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, 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

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

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

Accuracy (1-Error): $(\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN})$.

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

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

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

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

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

example 1

	Predicted	
True	email	spam
email	573	40
spam	53	334

spam =presence of disease, email=absence of disease

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

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 $X \in \mathcal{X} \subset \mathbb{R}^p$
- 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

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

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

$$p(Y=1 \mid X=x) = \frac{\pi_1 p_1(x)}{\pi_1 p_1(x) + (1-\pi_1)\, p_0(x)}$$

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

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

This decision-making process balances two types of information:

- Likelihood ratio (evidence) $\frac{p_1(x)}{p_0(x)}$: compares how probable it is that the observed data x comes from class 1 as opposed to class 0.
- Prior ratio $\frac{\pi_0}{\pi_1}$: our initial belief about the relative frequency of class 0 to class 1 before seeing any data (say x).

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,

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

the Bayes rule

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

In light of 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 $Y = j$.

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

By changing the weights for $L(0,1)$ and $L(1,0)$, we can effectively change the classification threshold.

- $L(0,1)$ = the cost of predicting a "non-disease" example to "disease"
- $L(1,0)$ = the cost 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?

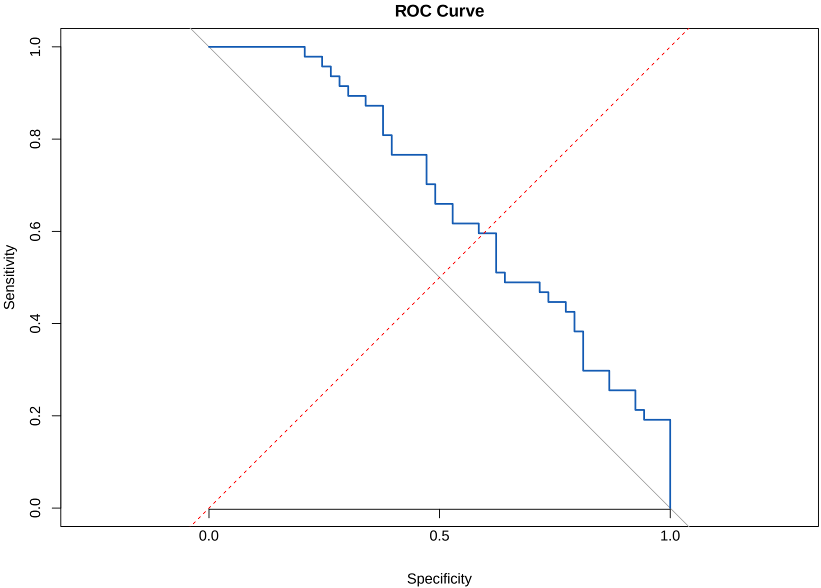
- Increase $L(1,0)$ and decrease $L(0,1)$.

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

- Increase $L(0,1)$ and decrease $L(1,0)$.

Receiver Operating Characteristic (ROC) curve

A ROC curve is a plot of sensitivity v.s. specificity:



- An ideal ROC curve will hug the top right corner.
- An alternative ROC curve will be a curve plotting the sensitivity (true positive rate or 1-Type II error) against the false positive rate (Type I error).

- The **area under curve (AUC)** is a commonly used quantitative measure of overal predictive performance.
  - A value of 0.5 means the predictions were no better than random guessing.

precision-recall relation

$$\begin{aligned}\text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{\text{TP}}{\text{P}^*} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{\text{P}}\end{aligned}$$

In general, as the threshold varies, a low **recall** is associated with a high **precision**, and vice versa (the pattern however, is not exact).

F_1 score is the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- F_1 is from 0 to 1.
- a score of 0 means very poor performance (either precision or recall or both are zero)
- a score of 1 indicates perfect performance (both precision and recall are at their maximum).

Bias-variance decomposition for binary response (0-1 loss)

The bias-variance tradeoff behaves differently for $0-1$/classification loss than it does for squared error loss.

But the prediction error (classification error/$0-1$ loss) is no longer the sum of squared bias and variance.

What matters is that $\mathrm{E}\hat{m}(x_0)$ and $m(x_0)$ is on the same side of $1/2$ (thus correct classification).

For $(X, Y) \in \mathbb{R}^p \times \{0, 1\}$, consider the regression function,

$$m(x) = \mathrm{E}(Y \mid X = x) = P(Y = 1 \mid X = x)$$

The Bayes classifier is given by

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

The plug-in classifier is given by

$$\hat{h}(x) = \begin{cases} 0 & \text{if } \hat{m}(x) \leq 1/2 \\ 1 & \text{if } \hat{m}(x) > 1/2 \end{cases}$$

$$\begin{aligned} \operatorname{Err}(x_0) &= \mathrm{P}\left(Y_0 \neq \hat{h}(X_0) \mid X_0 = x_0\right) \\ &= \operatorname{Err}_{\mathrm{B}}(x_0) + |2m(x_0) - 1| \, \mathrm{P}\left(\hat{h}(X_0) \neq h^*(X_0) \mid X_0 = x_0\right) \end{aligned}$$

where $\operatorname{Err}_{\mathrm{B}}(x_0) = \mathrm{P}(Y_0 \neq h^*(X_0) \mid X_0 = x_0)$, the irreducible Bayes error at x_0.

Using the approximation $\hat{m}(x_0) \sim N(\mathrm{E}\hat{m}(x_0), \mathrm{Var}(\hat{m}(x_0)))$, it can be shown that

$$\Pr\left(\hat{h}(X^*) \neq h^*(X^*) \mid X^* = x_0\right) \approx \Phi\left(\frac{\mathrm{sign}\left(\frac{1}{2} - m(x_0)\right)\left(\mathrm{E}\hat{m}(x_0) - \frac{1}{2}\right)}{\sqrt{\mathrm{Var}(\hat{m}(x_0))}}\right)$$

The term $\mathrm{sign}\left(\frac{1}{2} - m(x_0)\right)\left(\mathrm{E}\hat{m}(x_0) - \frac{1}{2}\right)$ is a kind of **boundary-bias term**, as it depends on the true $m(x_0)$ only through which side of the boundary 1/2 that it lies. The bias and variance

combine in a multiplicative rather than additive fashion.

Multi-class Classification

Multi-class Classification

Previous metrics for binary classification can be applied for K-classes, for example, by calculating precision, recall, and F_1 score independently for each class, treating each class as a binary classification (the class vs. all others), and then averaging them.

Macro averaging

$$\text{Precision}_{\text{macro}} = \frac{1}{K}\sum_{i=1}^{K}\text{Precision}_k$$

$$\text{Recall}_{\text{macro}} = \frac{1}{K}\sum_{i=1}^{K}\text{Recall}_k$$

$$\text{F}_{1,\text{macro}} = \frac{1}{K}\sum_{i=1}^{K}\text{F}_{1,k}$$

Macro averaging treats all classes equally.

Alternative: Sum up the individual true positives, false positives, and false negatives across all classes and then calculate the metrics.

Micro averaging

$$\text{Precision}_{\text{micro}} = \frac{\sum_{i=1}^{K} \text{TP}_k}{\sum_{i=1}^{K} \text{TP}_k + \sum_{i=1}^{K} \text{FP}_k}$$

$$\text{Recall}_{\text{micro}} = \frac{\sum_{i=1}^{K} \text{TP}_k}{\sum_{i=1}^{K} \text{TP}_k + \sum_{i=1}^{K} \text{FN}_k}$$

$$\text{F}_{1,\text{micro}} = 2 \times \frac{\text{Precision}_{\text{micro}} \times \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}}$$

Micro averaging is more suitable when class imbalance is present.

Set up

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

The loss function $L(Y, h(X)) = \sum_{k=1}^{L} \sum_{l=1}^{K} C(l, k) 1(Y = l, h(X) = k)$ 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 $k = 1, \cdots, K$, but equal to 1 otherwise, the rule h that minimizes $R(h)$ is

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

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

We generally need to estimate multiple **discriminant functions** $\delta_k(x), k = 1, \cdots, K$

- 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

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

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 $P(Y = k \mid x)$.

LDA and QDA

Gassuain discriminant analysis: Binary classification

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

Using the Bayes' Theorem,

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

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

Note: LDA is a special case where $\Sigma_1 = \Sigma_0$.

Quadratic discriminant analysis (QDA)

$$\log \frac{\Pr(Y=1 \mid X=x)}{\Pr(Y=0 \mid X=x)} = \log \frac{\pi_1 \phi\left(x; \mu_1, \Sigma_1\right)}{\pi_0 \phi\left(x; \mu_0, \Sigma_0\right)} = \delta_1(x) - \delta_0(x).$$

The Bayes rule is

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

where

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

This is called the **Gaussian discriminant function**

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

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

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

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

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

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

Linear discriminant analysis (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 Gaussian discriminant function can be simplified

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

- The decision boundary: $\{x \in \mathcal{X} : \delta_1(x) = \delta_0(x)\}$
  - **linear discriminant analysis** (LDA): boundary is linear

Pooled estimate of the Σ:

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

Multi-class classification (trivial extension)

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

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

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

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

Logistic Regression: binary case

Binary case

The logistic regression assumes that

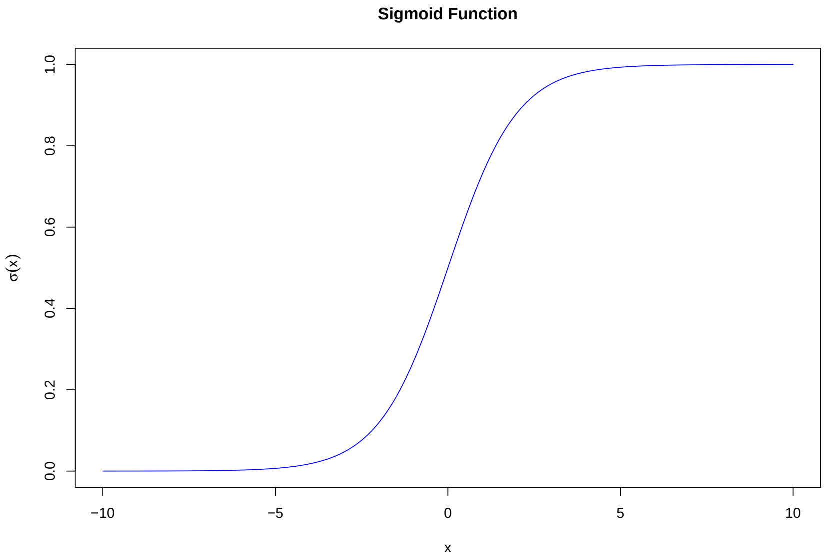
$$p_1(x; \beta_0, \boldsymbol{\beta}_1) := P(Y = 1 \mid X = x) = \frac{\exp\left(\beta_0 + x^\top \boldsymbol{\beta}_1\right)}{1 + \exp\left(\beta_0 + x^\top \boldsymbol{\beta}_1\right)}$$

The model can be written as

$$\operatorname{logit}(x) := \operatorname{logit}(\Pr(Y = 1 \mid X = x)) = \log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = 0 \mid X = x)} = \beta_0 + \boldsymbol{\beta}_1^\top x$$

- **logit function**: $\operatorname{logit}(a) = \log(a/(1-a)) : (0,1) \mapsto \mathbb{R}$
  - $\beta_0 + \boldsymbol{\beta}_1^\top x$: **logits**, **net input** or **pre-activation value**
- The inverse of logit function (**logistic function** or **sigmoid function**):

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


Sigmoid Function
σ(x)
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
x

MLE for logistic models

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

$$\begin{aligned}
\boldsymbol{\beta} &:= \{\beta_0, \boldsymbol{\beta}_1^\top\}^\top \\
\mathbf{y} &:= [y_1, \cdots, y_n]^\top \\
\mathbf{p} &:= \mathbf{p}(\boldsymbol{\beta}) = [p(x_1;\boldsymbol{\beta}), \cdots, p(x_n;\boldsymbol{\beta})]^\top \\
\mathbf{W} &:= \mathbf{W}(\boldsymbol{\beta}) = \operatorname{diag}\{p(x_i;\boldsymbol{\beta})(1-p(x_i;\boldsymbol{\beta}))\} : n \times n
\end{aligned}$$

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

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

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

The score and Hessian are given by

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

Gradient descent step: In the k-th sep,

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

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

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

where we defined the adjusted response

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

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

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

Interpretation of β_j

$$e^{\beta_j} = \underbrace{\frac{P\left(Y=1|\ldots,X_j=x+1,\ldots\right)/P\left(Y=0|\ldots,X_j=x+1,\ldots\right)}{P\left(Y=1|\ldots,X_j=x,\ldots\right)/P\left(Y=0|\ldots,X_j=x,\ldots\right)}}_{\text{odds ratio}}$$

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, $e^{\beta_0} \div \left(1 + e^{\beta_0}\right)$ 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

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

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

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

So as $n \to \infty$,

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

where

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

Alternative formulation: logistic regression

Suppose that

$$Y_i = 1\{\beta_0 + X_i^\top \boldsymbol{\beta}_1 + \varepsilon_i\}$$

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

$$P(Y_i = 1|X_i = x) = \frac{exp(\beta_0 + x^\top \boldsymbol{\beta}_1)}{1 + exp(\beta_0 + x^\top \boldsymbol{\beta}_1)}.$$

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 $Pr(Y = k|x; \boldsymbol{\beta}_0, \boldsymbol{\beta})$ as

$$Pr(Y = k|x; \boldsymbol{\beta}_0, \boldsymbol{\beta}) = \frac{\exp\left(x^\top \boldsymbol{\beta}_k + \beta_{k0}\right)}{\sum_{k'=1}^{K} \exp\left(x^\top \boldsymbol{\beta}_{k'} + \beta_{k'0}\right)}$$

- $\boldsymbol{\beta}_0 := (\beta_{10}, \ldots, \beta_{K0})$ the vector of K intercepts.
- $\boldsymbol{\beta} := (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K)^\top$, a K by p matrix.
- $\boldsymbol{\eta} := \boldsymbol{\beta} x + \boldsymbol{\beta}_0$ the vector of K logits

To ensure identification of the parameters:

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

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

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

$$\log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = K \mid X = x)} = \beta_{10} + \boldsymbol{\beta}_1^\top x$$

$$\log \frac{\Pr(Y = 2 \mid X = x)}{\Pr(Y = K \mid X = x)} = \beta_{20} + \boldsymbol{\beta}_2^\top x$$

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

interpretation of coefficients:

$$e^{\beta_{kj}} = \frac{P\left(Y = k \mid \ldots, X_j = x + 1, \ldots\right) / P\left(Y = K \mid \ldots, X_j = x + 1, \ldots\right)}{P\left(Y = k \mid \ldots, X_j = x, \ldots\right) / P\left(Y = K \mid \ldots, X_j = x, \ldots\right)}$$

Equivalently,

$$p_k(x) \equiv \Pr(Y = k \mid x) = \frac{\exp\left(\beta_{k0} + \beta_k^\top x\right)}{1 + \sum_{l=1}^{K-1} \exp\left(\beta_{l0} + \beta_I^\top x\right)}, \quad k = 1, \ldots, K-1$$

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

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

$$\boldsymbol{\theta} = \left\{\beta_{10}, \boldsymbol{\beta}_1^\top, \ldots, \beta_{(K-1)0}, \boldsymbol{\beta}_{K-1}^\top\right\}^\top$$

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 $\boldsymbol{\beta}_K := 0$, the log-likelihood:

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

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

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

Side note

If prediction is the only concern, we may fit a overparametrized model. This is the approach taken in the perceptron-based model.

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

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

$$Pr(Y = k | x; \boldsymbol{\beta}_0, \boldsymbol{\beta}) := Pr(Y = k | \boldsymbol{\eta}) = \mathbf{S}(\boldsymbol{\eta})_k$$

interpretation of coefficients:

$$e^{\beta_{kj} - \beta_{k'j}} = \frac{P\left(Y = k \mid \ldots, X_j = x+1, \ldots\right) / P\left(Y = k' \mid \ldots, X_j = x+1, \ldots\right)}{P\left(Y = k \mid \ldots, X_j = x, \ldots\right) / P\left(Y = k' \mid \ldots, X_j = x, \ldots\right)}$$

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

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

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

Regularized logistic regression

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

$$\min_{\beta_0,\boldsymbol{\beta}_1} \sum_{i=1}^{n} \left(-y_i\left(\beta_0 + x_i^\top \boldsymbol{\beta}_1\right) + \log\left(1 + e^{\beta_0 + x_i^\top \boldsymbol{\beta}_1}\right)\right) + \lambda J(\boldsymbol{\beta}_1)$$

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

$$(\beta_0^{(k+1)}, \boldsymbol{\beta_1}^{(\mathrm{k+1})}) = \arg\min_{\boldsymbol{\beta}} \left\{(\mathbf{z}^{(k)} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{W}^{(k)}(\mathbf{z}^{(k)} - \mathbf{X}\boldsymbol{\beta}) + \lambda J(\boldsymbol{\beta}_1)\right\}.$$

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

Sparse logistic regression

Algorithm (Coordinate descent for sparse logistic regression):

```
Let $\hat{\boldsymbol{\beta}}^{(0)} = (\hat\beta_0^{(0)}, \hat\beta_1^{(0)}, \ldots, \hat\beta_p^{(0)})^T$
For $k = 0, 1, 2, \ldots,$
  - Compute $p_1(x_i; \hat{\boldsymbol{\beta}}^{(k)}), z_i^{(k)}, w_{ii}^{(k)}, i = 1, \ldots, n.$
  - Let $\beta_0 = \sum_i [w_{ii}^{(k)}(z_i^{(k)} - \sum_{l=1}^p \hat\beta_l^{(k)} x_{il})] / \sum_i w_{ii}^{(k)}, \quad \beta_l = \hat\beta_l^{(k)}, l = 1, \ldots, p.$
    - for $j = 1, \ldots, p$ do
      - compute $r_{ij} = z_i^{(k)} - \beta_0 - \sum_{l \neq j} \beta_l x_{il}, i = 1, \ldots, n$
      - 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 $\beta_j = \text{soft}(u_j^{(k)}/v_j^{(k)}, \lambda/v_j^{(k)})$
    - $\hat{\boldsymbol{\beta}}_0^{(k+1)} = \beta_0, \hat\beta_j^{(k+1)} = \beta_j, j = 1, \ldots, p$
until convergence
```

Others: nonparametric classifier, robust loss, 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:

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

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,

$$\min_{f \in \mathcal{F}_{0/1}} \mathrm{E}_{X,Y}(1(Y \neq f(X)))$$

where $\mathcal{F}_{0/1}$ consists of function that maps to $\{0,1\}$.

The ERM solution is

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

- the choice of $\mathcal{F}_{0/1}$ **hard-classifier**:
  - perceptron: $\mathcal{F}_{0/1} = \{1(\beta_0 + \boldsymbol{\beta}^\top x) : \beta_0, \boldsymbol{\beta}\}$

Equivalent formulation, using $y_i \in \{-1,1\}$, and $\mathcal{F}_{\pm 1}$ consists of function that maps to $\{-1,1\}$:

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

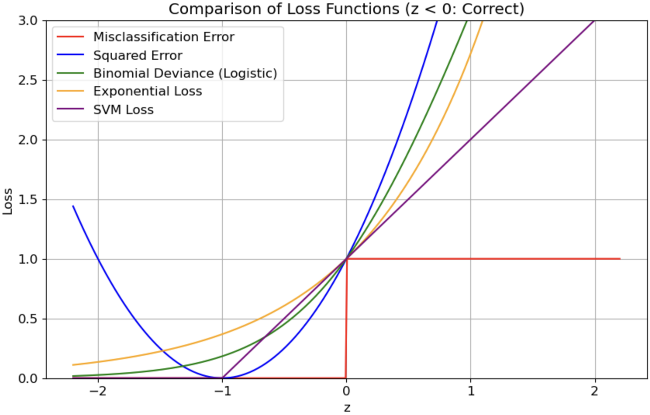
Using some smooth and convex surrogate function $\psi(z)$ for $1(z > 0)$ and relaxing the class of functions $\mathcal{F}$, solve

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

- 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: $\hat{Y}_i = \text{sign}(\hat{f}_n(X_i))$

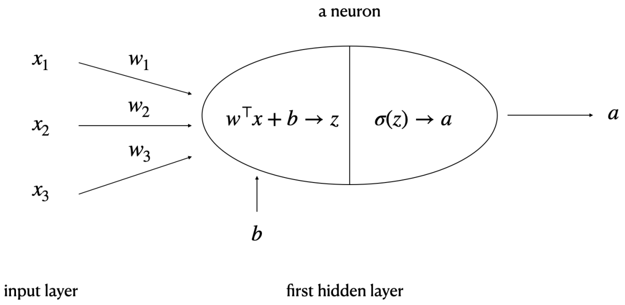
Let $z = -yf > 0$ indicate misclassification.

- Misclassification error: $1(z > 0)$
- Squared error: $(1 + z)^2$
- Binomial deviance (scaled): $\log(1 + \exp(2z))/\log(2)$
- Exponential loss: $\exp(z)$
- SVM loss: $(1 + z)_+$



More in future lessons. . .

Perceptron

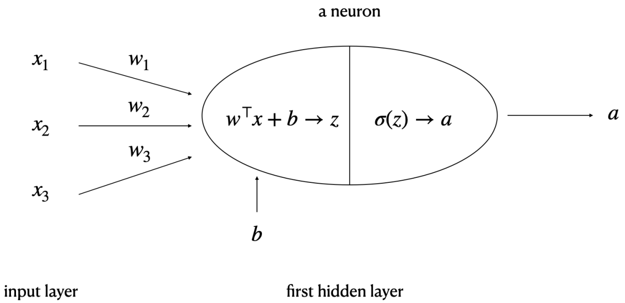


$\boldsymbol{x} = (x_1, x_2, x_3)$: three input features

$\sigma(\cdot)$: activation function– three simplest cases

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

The output $\hat{a}$ will be measured against the actual outcome value under some loss function.



Formally,

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

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

Rosenblatt's Perceptron

Rosenblatt's Perceptron: $\sigma(v) = 1(v > 0)$

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

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

- a **training epoch**: *one* loop over the whole training data
- the whole process repeats for K times or **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

$$\boldsymbol{\beta}^{(\mathrm{k+1})} = \boldsymbol{\beta}^{(\mathrm{k})} + \alpha \times \mathbf{X}^\top(\mathbf{y} - \mathbf{p}^{(k)}) = \boldsymbol{\beta}^{(\mathrm{k})} + \alpha \times \sum_i^n (y_i - p(x_i; \boldsymbol{\beta}^{(k)}))\boldsymbol{x}_i$$

The **on-line training** process is

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

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.

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

It can be implemented using gradient descent for OLS.

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

To classify based on the hard rule

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

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.