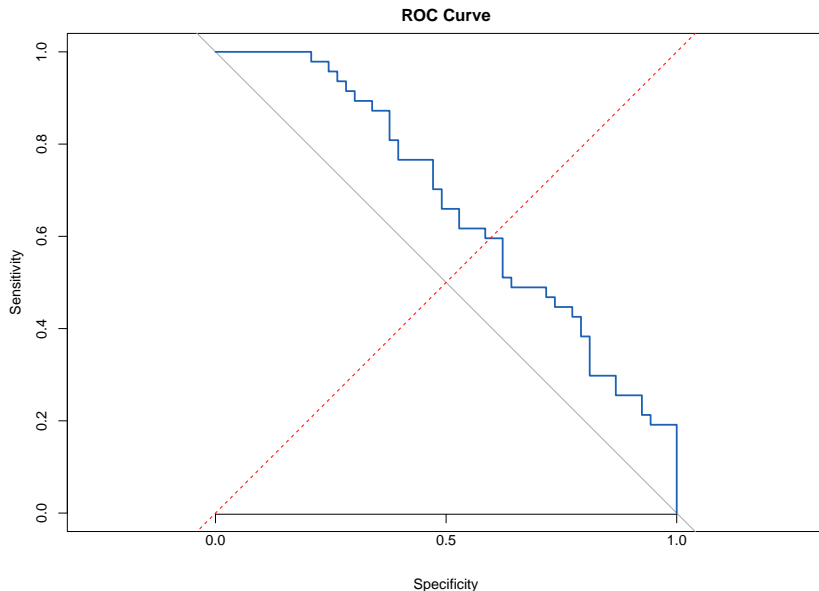
- ▶ Increase L_{10} and decrease L_{01} , say $L_{10} \gg L_{01}$.

How to increase the specificity and decrease the sensitivity of the rule?

- ▶ Increase L_{01} and decrease L_{10} , say $L_{01} \gg L_{10}$.

ROC curve

The ROC curve is a plot of the sensitivity versus specificity:



Multi-class Classification

Multi-class Classification

- ▶ Class label $Y \in \{1, \dots, K\}$, $K \geq 3$.
- ▶ The classifier $h : \mathbb{R}^d \longrightarrow \{1, \dots, K\}$.

The loss function $L(Y, h(X)) = \sum_{k=1}^L \sum_{l=1}^K C(l, k) I(Y = l, h(X) = k)$ where $C(l, k)$ = cost of classifying a sample in class l to class k .

The classification risk, or error rate, of h is defined as

$$R(h) = E_{X,Y}(L(Y, h(X)))$$

Using the 0-1 loss, $C(k, k) = 0$ for any $k = 1, \dots, K$, but equal to 1 otherwise, the rule h that minimizes $R(h)$ is

$$h^*(x) = \arg \max_{k=1, \dots, K} P(Y = k \mid x)$$

i.e., assign x to the most probable class using $P(Y \mid x)$.

We generally need to estimate multiple **discriminant functions**

$\delta_k(x), k = 1, \dots, K$

- ▶ Each $\delta_k(x)$ is associated with class k .
- ▶ $\delta_k(x)$ represents the evidence strength of a sample (x, y) belonging to class k .

The decision rule constructed using δ_k 's is

$$\hat{h}(x) = k^*, \quad \text{where} \quad k^* = \arg \max_{k=1, \dots, K} \delta_k(x)$$

The decision boundary of the classification rule \hat{h} between class k and class l is defined as

$$\{x : \delta_k(x) = \delta_l(x)\}$$

Note: $\delta_k(x)$ is related but need not be exact $P(Y = k | x)$.

LDA and QDA

Consider the binary classification

If $X \mid Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X \mid Y = 1 \sim N(\mu_1, \Sigma_1)$,

Bayes rule is

$$h^*(x) = \begin{cases} 1 & \text{if } r_1^2 < r_0^2 + 2 \log \left(\frac{\pi_1}{1-\pi_1} \right) + \log \left(\frac{|\Sigma_0|}{|\Sigma_1|} \right) \\ 0 & \text{otherwise} \end{cases}$$

where $r_i = \sqrt{(x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i)}$ for $i = 0, 1$ is the **Mahalanobis distance** between x and μ_i .

$$\log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = 0 \mid X = x)} = \log \frac{\pi_1 \phi(x; \boldsymbol{\mu}_1, \Sigma_1)}{\pi_0 \phi(x; \boldsymbol{\mu}_0, \Sigma_0)} = \delta_1(x) - \delta_0(x).$$

The Bayes rule is

$$h^*(x) = \operatorname{argmax}_{k \in \{0,1\}} \delta_k(x)$$

where

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) + \log \pi_k$$

This is called the **Gaussian discriminant function**

- ▶ The decision boundary: $\{x \in \mathcal{X} : \delta_1(x) = \delta_0(x)\}$
- ▶ **quadratic discriminant analysis (QDA)**

To estimate $\pi_0, \pi_1, \mu_0, \mu_1, \Sigma_0, \Sigma_1$:

$$\hat{\pi}_0 = \frac{1}{n} \sum_{i=1}^n (1 - Y_i), \quad \hat{\pi}_1 = \frac{1}{n} \sum_{i=1}^n Y_i$$

$$\hat{\mu}_0 = \frac{1}{n_0} \sum_{i:Y_i=0} X_i, \quad \hat{\mu}_1 = \frac{1}{n_1} \sum_{i:Y_i=1} X_i$$

$$\hat{\Sigma}_0 = \frac{1}{n_0 - 1} \sum_{i:Y_i=0} (X_i - \hat{\mu}_0) (X_i - \hat{\mu}_0)^\top$$

$$\hat{\Sigma}_1 = \frac{1}{n_1 - 1} \sum_{i:Y_i=1} (X_i - \hat{\mu}_1) (X_i - \hat{\mu}_1)^\top$$

LDA

LDA assumes both classes are from Gaussian and they have the same covariance matrix

$$\Sigma_k = \Sigma, \quad k = 0, 1$$

Note that

$$\log \Pr(Y = k \mid X = x) = -\frac{1}{2} (x - \boldsymbol{\mu}_k)^\top \Sigma^{-1} (x - \boldsymbol{\mu}_k) + \log \pi_k + \text{const.}$$

If prior probabilities are same, the LDA classifies x to the class with centroid closest to x , using the squared Mahalanobis distance, based on the common covariance matrix.

Alternatively,

$$h^*(x) = \begin{cases} 1 & \text{if } \delta_1(x) > \delta_0(x) \\ 0 & \text{otherwise} \end{cases}$$

where the discriminant function is simplified

$$\delta_k(x) = x^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k.$$

Pooled estimate of the Σ :

$$\hat{\Sigma} = \frac{(n_0 - 1) \hat{\Sigma}_0 + (n_1 - 1) \hat{\Sigma}_1}{n_0 + n_1 - 2}$$

Multi-class classification

QDA assume that $X | Y = k \sim N(\mu_k, \Sigma_k)$.

$$h^*(x) = \operatorname{argmax}_k \delta_k(x)$$

where

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) + \log \pi_k.$$

If all Gaussians assumed to have equal variance Σ ,

$$\delta_k(x) = x^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k.$$

The corresponding estimates are given by

$$\begin{aligned} \hat{\pi}_k &= \frac{1}{n} \sum_{i=1}^n 1(y_i = k), \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{i: Y_i = k} X_i \\ \hat{\Sigma}_k &= \frac{1}{n_k - 1} \sum_{i: Y_i = k} (X_i - \hat{\mu}_k)(X_i - \hat{\mu}_k)^\top \\ \hat{\Sigma} &= \frac{\sum_{k=0}^{K-1} (n_k - 1) \hat{\Sigma}_k}{n - K}. \end{aligned}$$

Logistic Regression: binary case

Binary case

The logistic regression assumes that

$$p_1(x; \beta_0, \beta_1) := P(Y = 1 \mid X = x) = \frac{\exp(\beta_0 + x^\top \beta_1)}{1 + \exp(\beta_0 + x^\top \beta_1)}$$

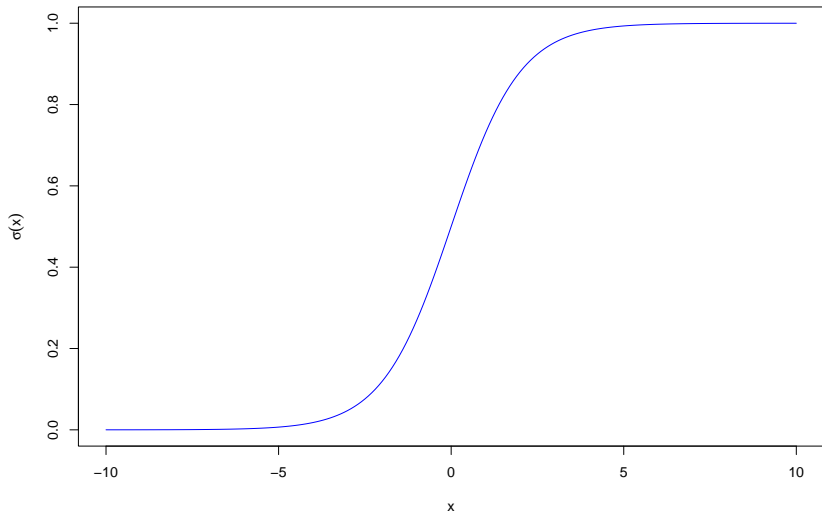
The model can be written as

$$\text{logit}(x) := \text{logit}(\Pr(Y = 1 \mid X = x)) = \log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = 0 \mid X = x)} = \beta_0 + \beta_1^\top x$$

- ▶ **logit function:** $\text{logit}(a) = \log(a/(1 - a)) : (0, 1) \mapsto \mathbb{R}$
- ▶ The inverse of logit function (**logistic function** or **sigmoid function**):

$$\sigma(a) = \exp(a)/(1 + \exp(a)) : \mathbb{R} \mapsto (0, 1)$$

Sigmoid Function



MLE for logistic models

Notations: assuming x_i contains the constant term 1 (thus a $p + 1$ vector).

$$\boldsymbol{\beta} := \{\beta_0, \boldsymbol{\beta}_1^\top\}^\top$$

$$\mathbf{y} := [y_1, \dots, y_n]^\top$$

$$\mathbf{p} := \mathbf{p}(\boldsymbol{\beta}) = [p(x_1; \boldsymbol{\beta}), \dots, p(x_n; \boldsymbol{\beta})]^\top$$

$$\mathbf{W} := \mathbf{W}(\boldsymbol{\beta}) = \text{diag}\{p(x_i; \boldsymbol{\beta})(1 - p(x_i; \boldsymbol{\beta}))\} : n \times n$$

The log (conditional) likelihood function is

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^n \{y_i \log p(x_i; \boldsymbol{\beta}) + (1 - y_i) \log [1 - p(x_i; \boldsymbol{\beta})]\}$$

The loss function, as negative loglikelihood function, is called **binomial deviance (loss)**:

$$L(Y, p_Y(X)) = -\{1(Y = 0) \log(\Pr(Y = 0 \mid X; \boldsymbol{\theta})) + 1(Y = 1) \log(\Pr(Y = 1 \mid X; \boldsymbol{\theta}))\}$$

$$\begin{aligned}
-\ell(\boldsymbol{\beta}) &= -\sum_{i=1}^n \{y_i \log p(x_i; \boldsymbol{\beta}) + (1 - y_i) \log [1 - p(x_i, \boldsymbol{\beta})]\} \\
&= -\sum_{i=1}^n \left\{ y_i \boldsymbol{\beta}^\top x_i - \log \left[1 + \exp \left(\boldsymbol{\beta}^\top x_i \right) \right] \right\}.
\end{aligned}$$

The score and Hessian are given by

$$\begin{aligned}
\frac{\partial(-\ell(\boldsymbol{\beta}))}{\partial \boldsymbol{\beta}} &= -\sum_{i=1}^n x_i [y_i - p(x_i; \boldsymbol{\beta})] = -\mathbf{X}^\top (\mathbf{y} - \mathbf{p}) \\
\frac{\partial^2(-\ell(\boldsymbol{\beta}))}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} &= \sum_{i=1}^n x_i x_i^\top p(x_i; \boldsymbol{\beta}) [1 - p(x_i; \boldsymbol{\beta})] = \mathbf{X}^\top \mathbf{W} \mathbf{X}
\end{aligned}$$

Gradient descent step: In the k-th sep,

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha \times \mathbf{X}^\top (\mathbf{y} - \mathbf{p}^{(k)})$$

Newton-Raphson step: In the k-th sep,

$$\begin{aligned}\boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(k)} + \left(\mathbf{X}^\top \mathbf{W}^{(k)} \mathbf{X} \right)^{-1} \mathbf{X}^\top (\mathbf{y} - \mathbf{p}^{(k)}) \\ &= \left(\mathbf{X}^\top \mathbf{W}^{(k)} \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{W}^{(k)} \left(\mathbf{X} \boldsymbol{\beta}^{(k)} + \mathbf{W}^{(k)-1} (\mathbf{y} - \mathbf{p}^{(k)}) \right) \\ &= \left(\mathbf{X}^\top \mathbf{W}^{(k)} \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{W}^{(k)} \mathbf{z}^{(k)}\end{aligned}$$

where we defined the adjusted response

$$\mathbf{z}^{(k)} = \mathbf{X} \boldsymbol{\beta}^{(k)} + \mathbf{W}^{(k)-1} (\mathbf{y} - \mathbf{p}^{(k)})$$

The Newton-Raphson's approach is equivalent to **Iteratively Reweighted Least Squares Algorithm**:

$$\boldsymbol{\beta}^{(k+1)} = \arg \min_{\boldsymbol{\beta}} (\mathbf{z}^{(k)} - \mathbf{X} \boldsymbol{\beta})^\top \mathbf{W}^{(k)} (\mathbf{z}^{(k)} - \mathbf{X} \boldsymbol{\beta}).$$

Interpretation of β_j

$$e^{\beta_j} = \frac{\text{odds}(\dots, X_j = x + 1, \dots)}{\text{odds}(\dots, X_j = x, \dots)} = \text{odds ratio}$$

When an increase of X_j by one unit from x to $x + 1$, while keeping all other predictors fixed, it multiplies the odds by e^{β_j} (relative change from the odds when $X_j = x$);

e.g.

- ▶ If $X_j = 0$ or 1, then for the group with $X_j = 1$, the odds of the event are e^{β_j} times that of the group with $X_j = 0$, with other values of X_{-j} fixed.
 - ▶ When $\beta_j > 0$, the group with $X_j = 1$ has $100(e^{\beta_j} - 1)\%$ more odds than the group with $X_j = 0$, with other values of X_{-j} fixed.
 - ▶ When $\beta_j < 0$, then it is a decrease in the odds by $100(1 - e^{\beta_j})\%$.

For intercept, e^{β_0} is the probability of the event for the base group when all X 's = 0.

Inferences: logistic regression

Assuming correct model specification, by central limit theorem, the MLE estimator

$$\hat{\beta} \rightarrow N\left(\beta^*, (\mathbf{X}'\mathbf{W}(\beta^*)\mathbf{X})^{-1}\right).$$

Here, β^* is the truth. The estimator for the variance of $\hat{\beta}$ is given by

$$\text{var}(\hat{\beta}) = \left(\mathbf{X}'\mathbf{W}(\hat{\beta})\mathbf{X}\right)^{-1}.$$

So as $n \rightarrow \infty$,

$$\frac{\hat{\beta}_j - \beta_j}{se(\hat{\beta}_j)} \sim N(0, 1),$$

where

$$se(\hat{\beta}_j) = \left[\left(\mathbf{X}'\mathbf{W}(\hat{\beta})\mathbf{X} \right)^{-1} \right]_{j,j}^{1/2}$$

Alternative formulation: logistic regression

Suppose that

$$Y_i = 1\{\beta_0 + X_i^\top \beta_1 + \varepsilon_i\}$$

where ε_i is independent of X_i , following a logistic distribution (mean 0 and standard deviation $\pi/\sqrt{3}$), aka, the c.d.f. of ε_i given by

$$F(\epsilon) = \frac{\exp(\epsilon)}{1 + \exp(\epsilon)}.$$

Then

$$P(Y_i|X_i = x) = \frac{\exp(\beta_0 + x^\top \beta_1)}{1 + \exp(\beta_0 + x^\top \beta_1)}.$$

If important predictors are omitted from the model (model misspecified), the usual interpretation linking the coefficients to (true) log-odds will break down.

Logistic Regression: multi-class case, sparsity

Logistic Regression: multi-class case, sparsity

Suppose there are K groups. Let the K -th group be the base group. One may model $Pr(Y = k|x; \beta_0, \beta)$ as

$$Pr(Y = k|x; \beta_0, \beta) = \frac{\exp(x^\top \beta_k + \beta_{k0})}{\sum_{k'=1}^K \exp(x^\top \beta_{k'} + \beta_{k'0})}$$

- ▶ $\beta_0 := (\beta_{10}, \dots, \beta_{K0})$ the vector of K intercepts.
- ▶ $\beta := (\beta_1, \dots, \beta_K)^\top$, a K by p matrix.
- ▶ $\eta := \beta x + \beta_0$ the vector of K logits

The \mathbf{S} is the **softmax function** $\mathbb{R}^K \mapsto (0, 1)^K$, defined as

$$\mathbf{S}(\eta)_k = \frac{e^{\eta_k}}{\sum_{k'}^K e^{\eta_{k'}}}, \quad k = 1, \dots, K; \quad \eta = (\eta_1, \dots, \eta_K)^\top$$

$$Pr(Y = k|x; \beta_0, \beta) := Pr(Y = k|\eta) = \mathbf{S}(\eta)_k$$

The multi-class logistic regression (or multinomial logistic regression) models $K - 1$ logits:

- ▶ set $\beta_{K0} = 0, \beta_K = 0$ to avoid overparametrization.
- ▶ treat K as the base group

$$\log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = K \mid X = x)} = \beta_{10} + \beta_1^\top x$$

$$\log \frac{\Pr(Y = 2 \mid X = x)}{\Pr(Y = K \mid X = x)} = \beta_{20} + \beta_2^\top x$$

$$\log \frac{\Pr(Y = K - 1 \mid X = x)}{\Pr(Y = K \mid X = x)} = \beta_{(K-1)0} + \beta_{K-1}^\top x$$

Equivalently,

$$p_k(x) \equiv \Pr(Y = k \mid x) = \frac{\exp(\beta_{k0} + \beta_k^\top x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^\top x)} \quad \text{for } k = 1, \dots, K-1$$

$$p_K(x) \equiv \Pr(Y = K \mid x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^\top x)}$$

Clearly $\sum_{k=1}^K p_k(x) = 1$. The parameter vector

$$\boldsymbol{\theta} = \left\{ \beta_{10}, \beta_1^\top, \dots, \beta_{(K-1)0}, \beta_{K-1}^\top \right\}^\top$$

Let $p_{k,i} := \Pr(Y_i = k | X = x_i, \boldsymbol{\theta})$ and $p_{y_i}(x_i; \boldsymbol{\theta}) = \Pr(Y_i = y_i | X = x_i, \boldsymbol{\theta})$.

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \sum_{i=1}^n \log p_{y_i}(x_i; \boldsymbol{\theta}) = \log \left(\prod_{i=1}^n \prod_{k=1}^K p_{k,i}^{1(y_i=k)} \right) \\ &= \sum_{i=1}^n \sum_{k=1}^K 1(y_i = k) \log(p_{k,i})\end{aligned}$$

Since $\beta_{K0} := 0$ and $\beta_K := 0$, the log-likelihood:

$$\sum_{i=1}^n \left\{ \beta_{y_i 0} + \beta_{y_i}^\top x_i - \log \left[1 + \sum_{k=1}^{K-1} \exp(\beta_{k0} + \beta_k^\top x_i) \right] \right\}$$

The loss function, as negative loglikelihood function, is called **multinomial deviance** (loss):

$$L(Y, p_Y(X)) = -\log p_Y(X; \theta) = -\sum_{k=1}^K 1(Y = k) \log(\Pr(Y = k | X; \theta)).$$

Compare logistic regression with LDA

Logistic regression:

- ▶ Maximizing the conditional likelihood, the multinomial likelihood with probabilities $\Pr(Y = k \mid \mathbf{X})$
- ▶ The marginal density $\Pr(X)$ is ignored (fully nonparametric)
 - ▶ **discriminative approach**: only modelling $\Pr(Y = k \mid x)$

LDA:

- ▶ Maximizing the full log-likelihood based on the joint density

$$\Pr(X, Y = k) = \phi(X; \boldsymbol{\mu}_k, \Sigma) \pi_k$$

- ▶ Marginal density does play a role $\Pr(\mathbf{X}) = \sum_k \pi_k \phi(X; \boldsymbol{\mu}_k, \Sigma)$
 - ▶ **generative approach**: joint modelling of $\Pr(X, Y = k)$; also modelling $\Pr(x \mid Y = k)$

Regularized logistic regression

The idea is to minimize the penalized negative likelihood function (binary response):

$$\min_{\beta_0, \beta_1} \sum_{i=1}^n \left(-y_i (\beta_0 + \mathbf{x}_i^\top \beta_1) + \log \left(1 + e^{\beta_0 + \mathbf{x}_i^\top \beta_1} \right) \right) + \lambda J(\beta_1)$$

The update is equivalent to solving the weighted LS till convergence (Iteratively Reweighted Least Squares Algorithm):

$$(\beta_0^{(k+1)}, \beta_1^{(k+1)}) = \arg \min_{\beta} \left\{ (\mathbf{z}^{(k)} - \mathbf{X}\beta)^\top \mathbf{W}^{(k)} (\mathbf{z}^{(k)} - \mathbf{X}\beta) + \lambda J(\beta_1) \right\}.$$

For the Lasso penalty, it can be solved using coordinate descent.

Sparse logistic regression

Algorithm (Coordinate descent for sparse logistic regression):

Let $\hat{\beta}^{(0)} = (\hat{\beta}_0^{(0)}, \hat{\beta}_1^{(0)}, \dots, \hat{\beta}_p^{(0)})^T$

For $k = 0, 1, 2, \dots$,

- ▶ Compute $p_1(x_i; \hat{\beta}^{(k)}), z_i^{(k)}, w_{ii}^{(k)}, i = 1, \dots, n$.
- ▶ Let $\beta_0 = \sum_i [w_{ii}^{(k)}(z_i^{(k)} - \sum_{l=1}^p \hat{\beta}_l^{(k)} x_{il})] / \sum_i w_{ii}^{(k)}, \quad \beta_l = \hat{\beta}_l^{(k)}, l = 1, \dots, p$.
 - ▶ for $j = 1, \dots, p$ do
 - ▶ compute $r_{ij} = z_i^{(k)} - \beta_0 - \sum_{l \neq j} \beta_l x_{il}, i = 1, \dots, n$
 - ▶ compute $u_j^{(k)} = \sum_{i=1}^n w_{ii}^{(k)} r_{ij} x_{ij}$,
 - ▶ compute $v_j^{(k)} = \sum_{i=1}^n w_{ii}^{(k)} x_{ij}^2$,
 - ▶ compute $\beta_j = \text{soft}(u_j^{(k)} / v_j^{(k)}, \lambda / v_j^{(k)})$
 - ▶ $\hat{\beta}_0^{(k+1)} = \beta_0, \hat{\beta}_j^{(k+1)} = \beta_j, j = 1, \dots, p$

until convergence

Others: nonparametric classifier, robust loss,
viewpoint from perceptrons

KNN

For any given $X = x_0$, we find the K closest neighbors to $X = x_0$ in the training data, and examine their corresponding Y :

$$P(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_K(x_0)} 1(y_i = j)$$

Estimate the conditional probability for group j by the proportion out of the k neighbors that are in group j .

The smaller that K is the more flexible the method will be.

Note: more on nonparametric method (e.g., nonparametric logistic regression) in future lessons.

Others: Alternative loss functions

For binary classification,

$$\min_{f \in \mathcal{F}_{\pm 1}} \mathbb{E}_{X,Y}(1(Y \neq f(X)))$$

where $\mathcal{F}_{\pm 1}$ consists of function that maps to $\{-1, 1\}$.

The ERM solution is

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}_{\pm 1}} \frac{1}{n} \sum_{i=1}^n 1(y_i \neq f(x_i))$$

- ▶ the choice of $\mathcal{F}_{\pm 1}$: **hard-classifier** $f \in \mathcal{F}_{\pm 1}$
 - ▶ perceptron: $\mathcal{F}_{\pm 1} = \{\text{sign}(\beta_0 + \beta^\top x) : \beta_0, \beta\}$

Equivalent formulation, but using $y_i \in \{-1, 1\}$,

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}_{\pm 1}} \frac{1}{n} \sum_{i=1}^n 1(-y_i f(x_i) > 0)$$

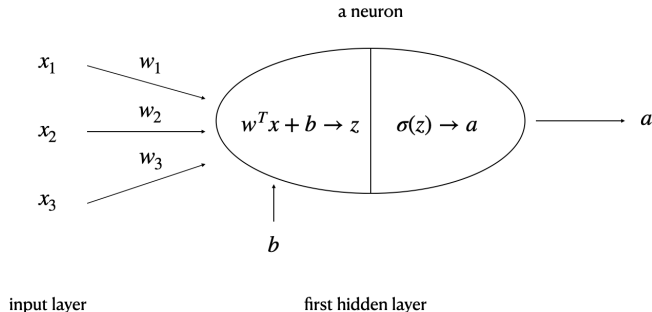
Using some smooth and convex surrogate function $\psi(z)$ for $1(z > 0)$ and relaxing the class of functions \mathcal{F} , solve

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) = \frac{1}{n} \sum_{i=1}^n \psi(-y_i f(x_i))$$

- ▶ Choice of $L(y, f)$:
 - ▶ Squared error: $(y - f)^2 = (1 - yf)^2$
 - ▶ Binomial deviance (logistic): $\log(1 + \exp(-2yf))$
 - ▶ Exponential loss: $\exp(-yf)$
 - ▶ SVM loss: $(1 - yf)_+$
- ▶ Choice of \mathcal{F} : **soft-classifier** $f \in [-1, 1]$ or \mathbb{R}
 - ▶ decision: $\hat{Y}_i = \text{sign}(\hat{f}_n(X_i))$

More in future lessons...

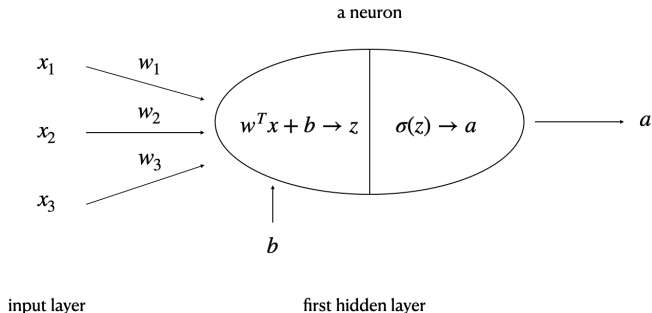
Rosenblatt's Perceptron



Here, $\sigma(\cdot)$ is an activation function: three simplest cases

- i. identity $\sigma(v) = v$
- ii. indicator (Heaviside step) $\sigma(v) = 1(v > 0)$
- iii. sigmoid $\sigma(v) = 1/(1 + e^{-v})$

The output \hat{a} will be measured against the actual outcome value with suitable choice of loss function.



Formally,

$$\sigma \left(\sum_{j=1}^p x_j w_j + b' \right) = \sigma \left(\mathbf{x}^\top \mathbf{w} + b \right), \quad \boldsymbol{\beta} := (\mathbf{w}, b)$$

Note: OLS in regression is a special case when $\sigma(\cdot)$ is identity function and squared error loss (see Adaline).

The Perceptron Learning Algorithm.

let β include the b , \mathbf{x} includes the intercept, $\alpha > 0$ small constant (learning rate). Start with random weight, then iteratively update the weight

- ▶ For $k = 1, 2, \dots, K$:
 - ▶ for each $i = 1, \dots, n$:
 - ▶ compute $\hat{y}_i^{(k)} = 1 \left(\mathbf{x}_i^\top \beta^{(k)} > 0 \right)$ (Heaviside)
 - ▶ compute $e_i^{(k)} = y_i - \hat{y}_i^{(k)}$
 - ▶ update $\beta^{(k+1)} = \beta^{(k)} + \alpha \times e_i^{(k)} \times \mathbf{x}_i$

A loop over the whole training data for each k is called a **training epoch**; the whole process repeats the loops for K epochs.

A training algorithm that updates the parameter after seeing one example is called **on-line training** (stochastic gradient descent).

Compared with logistic regression

Recall logistic regression, the update at the k -th iteration

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha \times \mathbf{X}^\top (\mathbf{y} - \mathbf{p}^{(k)})$$

The **on-line training** process is

- ▶ For $k = 1, 2, \dots, K$:
 - ▶ for each $i = 1, \dots, n$:
 - ▶ compute $\hat{y}_i^{(k)} = p(x_i; \boldsymbol{\beta}^{(k)})$ (sigmoid)
 - ▶ compute $e_i^{(k)} = y_i - \hat{y}_i^{(k)}$
 - ▶ $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha \times e_i^{(k)} \times \mathbf{x}_i$

Note

- ▶ Perceptron uses hard-classifier, while logistic uses soft-classifier
- ▶ Perceptron algorithm does not converge if the data is not linearly separable
- ▶ SGD (WNLS) for logistic regression converges to global minimum of the binary entropy loss function (even when the data is not linearly separable)

Adaline (ADaptive LInear NEuron)/OLS

Adaline for classification is a special case when $\sigma(\cdot)$ is identity function and the loss is squared error loss.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{a}_i - y_i)^2, \quad \hat{a}_i = \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}$$

It can be implemented using gradient descent for OLS.

$$\blacktriangleright \boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha \times 2(y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}^{(k)}) \times \mathbf{x}_i$$

As a byproduct, the Adaline classifies based on the hard rule

$$\hat{y}_i = 1 \left(\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} > 0.5 \right)$$

A major difference from Rosenblatt's Perceptron is that

- ▶ \hat{a}_i is the continuous output (as opposed to \hat{y}_i) that is measured against the true y_i in the loss calculation
 - ▶ That is, the hard-classification step is not backpropagated.