Machine Learning I

Linear classification

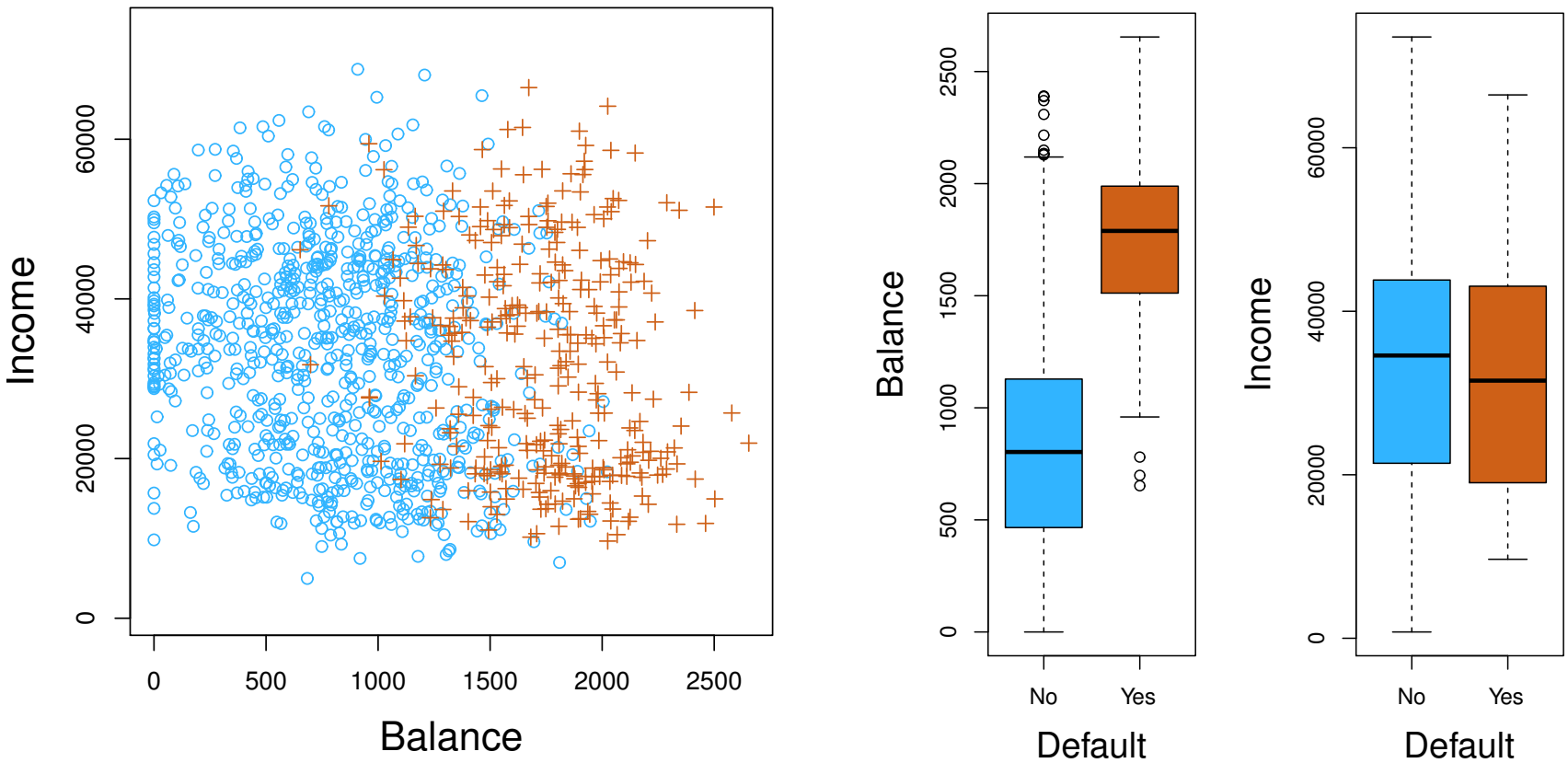
Souhaib Ben Taieb

University of Mons

Table of contents

Outline

An example



The **annual incomes** and **monthly credit card balances** of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.

Classification with the zero-one loss

Consider a **multi-class classification** problem with K categories where $\mathcal{Y} = \{C_1, \ldots, C_K\}$ and $\mathcal{X} \subseteq \mathbb{R}^p$.

What is the optimal classifier[1] $f : \mathcal{X} \to \mathcal{Y}$ with the **zero-one loss function** $L(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$?

[1] Optimal classifier means the classifier with the smallest expected prediction error.

Classification with the zero-one loss

Recall that the **optimal prediction (classification)** at x is given by

$$f(x) = \underset{h(x) \in \mathcal{Y}}{\operatorname{argmin}} \, \mathbb{E}[\mathbb{1}\{y \neq h(x)\}|x] = \underset{h(x) \in \mathcal{Y}}{\operatorname{argmax}} \, \mathbb{P}(y = h(x)|x),$$

In **binary classification** with $\mathcal{Y} = \{0, 1\}$, the **optimal prediction (classification)** at x is

$$f(x) = \mathbb{1}\left\{\mathbb{P}(y = 1|x) \geq \mathbb{P}(y = 0|x)\right\} = \mathbb{1}\left\{\mathbb{P}(y = 1|x) \geq 0.5\right\}.$$

Recall that the threshold (here, $\alpha = 0.5$) changes when we consider a **general binary classification loss** with different costs associated to different misclassifications.

Classification with the zero-one loss

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ and a hypothesis set $\mathcal{H}$, we could compute

$$g = \underset{h:\mathcal{X}\to\mathcal{Y}, h\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \neq h(x_i)\} = \underset{h:\mathcal{X}\to\mathcal{Y}, h\in\mathcal{H}}{\operatorname{argmax}} \hat{\mathbb{P}}(y = h(x)).$$

However, the zero-one loss function is **non-convex** and **discontinuous**. Solving for the optimal solution is an NP-hard combinatorial optimization problem.

Why not **represent** each category with **a real number** and solve a linear **regression problem**? In other words, instead of training a classifier $h : \mathcal{X} \to \mathcal{Y}$, we train a *regression function* $h : \mathcal{X} \to \mathbb{R}$ and classify using the *predicted real values*.

Why not solve a linear regression problem?

Consider a **multi-class classification** problem where we must classify patients present at the emergency room in a category from the set $\mathcal{Y} = \{\textit{stroke}, \text{drug overdose}, \text{epileptic seizure}\}$ based on their symptoms.

We can represent each class for example using the coding $\{1, 2, 3\}$ and solve a regression problem. However, by doing so we include an *ordering* which does not exist.

In **binary classification** where $\mathcal{Y} = \{0, 1\}$, we have

$$\mathbb{E}[y|x] = 0 \times \mathbb{P}(y = 0|x) + 1 \times \mathbb{P}(y = 1|x) = \mathbb{P}(y = 1|x).$$

In this case, since the regression function $\mathbb{E}[y|x]$ is equal to the conditional probability $\mathbb{P}(y = 1|x)$, we could use regression to estimate $\mathbb{P}(y|x)$.

However, linear regression (estimates) might produce probabilities **less than zero** or **larger than one**.

Why not solve a linear regression problem?

Consider a **multi-class classification** problem where we must classify patients present at the emergency room in a category from the set $\mathcal{Y} = \{\textit{stroke}, \text{drug overdose}, \text{epileptic seizure}\}$ based on their symptoms.

We can represent each class for example using the coding $\{1, 2, 3\}$ and solve a regression problem. However, by doing so we include an *ordering* which does not exist.

In **binary classification** where $\mathcal{Y} = \{0, 1\}$, we have

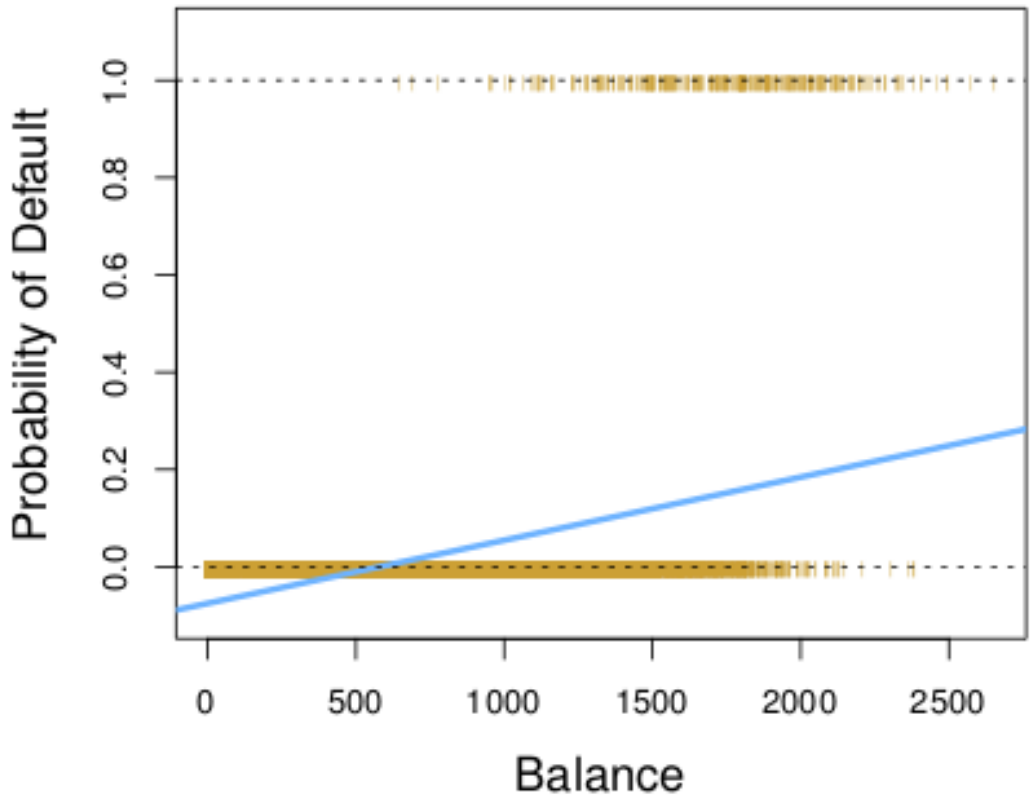
$$\mathbb{E}[y|x] = 0 \times \mathbb{P}(y = 0|x) + 1 \times \mathbb{P}(y = 1|x) = \mathbb{P}(y = 1|x).$$

In this case, since the regression function $\mathbb{E}[y|x]$ is equal to the conditional probability $\mathbb{P}(y = 1|x)$, we could use regression to estimate $\mathbb{P}(y|x)$.

However, linear regression (estimates) might produce probabilities **less than zero** or **larger than one**.

Why not solve a linear regression problem?

An example with $p = 1$ and $K = 2$.

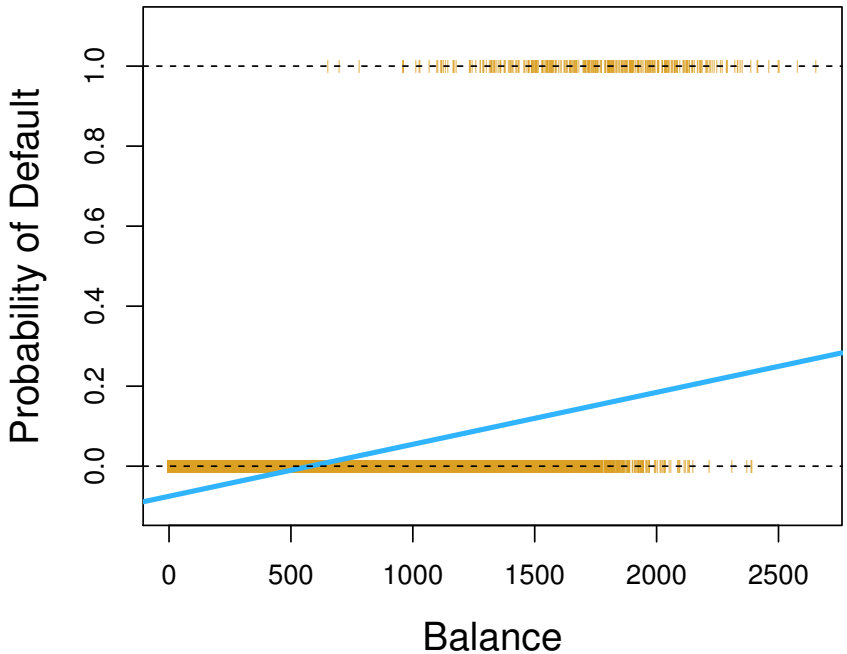


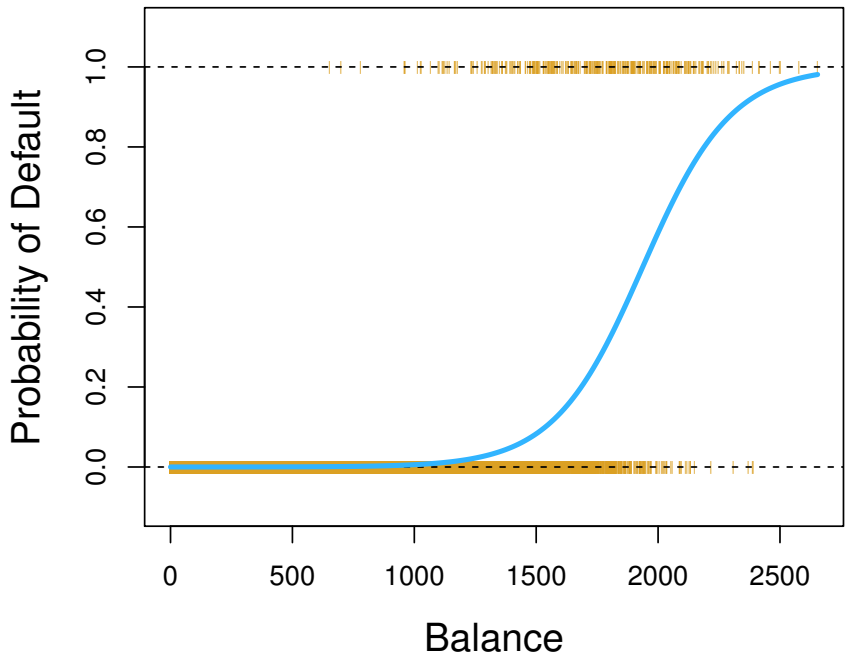
$$\mathbb{E}[y|x] = \mathbb{P}(y = 1|x) = \beta_0 + \beta_1 x$$

Outline

Linear vs logistic regression

An example with $p = 1$ and $K = 2$.





$$\mathbb{P}(y = 1|x) = \beta_0 + \beta_1 x$$

$$h : \mathcal{X} \to \mathbb{R}$$

$$\mathbb{P}(y = 1|x) = \sigma(\beta_0 + \beta_1 x)$$

$$\sigma(s) = \frac{e^s}{1 + e^s} = \frac{1}{1 + e^{-s}}$$

$$h : \mathcal{X} \to [0, 1]$$

Binary logistic regression

A binary ($K = 2$) logistic regression classifier is a **probabilistic classifier** , i.e. $h : \mathcal{X} \to [0, 1]$, where

$$\mathbb{P}(y = c|x) = p(c|x; \beta) = \begin{cases} h(x), & \text{if } c = 1 \ (= \mathcal{C}_1) \\ 1 - h(x), & \text{if } c = 0 \ (= \mathcal{C}_2) \end{cases}$$
$$= \begin{cases} \sigma(\beta^T x), & \text{if } c = 1 \ (= \mathcal{C}_1) \\ 1 - \sigma(\beta^T x), & \text{if } c = 0 \ (= \mathcal{C}_2) \end{cases}$$

where $\sigma(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, the coefficients/parameters β can be estimated using **maximum likelihood estimation**.

We classify by **maximizing** the conditional probabilities.

Binary logistic regression and log-odds

Note that $h(x) = \sigma(\beta^T x)$ is **not** linear in x even if the **decision boundary** of (linear) logistic regression is defined by a linear equation in x,

Instead, the **log-odds** or **logit transformation** of $h(x)$ is linear in x, i.e.

$$\begin{aligned}
\log\left(\frac{\mathbb{P}(y=\mathcal{C}_1|x)}{\mathbb{P}(y=\mathcal{C}_2|x)}\right) &= \log\left(\frac{\mathbb{P}(y=\mathcal{C}_1|x)}{1-\mathbb{P}(y=\mathcal{C}_1|x)}\right) \\
&= \log\left(\frac{h(x)}{1-h(x)}\right) \\
&= \beta^T x.
\end{aligned}$$

Binary logistic regression and log-odds

- The quantity $\mathbb{P}(y = \mathcal{C}_1|x)/\mathbb{P}(y = \mathcal{C}_2|x)$, called the **odds**, take values between 0 and ∞. The **log-odds** values are in $(-\infty, +\infty)$.
- For the credit card example, values of the odds close to 0 and ∞ indicate very low and very high probabilities of default, respectively.
- For example, if $\mathbb{P}(y = 1|x) = 0.2$, we have an odds of $0.2/0.8 = 1/4$. In other words, 1 in 5 people will defaut on average.
- If $\mathbb{P}(y = 1|x) = 0.9$, we have an odds of $0.9/0.1 = 9/1$. On average, nine out of every ten people with an odds of 9 will default.

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

MLE in binary logistic regression

The **(conditional) likelihood** is given by

$$\begin{aligned}
&\mathcal{L}(\beta; \mathcal{D}) \\
&= p(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\
&= \Pi_{i=1}^{n} p(y_i | x_i; \beta) \\
&= \prod_{i:y_i=1} \mathbb{P}(y = 1 | x_i; \beta) \prod_{i:y_i=0} \mathbb{P}(y = 0 | x_i; \beta) \\
&= \prod_{i:y_i=1} \sigma(\beta^T x_i) \prod_{i:y_i=0} (1 - \sigma(\beta^T x_i)),
\end{aligned}$$

The **(conditional) log-likelihood** is given by

$$\begin{aligned}
&\log \mathcal{L}(\beta; \mathcal{D}) \\
&= \sum_{i:y_i=1} \log(\sigma(\beta^T x_i)) + \sum_{i:y_i=0} \log(1 - \sigma(\beta^T x_i)) \\
&= \Sigma_{i=1}^{n} y_i \log(\sigma(\beta^T x_i)) + (1 - y_i) \log(1 - \sigma(\beta^T x_i)).
\end{aligned}$$

The cross-entropy loss function

Maximizing the log-likelihood function is equivalent to **minimize** the **cross-entropy loss** or the **logistic loss**:

$$
\begin{aligned}
& E_{\text{in}}(\beta) \\
&= -\frac{1}{n} \log \mathcal{L}(\beta; \mathcal{D}) \\
&= -\frac{1}{n} \Sigma_{i=1}^{n} y_i log(\sigma(\beta^T x_i)) + (1 - y_i) log(\sigma(-\beta^T x_i)) \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \left[y_i log\left(\frac{1}{\sigma(\beta^T x_i)}\right) + (1 - y_i) log\left(\frac{1}{1 - \sigma(\beta^T x_i)}\right) \right] \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \text{CE}\left(\{y_i, 1 - y_i\}, \{\sigma(\beta^T x_i), 1 - \sigma(\beta^T x_i)\}\right)
\end{aligned}
$$

where

$$
\text{CE}(\{p, 1-p\}, \{q, 1-q\}) = p \ log \ \frac{1}{q} + (1-p) \ log \ \frac{1}{1-q},
$$

is the **cross-entropy** (from information theory) between two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes.

The cross-entropy loss function

Maximizing the log-likelihood function is equivalent to **minimize** the **cross-entropy loss** or the **logistic loss**:

$$
\begin{aligned}
& E_{\text{in}}(\beta) \\
&= -\frac{1}{n} \log \mathcal{L}(\beta; \mathcal{D}) \\
&= -\frac{1}{n} \Sigma_{i=1}^{n} y_i log(\sigma(\beta^T x_i)) + (1 - y_i) log(\sigma(-\beta^T x_i)) \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \left[y_i log\left(\frac{1}{\sigma(\beta^T x_i)}\right) + (1 - y_i) log\left(\frac{1}{1 - \sigma(\beta^T x_i)}\right) \right] \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \text{CE}\left(\{y_i, 1 - y_i\}, \{\sigma(\beta^T x_i), 1 - \sigma(\beta^T x_i)\}\right)
\end{aligned}
$$

where

$$
\text{CE}(\{p, 1-p\}, \{q, 1-q\}) = p \ log \ \frac{1}{q} + (1-p) \ log \ \frac{1}{1-q},
$$

is the **cross-entropy** (from information theory) between two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes.

The cross-entropy loss function

Maximizing the log-likelihood function is equivalent to **minimize** the **cross-entropy loss** or the **logistic loss**:

$$
\begin{aligned}
& E_{\text{in}}(\beta) \\
&= -\frac{1}{n} \log \mathcal{L}(\beta; \mathcal{D}) \\
&= -\frac{1}{n} \Sigma_{i=1}^{n} y_i log(\sigma(\beta^T x_i)) + (1 - y_i) log(\sigma(-\beta^T x_i)) \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \left[y_i log \left(\frac{1}{\sigma(\beta^T x_i)} \right) + (1 - y_i) log \left(\frac{1}{1 - \sigma(\beta^T x_i)} \right) \right] \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \text{CE} \left(\{y_i, 1 - y_i\}, \{\sigma(\beta^T x_i), 1 - \sigma(\beta^T x_i)\} \right)
\end{aligned}
$$

where

$$
\text{CE}(\{p, 1-p\}, \{q, 1-q\}) = p \ log \ \frac{1}{q} + (1-p) \ log \ \frac{1}{1-q},
$$

is the **cross-entropy** (from information theory) between two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes.

The cross-entropy loss function

Maximizing the log-likelihood function is equivalent to **minimize** the **cross-entropy loss** or the **logistic loss**:

$$
\begin{aligned}
& E_{\text{in}}(\beta) \\
&= -\frac{1}{n} \log \mathcal{L}(\beta; \mathcal{D}) \\
&= -\frac{1}{n} \Sigma_{i=1}^{n} y_i log(\sigma(\beta^T x_i)) + (1 - y_i) log(\sigma(-\beta^T x_i)) \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \left[y_i log \left(\frac{1}{\sigma(\beta^T x_i)} \right) + (1 - y_i) log \left(\frac{1}{1 - \sigma(\beta^T x_i)} \right) \right] \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \text{CE} \left(\{y_i, 1 - y_i\}, \{\sigma(\beta^T x_i), 1 - \sigma(\beta^T x_i)\} \right)
\end{aligned}
$$

where

$$
\text{CE}(\{p, 1-p\}, \{q, 1-q\}) = p \ log \ \frac{1}{q} + (1-p) \ log \ \frac{1}{1-q},
$$

is the **cross-entropy** (from information theory) between two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes.

The cross-entropy loss function

Maximizing the log-likelihood function is equivalent to **minimize** the **cross-entropy loss** or the **logistic loss**:

$$
\begin{aligned}
& E_{\text{in}}(\beta) \\
&= -\frac{1}{n} \log \mathcal{L}(\beta; \mathcal{D}) \\
&= -\frac{1}{n} \Sigma_{i=1}^{n} y_i log(\sigma(\beta^T x_i)) + (1 - y_i) log(\sigma(-\beta^T x_i)) \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \left[y_i log \left(\frac{1}{\sigma(\beta^T x_i)} \right) + (1 - y_i) log \left(\frac{1}{1 - \sigma(\beta^T x_i)} \right) \right] \\
&= \frac{1}{n} \Sigma_{i=1}^{n} \text{CE} \Big(\{y_i, 1 - y_i\}, \{\sigma(\beta^T x_i), 1 - \sigma(\beta^T x_i)\} \Big)
\end{aligned}
$$

where

$$
\text{CE}(\{p, 1-p\}, \{q, 1-q\}) = p \ log \ \frac{1}{q} + (1-p) \ log \ \frac{1}{1-q},
$$

is the **cross-entropy** (from information theory) between two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes.

MLE in binary logistic regression

One can show that the solution to the logistic regression problem (i.e. with the **logistic loss**) allow for the recovery of the actual solution to the original classification problem (i.e. with the **zero-one loss**).

One advantage of the logistic loss function is that it **continuous** and **convex** which is more tractable for optimization.

In contrast to least squares regression, the previous optimizaiton problem does **not** have a **closed-form solution**. Nonlinear **iterative optimization methods** such as the Newton–Raphson algorithm are used.

Interpretation

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.6513	0.3612	-29.5	< 0.0001
`balance`	0.0055	0.0002	24.9	< 0.0001

- **Linear regression**
  - β_1 gives the average change in y associated with a one-unit increase in x
- **Logistic regression**
  - Increasing x by one unit changes the **log odds** by β_1, or equivalently it multiplies the odds by e^{β_1}.

Making predictions

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.6513	0.3612	-29.5	< 0.0001
`balance`	0.0055	0.0002	24.9	< 0.0001

What is our estimated probability of default for someone with a balance of \$1000 and \$2000?

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-3.5041	0.0707	-49.55	< 0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

What is our estimated probability of default for a student and non-student?

Logistic regression with several variables

$p = 1$

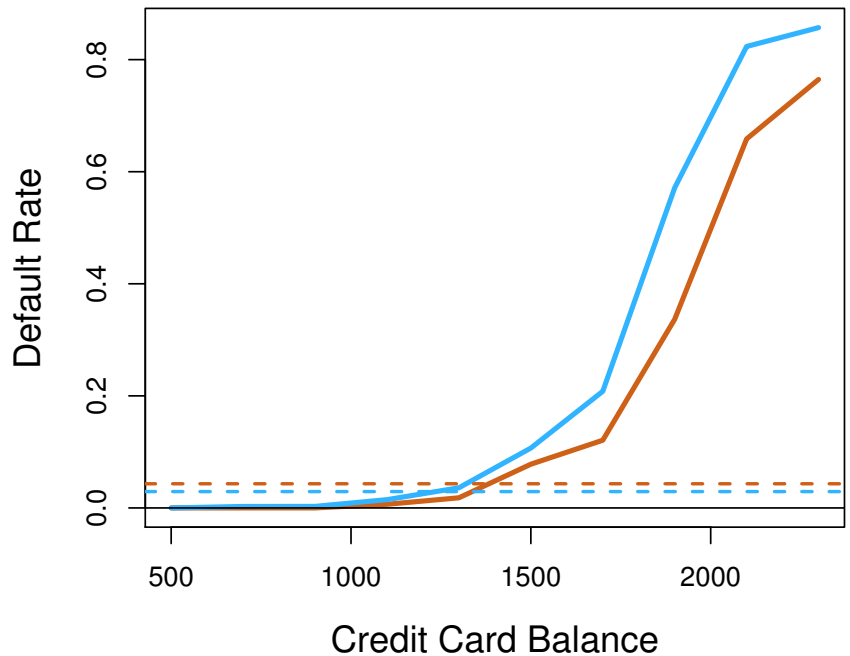
	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-3.5041	0.0707	-49.55	< 0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

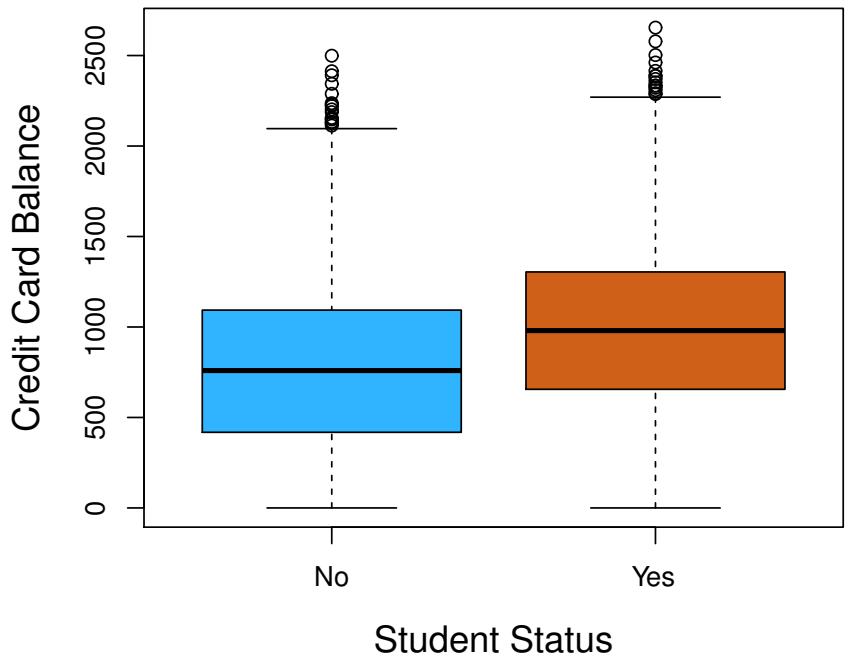
$p = 3$

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Why is the coefficient for student negative with $p = 3$, while it was positive with $p = 1$?

Logistic regression with several variables





- Students tend to have higher balances than non-students, so their marginal default rate is higher than for non-students.
- But for each level of balance, students default less than non-students.

Multiclass logistic regression

We can generalize logistic regression to K classes, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$, as follows:

$$\mathbb{P}(y = \mathcal{C}_k | x) = p_k(x; \beta^{(k)}) = \frac{e^{\beta^{(k)^T} x}}{\sum_{l=1}^{K} e^{\beta^{(l)^T} x}},$$

where $k = 1, 2, \ldots, K-1$. The model is trained by minimiznig the **multi-class cross-entropy** with a **one-hot encoding** of the classes.

Multiclass logistic regression is also referred to as **multinomial regression**.

Outline

Discriminant Analysis

- With **logistic regression**, we directly parametrize the conditional distribution $p_{y|x}$ using a logistic transformation of a linear combination of the inputs.
- With **discriminant analysis**, we use the Bayes theorem to compute

$$\begin{aligned}\mathbb{P}(y = C_k|x) = p_k(x) &= \frac{f(x|y = C_k) \cdot \mathbb{P}(y = C_k)}{f(x)} \\ &= \frac{f(x|y = C_k) \cdot \mathbb{P}(y = C_k)}{\sum_{l=1}^{K} f(x|y = C_l) \cdot \mathbb{P}(y = C_l)} = \frac{f_k(x) \cdot \pi_k}{\sum_{l=1}^{K} f_l(x) \cdot \pi_l}\end{aligned}$$

where

- $f_k(x)$ is the density for x in class C_k.
- π_k is the marginal or prior probability for class C_k.

Discriminant Analysis

- With **logistic regression**, we directly parametrize the conditional distribution $p_{y|x}$ using a logistic transformation of a linear combination of the inputs.
- With **discriminant analysis**, we use the Bayes theorem to compute

$$\begin{aligned}\mathbb{P}(y = C_k|x) &= p_k(x) \\ &= \frac{f(x|y = C_k) \cdot \mathbb{P}(y = C_k)}{f(x)} \\ &= \frac{f(x|y = C_k) \cdot \mathbb{P}(y = C_k)}{\sum_{l=1}^{K} f(x|y = C_l) \cdot \mathbb{P}(y = C_l)} = \frac{f_k(x) \cdot \pi_k}{\sum_{l=1}^{K} f_l(x) \cdot \pi_l}\end{aligned}$$

where

- $f_k(x)$ is the density for x in class C_k.
- π_k is the marginal or prior probability for class C_k.

Discriminant Analysis

For a given input x, we obtain the predicted class by **maximizing** the conditional probability $p_k(x)$ as a function of k. In other words, we compute

$$\begin{aligned} & \underset{k \in \{1,2,\ldots,K\}}{\text{argmax}} \quad p_k(x) \\ = & \underset{k \in \{1,2,\ldots,K\}}{\text{argmax}} \quad \frac{f_k(x) \cdot \pi_k}{\sum_{l=1}^{K} f_l(x) \cdot \pi_l} \\ = & \underset{k \in \{1,2,\ldots,K\}}{\text{argmax}} \quad f_k(x) \cdot \pi_k \end{aligned}$$

The **decision boundary** between each pair of classes k_1 and k_2, where $k_1, k_2 \in \{1, 2, \ldots, K\}$ and $k_1 \neq k_2$, is described by the set

$$\{x : p_{k_1}(x) = p_{k_2}(x)\}$$

Linear Discriminant Analysis

We will focus on the univariate case ($p = 1$) with normal distributions. However, this approach is general, and other distributions can be used as well.

The univariate Normal density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

where μ_k is the mean, and σ_k^2 is the variance (in class C_k).

With **Linear Discriminant Analysis (LDA)**, we assume $\sigma_k = \sigma$, which gives

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{k} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Linear Discriminant Analysis

We will focus on the univariate case ($p = 1$) with normal distributions. However, this approach is general, and other distributions can be used as well.

The univariate Normal density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

where μ_k is the mean, and σ_k^2 is the variance (in class C_k).

With **Linear Discriminant Analysis (LDA)**, we assume $\sigma_k = \sigma$, which gives

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{k} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Linear Discriminant Analysis

We will focus on the univariate case ($p = 1$) with normal distributions. However, this approach is general, and other distributions can be used as well.

The univariate Normal density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

where μ_k is the mean, and σ_k^2 is the variance (in class C_k).

With **Linear Discriminant Analysis (LDA)**, we assume $\sigma_k = \sigma$, which gives

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{k} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Why is LDA a linear classifier?

$$\underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ p_k(x) \equiv \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ \pi_k \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)$$

$$= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right)$$

$$= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ -\frac{1}{2\sigma^2}(x-\mu_k)^2 + log(\pi_k)$$

$$= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ -\frac{1}{2\sigma^2}x^2 + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

$$= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) := \delta_k(x)$$

For LDA, the **discriminant score** $\delta_k(x)$ is a linear function of x, which is due to the assumption $\sigma_k = \sigma$.

The **decision boundary** between each pair of classes k_1 and k_2 can also be described by the set $\{x : \delta_{k_1}(x) = \delta_{k_2}(x)\}$.

Why is LDA a linear classifier?

$$\begin{aligned}
\underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; p_k(x) &\equiv \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; \pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}(x-\mu_k)^2 + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}x^2 + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) := \delta_k(x)
\end{aligned}$$

For LDA, the **discriminant score** $\delta_k(x)$ is a linear function of x, which is due to the assumption $\sigma_k = \sigma$.

The **decision boundary** between each pair of classes k_1 and k_2 can also be described by the set $\{x : \delta_{k_1}(x) = \delta_{k_2}(x)\}$.

Why is LDA a linear classifier?

$$\begin{aligned}
\underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; p_k(x) &\equiv \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; \pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}(x-\mu_k)^2 + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}x^2 + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) := \delta_k(x)
\end{aligned}$$

For LDA, the **discriminant score** $\delta_k(x)$ is a linear function of x, which is due to the assumption $\sigma_k = \sigma$.

The **decision boundary** between each pair of classes k_1 and k_2 can also be described by the set $\{x : \delta_{k_1}(x) = \delta_{k_2}(x)\}$.

Why is LDA a linear classifier?

$$
\begin{aligned}
\underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; p_k(x) &\equiv \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; \pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}(x-\mu_k)^2 + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; -\frac{1}{2\sigma^2}x^2 + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}} \; x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) := \delta_k(x)
\end{aligned}
$$

For LDA, the **discriminant score** $\delta_k(x)$ is a linear function of x, which is due to the assumption $\sigma_k = \sigma$.

The **decision boundary** between each pair of classes k_1 and k_2 can also be described by the set $\{x : \delta_{k_1}(x) = \delta_{k_2}(x)\}$.

Why is LDA a linear classifier?

$$
\begin{aligned}
\underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ p_k(x) &\equiv \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ \pi_k \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ -\frac{1}{2\sigma^2}(x-\mu_k)^2 + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ -\frac{1}{2\sigma^2}x^2 + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \\
&= \underset{k\in\{1,2,\ldots,K\}}{\text{argmax}}\ x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) := \delta_k(x)
\end{aligned}
$$

For LDA, the **discriminant score** $\delta_k(x)$ is a linear function of x, which is due to the assumption $\sigma_k = \sigma$.

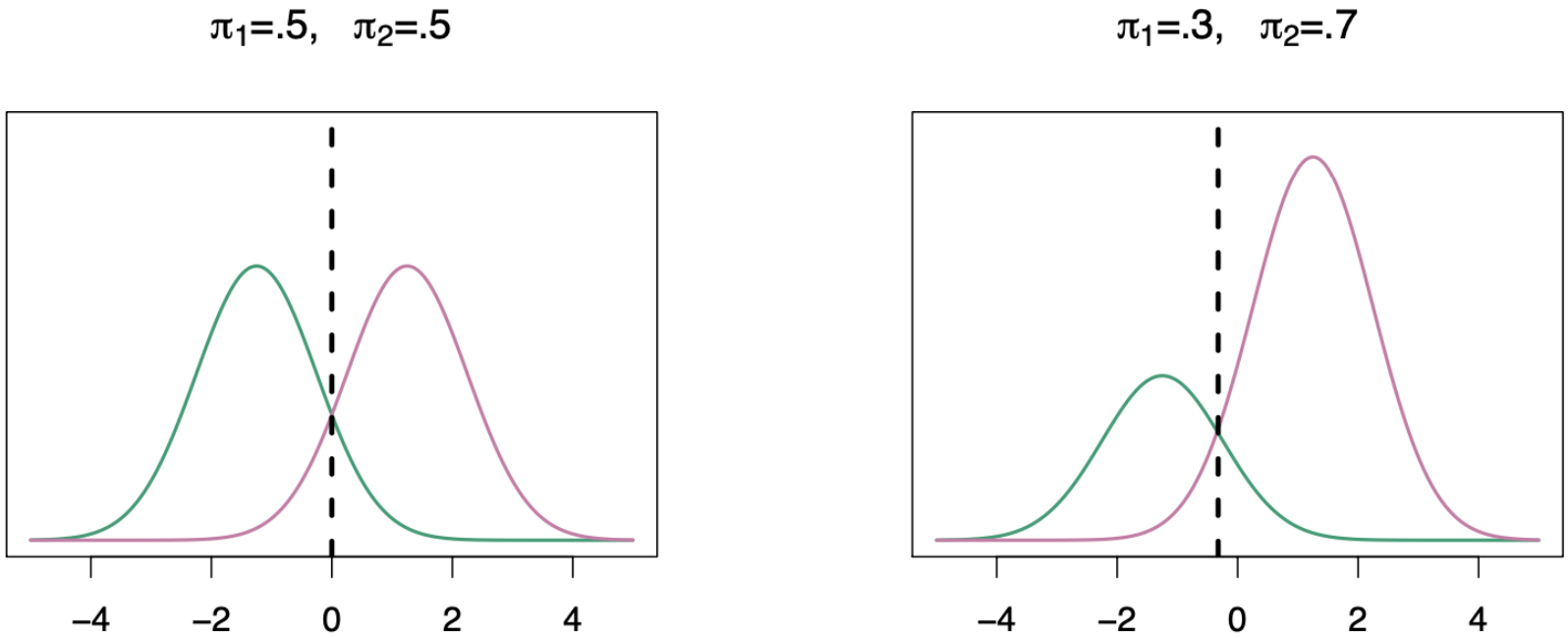
The **decision boundary** between each pair of classes k_1 and k_2 can also be described by the set $\{x : \delta_{k_1}(x) = \delta_{k_2}(x)\}$.

The LDA decision boundary ($K = 2$ and $\pi_1 = 0.5$)

For $K = 2$, and if $\pi_1 = \pi_2 = 0.5$, we have

$$
\begin{aligned}
&\{x : \delta_1(x) = \delta_2(x)\}. \\
&\equiv \left\{ x : x\frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + log(\pi_1) = x\frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + log(\pi_2) \right\} \\
&\equiv \left\{ x : x\frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} = x\frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} \right\} \\
&\equiv \left\{ x : 2x(\mu_1 - \mu_2) = \mu_1^2 - \mu_2^2 \right\} \\
&\equiv \left\{ x : x = \frac{\mu_1 + \mu_2}{2} \right\}
\end{aligned}
$$

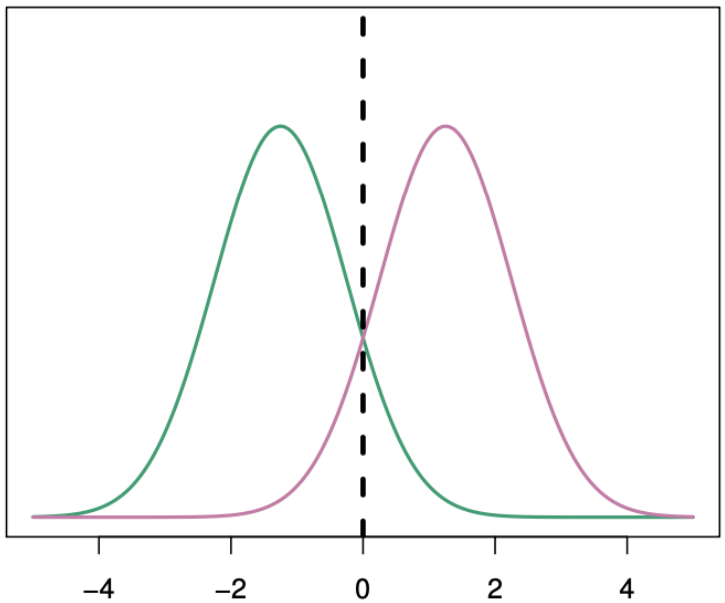
The LDA decision boundary ($K = 2$)

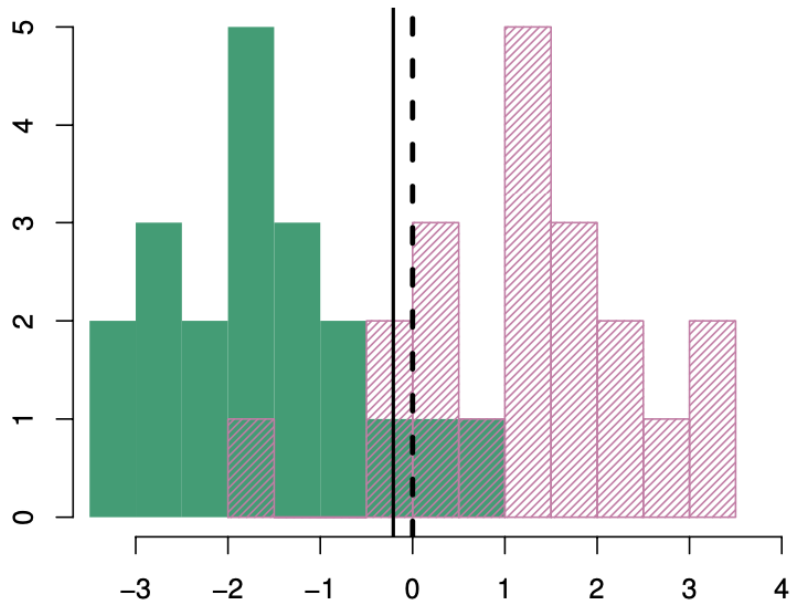


- We classify a new point according to which density is highest.
- When the priors are different, we take them into account as well. On the right, we favor the pink class (the decision boundary has shifted to the left).

Estimation of parameters

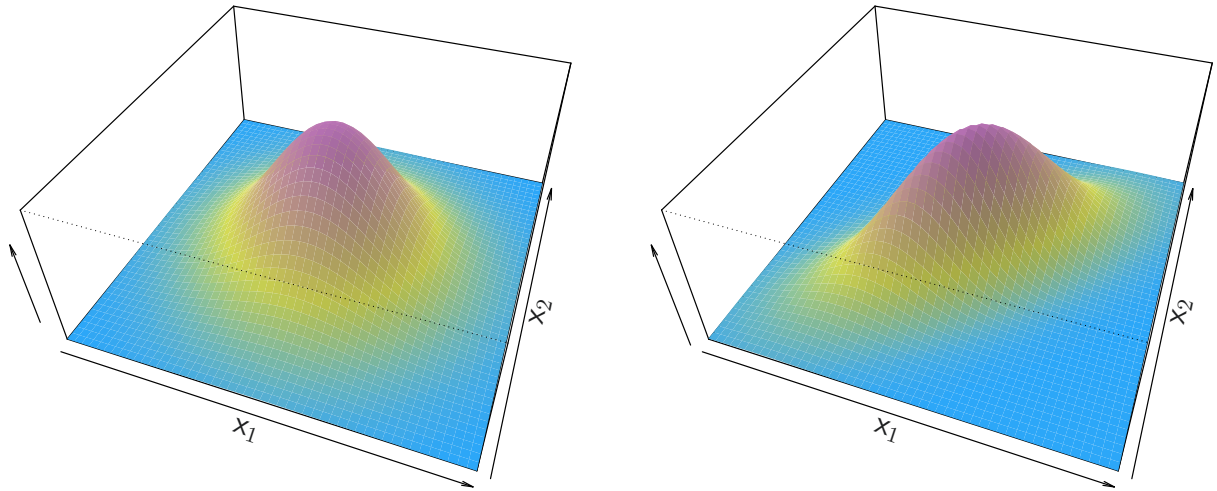
Example with $\mu_1 = -1.5$, $\mu_2 = 1.5$, $\pi_1 = \pi_2 = 0.5$, and $\sigma^2 = 1$.





In practice, we do **not** know these parameters. Given a dataset, we estimate the parameters, i.e. $\hat{\mu}_k, \hat{\pi}_k, \hat{\sigma}_k^2$, and plug them into the equation.

LDA with $p > 1$

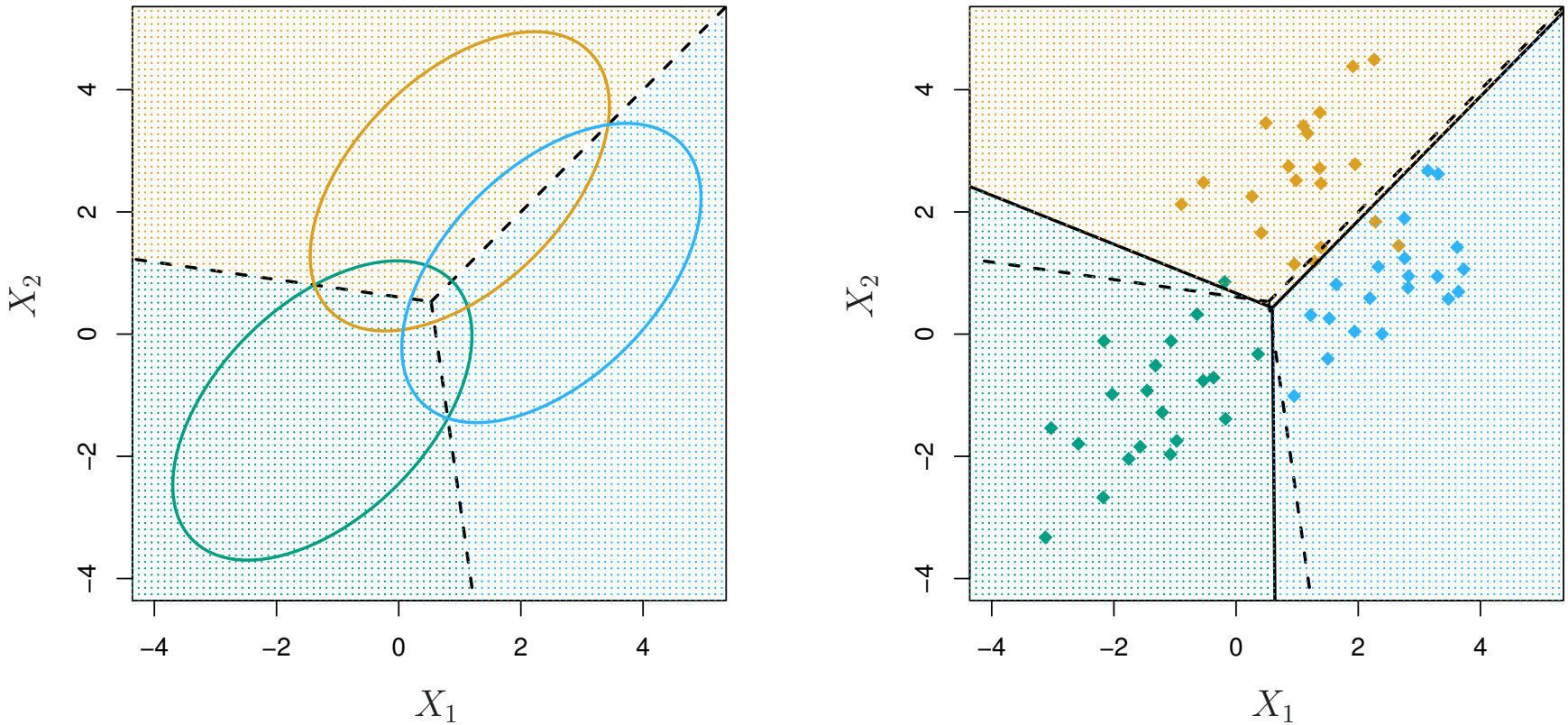


$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\mathbf{\Sigma}^{-1}(x-\mu)\right)$$

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

Despite its complex form, $\delta_k(x)$ is a linear function.

LDA with $p = 2$ and $K = 3$



The dashed and solid lines are the Bayes and estimated decision boundaries, respectively. In this example, we have $\pi_1 = \pi_2 = \pi_3 = 1/3$.

Outline

Other forms of Discrminant Analysis

$$\mathbb{P}(y = C_k|x) = \frac{f_k(x) \cdot \pi_k}{\sum_{l=1}^{K} f_l(x) \cdot \pi_l}$$

When $f_k(x)$ are **Gaussian** densities, with the **same** covariance matrix Σ in each class, i.e. $\Sigma_k = \Sigma$, this leads to LDA.

By altering the forms of $f_k(x)$, we obtain different classifiers.

We will present the following two classifiers:

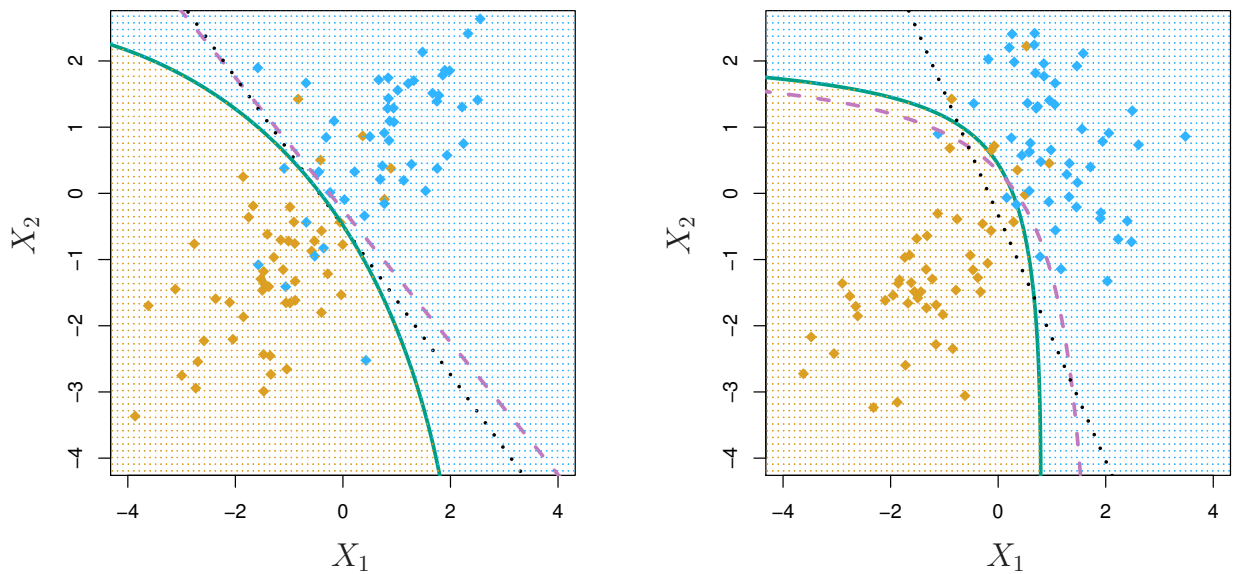
- The **Quadratic discriminant analysis** (QDA) classifier
- The **Naive Bayes** classifier

Quadratic discriminant analysis

With Gaussian densities but different Σ_k in each class, we obtain **quadratic discriminant analysis (QDA)**. The discriminant scores are given by

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

Because the Σ_k are different, the quadratic term matters.



Naive Bayes

If we make a **conditional independence** assumption in each class, i.e.

$$f_k(x) = \prod_{j=1}^{p} f_{jk}(x_j),$$

we obtain the **Naive Bayes** classifier, which is useful when p is large.

With **Gaussian** densities, this means the covariance matrices Σ_k are diagonal:

$$\delta_k(x) \propto \log\left[\pi_k \prod_{j=1}^{p} f_{jk}(x_j)\right] = -\frac{1}{2}\sum_{j=1}^{p} \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2} + \log(\pi_k)$$

It can be used with **mixed variables** (quantitative and qualitative). For qualitative variables, $f_{jk}(x_j)$ becomes a probability mass function (histogram) over discrete categories.

Naive Bayes

If we make a **conditional independence** assumption in each class, i.e.

$$f_k(x) = \prod_{j=1}^{p} f_{jk}(x_j),$$

we obtain the **Naive Bayes** classifier, which is useful when p is large.

With **Gaussian** densities, this means the covariance matrices $\mathbf{\Sigma}_k$ are diagonal:

$$\delta_k(x) \propto \log\left[\pi_k \prod_{j=1}^{p} f_{jk}(x_j)\right] = -\frac{1}{2}\sum_{j=1}^{p} \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2} + \log(\pi_k)$$

It can be used with **mixed variables** (quantitative and qualitative). For qualitative variables, $f_{jk}(x_j)$ becomes a probability mass function (histogram) over discrete categories.

Outline

The accuracy

- The average of the zero-one loss is the **error rate** or **misclassification rate**.
- While we previously mentionned that it is hard to optimize the error rate, it is still a useful metric to track. Equivalently, we can track the **accuracy**, or fraction of correct classifications
- For the following example, we have a 2.75% misclassification rate ((23+252)/10000).

	No	Yes	Total
No	9644	**23**	9667
Yes	**252**	81	333
Total	9896	104	10000

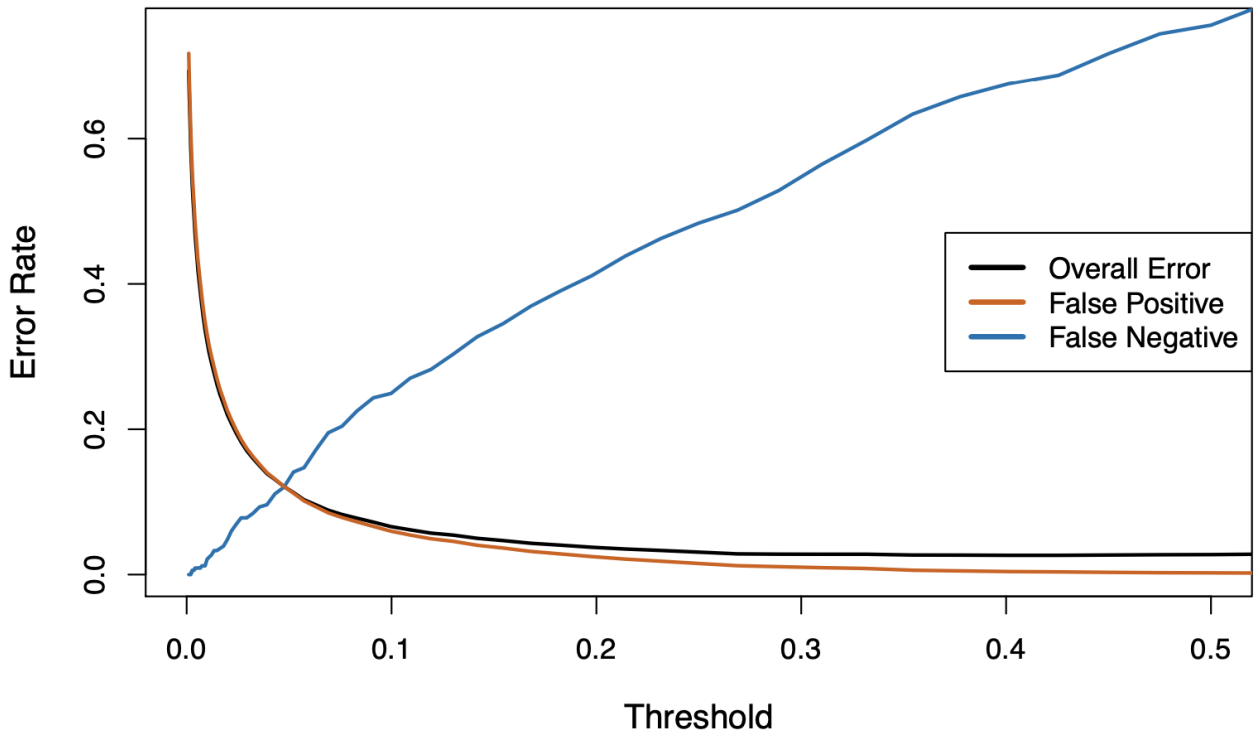
The limitations of accuracy

	No	Yes	Total
No	9644	**23**	9667
Yes	**252**	81	333
Total	9896	104	10000

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

- The accuracy is $\frac{\text{TP+TN}}{P+N} = \frac{\text{TP+TN}}{\text{TP+FN+TN+FP}}$
  - Always predicting No would give 333/10000 errors, or 3.33%.
  - Accuracy is highly sensitive to **class imbalance**.
- Of the true No's, we make 23/9667 = 0.2% errors
  - The **False positive rate**: $FPR = \frac{FP}{N} = \frac{FP}{TN+FP}$
- Of the true Yes's, we make 252/333 = **75.7%** errors!
  - The **False negative rate**: $FNR = \frac{FN}{P} = \frac{FN}{TP+FN}$

Varying the threshold



To reduce the false negative rate, we may want to reduce the threshold to 0.1 or less.

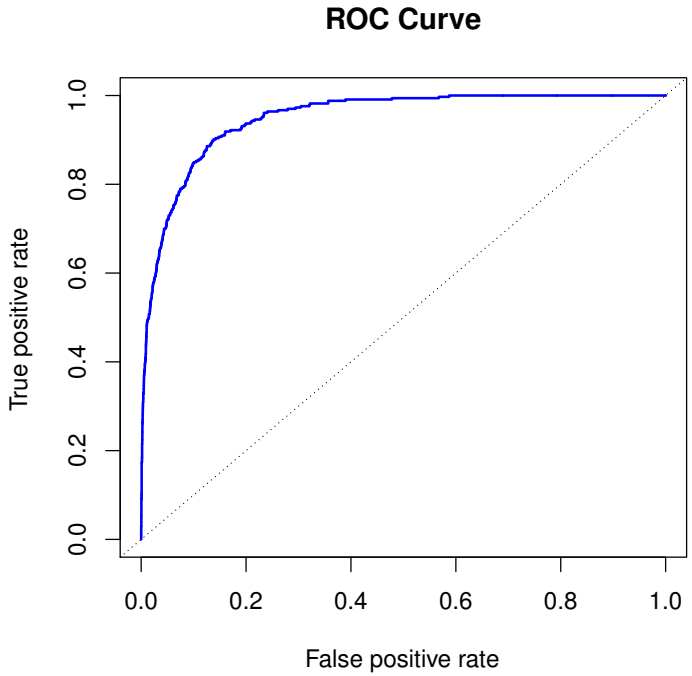
Types of errors

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

- **Sensitivity** and **specificity** are useful metrics even under class imbalance.
  - Sensitivity (True positive rate, 1 - FNR): $\frac{\text{TP}}{\text{TP+FN}} = \frac{TP}{P}$
  - Specificity (True negative rate, 1 - FPR): $\frac{\text{TN}}{\text{TN+FP}} = 1 - \frac{FP}{N}$
  - Tradeoff between sensitivity and specificity

The ROC curve



		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

- The ROC plot displays $(\text{FPR}(\alpha), \text{TPR}(\alpha))$ for all thresholds $\alpha \in [0, 1]$
- We can summarize the overall performance by computing the area under the curve (AUC) where AUC $\in [0, 1]$. The higher the better.
- With a classifier that randomly assign to the positive and negative class with probability q and $1 - q$, respectively, we have
  - $\text{TPR} = \text{TP}/\text{P} = (q \times \text{P})/\text{P} = q$
  - $\text{FPR} = \text{FP}/\text{N} = (q \times \text{N})/\text{N} = q$

Outline

Consider a **multi-class classification** problem with K categories where $\mathcal{Y} = \{C_1, \ldots, C_K\}$ and $\mathcal{X} \subseteq \mathbb{R}^p$. Recall that we assign an input x to the class k that maximizes $\mathbb{P}(y = \mathcal{C}_k|x)$. Equivalently, we can set K as the **baseline class** and assign the input x to the class k that maximizes the log-odds

$$\log\left(\frac{\mathbb{P}(y = \mathcal{C}_k|x)}{\mathbb{P}(y = \mathcal{C}_K|x)}\right),$$

where $k = 1, 2, \ldots, K$.

Let us examine the specific form of the log-odds for different classifiers to understand their similarities and differences.

Linear Discriminant Analysis

$$
\begin{aligned}
& \log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right) \\
= \; & \log\left(\frac{\pi_k \exp\left(-\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}^{-1}(x-\mu_k)\right)}{\pi_K \exp\left(-\frac{1}{2}(x-\mu_K)^T\mathbf{\Sigma}^{-1}(x-\mu_K)\right)}\right) \\
= \; & \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}^{-1}(x-\mu_k) \\
& + \frac{1}{2}(x-\mu_K)^T\mathbf{\Sigma}^{-1}(x-\mu_K) \\
= \; & \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}(\mu_k+\mu_K)^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_K) \\
& + x^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_K) \\
= \; & a_k + \sum_{j=1}^{p} b_{kj}x_j,
\end{aligned}
$$

- LDA, like logistic regression, assumes that the log odds is **linear** in x.
- In LDA, the coefficients are estimated by assuming a normal distribution within each class.
- In logistic regression, the coefficients are estimated using MLE.
- We expect LDA to outperform logistic regression if the normality assumption (approximately) holds.

Quadratic Discriminant Analysis

For QDA, using similar calculations, we obtain

$$a_k + \sum_{j=1}^{p} b_{kj} x_j + \sum_{j=1}^{p} \sum_{l=1}^{p} c_{kjl} x_j x_l$$

- As the name suggests, QDA assumes that the log odds is **quadratic** in x.
- LDA is a special case of QDA with $c_{kjl} = 0$ for all $j = 1, \ldots, p, l = 1, \ldots, p$, and $k = 1, \ldots, K$.

Naive Bayes

$$
\begin{aligned}
& \log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right) \\
= \quad & \log\left(\frac{\pi_k \prod_{j=1}^{p} f_{kj}(x_j)}{\pi_K \prod_{j=1}^{p} f_{Kj}(x_j)}\right) \\
= \quad & \log\left(\frac{\pi_k}{\pi_K}\right) + \sum_{j=1}^{p} \log\left(\frac{f_{kj}(x_j)}{f_{Kj}(x_j)}\right) \\
= \quad & a_k + \sum_{j=1}^{p} g_{kj}(x_j)
\end{aligned}
$$

With Naive Bayes, the log-odds takes the form of a **generalized additive model**.

Some useful observations

- Any classifier with a linear decision boundary is a special case of naive Bayes with $g_{kj}(x_j) = b_{kj}x_j$. In particular, this means that **LDA is a special case of naive Bayes**! This is not at all obvious since each method makes very different assumptions.
- Naive Bayes with a one-dimensional Gaussian distribution is a **special case of LDA** with the covariance matrix restricted to be a digonal matrix.
- Neither QDA nor naive Bayes is a special case of the other. Naive Bayes can produce a **more flexible fit**, since any choice can be made for g_{kj}. However, QDA has the potential to be more accurate in settings where **interactions** among the predictors are important in discriminating between classes.
- None of these methods uniformly **dominates** the others. The performance will notably depend on the **true distribution** of the predictors in each of the K classes as well as the values of n and p. It is always a question of **bias-variance trade-off**.

K-Nearest Neighbors

- KNN is a completely **non-parametric approach**: no assumptions are made about the shape of the decision boundary.
- We expect KNN to dominate LDA and logistic regression when the decision boundary is **highly non-linear**, provided that n is **very large** and p is **small**.
- KNN tends to **reduce the bias** while incurring **a lot of variance**. KNN requires a lot of observations relative to the number of predictors.
- Event if the decision boundary is **non-linear** but n is only modest, or p is not very small, then QDA may be preferred to KNN.
- Unlike logistic regression, KNN is less **interpretable**: we don't get a table of coefficients.