Statistical learning - Vinciotti (2022)

Stefano Cretti

telegram: @StefanoCretti

Github: https://github.com/StefanoCretti/StatisticalLearning.git

May 24, 2022

Contents

Part I

Introduction

Chapter 1

General course information

1.1 Textbooks

- James et al (2021), Introduction to statistical learning in R, 2nd edition. (Book that is used as guidline for the course, but further concepts will be added during the lectures)
- Hastie et al (2001), Elements of statistical learning. (More advanced book for those who want to study the subject more in depth)

1.2 Assessment

- Three homework tasks during the course (Uploaded on moodle, two weeks of time for each, more practical and mainly focused on applying methods to some data. If done well they will add 2 points to the written exam score)
- Final written exam (More theoretical but still connected to the practical part, for instance by commenting on analysis output)

1.3 Topics

- Linear regression (Gauss, 1800) (Assumed to be already known from Statistical Learning 1)
- Linear discriminant analysis, LDA (Fisher, 1936) (Later extended to quadratic discriminant analysis, QDA)
- Logistic regression (1940s)
- Generalized linear models (Nelder and Wedderburn, 1972)
- Classification and regression trees (Breiman and Freidman, 1980s) (First introduction of computer intensive methods)
- Machine learning (1990s): support vector machines, neural networks/deep learning, unsupervised learning (clustering, PCA)
- Individual methods: theory, details, implementation ...

- General concept: model selection, inference, prediction ...

Part II

Statistical learning

Chapter 2

What is statistical learning?

2.1 Definition of statistical learning

In general, a statistical learning problem can be formalized as follows:

- Y : response/dependent/outcome variable
- $\underline{X} = (X_1, \ldots, X_p)$: vector of predictors/features/independent variables/covariates

We assume that there is a relationship between Y and $\underline{X}$, which can be written as:

$$Y = f(\underline{X}) + \varepsilon$$

Where:

- $f(\underline{X})$ is the deterministic (but unknown) function of the vector $\underline{X} = (X_1, \ldots, X_p)$
- ε is the error (stochastic part), for which we assume the following properties:
  - $E[\varepsilon] = 0$ (Its expected value is zero)
  - $\varepsilon \perp \underline{X}$ (It is independent from $\underline{X}$)

the ϵ value represents the stochastic error, which is the estimated error in a model that arises from the exclusion of an important explanatory variable or due to incorrect specification. Therefore, the expression **statistical learning** encompasses different methods to estimate $f(\underline{X})$.

2.2 Why estimate f?

There are two main reasons to estimate f, those two being **prediction** and **inference**.

2.2.1 Prediction

Predict Y when we only have observations about $\underline{X}$. Since $E[\varepsilon] = 0$, we usually take:

$$\hat{Y} = \hat{f}(\underline{X})$$

With $\hat{f}$ being our estimate of f.

If this is the only reason to estimate f, then $\hat{f}$ can be a ***black-box*** method (deep learning). The accuracy of $\hat{Y}$ as a predictor of Y can be described calculating the expected **MSE**, the test Mean Squared Error.

$$
\begin{aligned}
E[(Y - \hat{f}(\underline{X}))^2 \,|\, \underline{X} = \underline{x}] = & && \text{where } \hat{f} \text{ is a fixed known function} \\
= E[(f(\underline{X}) + \varepsilon - \hat{f}(\underline{X}))^2] & && \text{since } Y = f(\underline{X}) + \varepsilon \\
= E[(f(\underline{X}) - \hat{f}(\underline{X})) + \varepsilon)^2] & && \text{rearranging} \\
= E[(f(\underline{X}) - \hat{f}(\underline{X}))^2 + \varepsilon^2 + 2\varepsilon(f(\underline{X}) - \hat{f}(\underline{X}))] & && \text{solving the square} \\
= E[(f(\underline{X}) - \hat{f}(\underline{X}))^2] + E[\varepsilon^2] + 2E[\varepsilon(f(\underline{X}) - \hat{f}(\underline{X}))] & && \text{separating the expectations}
\end{aligned}
$$

Furthermore, since we know that:

$$
\begin{aligned}
Var(\varepsilon) &= E[(\varepsilon - E(\varepsilon))^2] && \text{formal definition of variance} \\
&= E[\epsilon^2 + (E(\epsilon))^2 - 2E(\epsilon)\epsilon] && \text{solving the square} \\
&= E[\varepsilon^2] - (E[\varepsilon])^2 && \text{definition generally used during calculation} \\
&= E[\varepsilon^2] && \text{since } E[\varepsilon] = 0
\end{aligned}
$$

Thus we get:

$$
\begin{aligned}
E[(Y - \hat{f}(\underline{X}))^2 \,|\, \underline{X} = \underline{x}] = & && \\
= E[(f(\underline{X}) - \hat{f}(\underline{X}))^2] + Var(\varepsilon) + 2E[\varepsilon(f(\underline{X}) - \hat{f}(\underline{X}))] & && \text{substituting } E[\varepsilon^2] = Var(\varepsilon) \\
= (f(\underline{X}) - \hat{f}(\underline{X}))^2 + Var(\varepsilon) + 2(f(\underline{X}) - \hat{f}(\underline{X}))E[\varepsilon] & && \text{since } f(\underline{X}) - \hat{f}(\underline{X}) \text{ is a constant} \\
= (f(\underline{X}) - \hat{f}(\underline{X}))^2 + Var(\varepsilon) & && \text{since } E[\varepsilon] = 0
\end{aligned}
$$

With:

- $(f(\underline{X}) - \hat{f}(\underline{X}))^2$ being the **reducible error**. The model choice can increase or reduce this value, hence it is mostly controllable.
- $Var(\varepsilon)$ being the **irreducible error**. This value depends on the innate randomness present in the data, hence you can only try and minimize it by deciding which variables to use in your prediction (but it will never be zero otherwise you would have a deterministic situation).

2.2.2 Inference

Inference is used when you want to understand the relation between Y and $\underline{X}$ (and not just be able to make predictions). Namely, inference answers questions such as:

- *Which predictors/factors are most associated with the response?*
- *What is the relationship between Y and X_j?*

2.3 How to extimate f?

Given some **training data** $(\underline{x}_i, y_i)$, $i = 1, \ldots, n$, where $\underline{x}_i = (x_{i1}, \ldots, x_{ip})^t$ is the vector of observations of unit i while y_i is the response for unit i, broadly speaking there are two types of methods to estimate f: **parametric methods** and **non-parametric methods**.

2.3.1 Parametric methods

In order to use parametric methods, we make an assumption about the functional form of $f(\underline{X})$, that assumption being that the form of the function depends on some *parameters* (which we can estimate). An example of parametric method is **linear regression**, which implies that $f(\underline{X})$ is in the form:

$$f(\underline{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Where the great Xs represent the predictors. Given this assumption, statistical learning becomes **fitting** (or training) the model on the data, which means estimating the parameters $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ such that:

$$\hat{f}(\underline{X}) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$$

The main disadvantage of these methods is that they may be **too restrictive**, as they would depend on a limited number of predictors.

Notice that linear models are linear in the parameters, not in the predictors, hence:

- $f(X_1) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_1^3 + \varepsilon$ (polynomial regression) is a linear model.
- $f(X_1) = \beta_0 X_1^{\beta_1}$ is not a linear model.

2.3.2 Non-parametric methods

Non-parametric methods do not make any explicit assumption on the function form of $f(\underline{X})$. These methods want to estimate f by getting as close as possible to the data, without being too *rough or wiggly* (basically **overfitting**).

2.3.3 Parametric vs non-parametric methods

Despite parametric methods being more restrictive than non-parametric ones, we might still choose to adopt them for the sake of interpretability and generalizability outside of the training data.

2.4 Bias-variance trade-off

Assume you have some data pairs in the form (x_i, y_i), $i = 1, \ldots, n$; you can then define the estimate function $\hat{f}(x)$ as a linear model with up to $n - 1$ parameters (more parameters give the same result

as $n-1$ parameters) and an intercept value. You could thus use, for instance, the models:

1 parameter	$f(x) = \beta_0 + \beta_1 x + \varepsilon$
2 parameters	$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$
$\vdots$	$\vdots$
$n-1$ parameters	$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{n-1} x^{n-1} + \varepsilon$

You could choose the model with the highest number of parameters; this model would explain all the variance of the training data ($R^2 = 1$) yet it would be very complex and perform badly with new data points. On the other hand a model with a lower number of parameters would explain less of the variance of the training data, yet it could perform better with new data points.

Keeping in mind that Y is random and that $\hat{f}$ is a random variable estimated from the data, when using the general formula to determine how well a model performs at generic $\underline{X}$, we notice:

$$
\begin{aligned}
&E[(Y - \hat{f}(\underline{X}))^2 \,|\, \underline{X} = \underline{x}\,] = \\
&\quad = E[(f(\underline{X}) + \varepsilon - \hat{f}(\underline{X}))^2] && \text{since } Y = f(\underline{X}) + \varepsilon \\
&\quad = E[(f(\underline{X}) + \varepsilon - \hat{f}(\underline{X}) + E[\hat{f}(\underline{X})] - E[\hat{f}(\underline{X})])^2] && \text{since } E[\hat{f}(\underline{X})] - E[\hat{f}(\underline{X})] = 0 \\
&\quad = E[((f(\underline{X}) - E[\hat{f}(\underline{X})]) + \varepsilon + (E[\hat{f}(\underline{X})] - \hat{f}(\underline{X})))^2] && \text{grouping, square of three elements} \\
&\quad = E[(f(\underline{X}) - E[\hat{f}(\underline{X})])^2] + E[\varepsilon^2] + E[(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))^2] + \\
&\qquad + 2E[(f(\underline{X}) - E[\hat{f}(\underline{X})])\,\varepsilon] + \\
&\qquad + 2E[(f(\underline{X}) - E[\hat{f}(\underline{X})])(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))] + && \text{solving the square and} \\
&\qquad + 2E[(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))\,\varepsilon] && \text{dividing the expectations}
\end{aligned}
$$

But notice that:

$$
\begin{aligned}
E[(f(\underline{X}) - E[\hat{f}(\underline{X})])\,\varepsilon] &= E[\varepsilon](f(\underline{X}) - E[\hat{f}(\underline{X})]) && \text{since } f(\underline{X}) - E[\hat{f}(\underline{X})] \text{ is constant} \\
&= 0 && \text{since } E[\varepsilon] = 0
\end{aligned}
$$

$$
\begin{aligned}
&E[(f(\underline{X}) - E[\hat{f}(\underline{X})])(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))] = \\
&\qquad = (f(\underline{X}) - E[\hat{f}(\underline{X})])E[E[\hat{f}(\underline{X})] - \hat{f}(\underline{X})] && \text{since } f(\underline{X}) - E[\hat{f}(\underline{X})] \text{ is constant} \\
&\qquad = (f(\underline{X}) - E[\hat{f}(\underline{X})])E[\hat{f}(\underline{X}) - \hat{f}(\underline{X})] && \text{since } \hat{f}(\underline{X}) \text{ is a constant} \\
&\qquad = (f(\underline{X}) - E[\hat{f}(\underline{X})])E[0] \\
&\qquad = 0
\end{aligned}
$$

$$
\begin{aligned}
E[(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))\,\varepsilon] &= E[\varepsilon]E[E[\hat{f}(\underline{X})] - \hat{f}(\underline{X})] && \text{since } \varepsilon \perp \underline{X} \implies E[\varepsilon \underline{X}] = E[\varepsilon]E[\underline{X}] \\
&= 0 && \text{since } E[\varepsilon] = 0
\end{aligned}
$$

Therefore we can simplify as:

$$\begin{aligned} E[(Y - \hat{f}(\underline{X}))^2 \,|\, \underline{X} = \underline{x}] &= E[(f(\underline{X}) - E[\hat{f}(\underline{X})])^2] \quad &+E[\varepsilon^2] \quad &+E[(E[\hat{f}(\underline{X})] - \hat{f}(\underline{X}))^2] \\ &= f(\underline{X}) - E[\hat{f}(\underline{X})])^2 \quad &+Var(\varepsilon) \quad &+Var(\hat{f}(\underline{X})) \end{aligned}$$

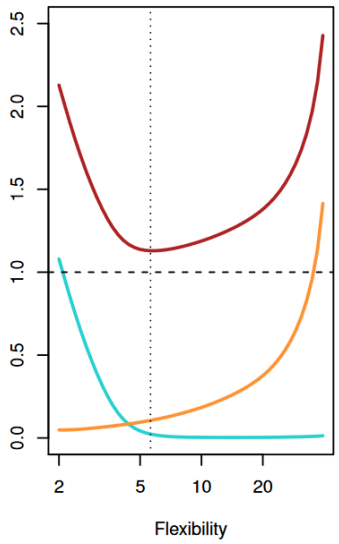
Where:

- $f(\underline{X}) - E[\hat{f}(\underline{X})])^2$ is the **bias**
- $Var(\varepsilon)$ is the **irreducible error**
- $Var(\hat{f}(\underline{X}))$ is the **variance** of the model

Bias is the amount that a model's prediction differs from the target value, compared to the training data. Generally bias is consequently results from simplifying the assumptions used in a model. If an estimated model performs well on multiple data sets, the estimated model has low bias. *High bias means that an estimated model is far from the real model.*

Variance refers instead to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. Ideally, the $\hat{f}$ calculated through different datasets should be not significantly different, the model should be fitted equivalently using a different training dataset. The more flexible models, the more they are influenced by a changes of the training dataset.

Figure 2.1: As the flexibility of a model increases, the Bias decreases, and the varaiance of the model increases



In order to minimize the expected error, we need to achieve low bias and low variance. In practice, one needs to find a good **trade-off** between bias and variance, since reducing one often involves increasing the other. In general:

- A **simpe model** has high bias (it is far from the real model) and low variance (when fitting using different training data you get similar estimated parameters).
- A **complex model** has low bias and high variance, as the complexity of the model has the aim of making it as near as possible to the real f.

Both in parametric and non-parametric methods, you generally have at least one **tuning parameter**, which is a parameter that can be tweaked (for instance the degree of the polinomial) in order to choose the balance between bias and variance.

Chapter 3

Statistical decision theory

3.1 Definition of statistical decision theory

The **statistical decision theory** is a set of quantitative methods for reaching optimal decisions for well posed problems. Statistical decision theory applies to supervised learning, while it does not apply to unsupervised learning. For most of the course we will focus on supervised learning (decision theory statistics from Britannica).

3.2 Supervised learning

Supervised learning is a set of methods whose objective is, given the observations $(\underline{x}_i, y_i)$ with $i = 1, \ldots, n$, to **find a rule $\hat{f}$ that allows us to predict Y from $\underline{X}$**, meaning that $\hat{Y} = \hat{f}(\underline{X})$. $\hat{f}(\underline{X})$ is thus an approximation of the true rule $f(\underline{X})$. Finding $\hat{f}$ corresponds to training (or fitting) a model using labelled data pairs (both predictors and response values are known). In this case, the term *fitting* refers to adjusting the parameters in the model to improve the accuracy. *Model fitting*, insted, is the measure of how well a machine learning model generalizes data similar to that with which it was trained. A good model fit refers to a model that accurately approximates the output when it is provided with unseen inputs.

Broadly, there are two main types of methods of supervised learning:

- **Regression methods**, in which the response Y is quantitative (numerical). In this case $\hat{f}$ is called ***regression function***.
- **Classification methods**, in which the response Y is qualitative (categorical). In this case $\hat{f}$ is called ***classifier***.

The predictors ($\underline{X}$) can take any form, numerical or categorical. In general there are no assumptions on the form of the predictors (but not always).

To evaluate a model you use **loss functions**, which are functions used to penalize the differences between Y and $\hat{f}(\underline{X})$. Many loss functions can be used; the choice of a specific loss function determines which function is considered to be the true rule $f(\underline{X})$ (since this moment I will refer $\hat{f}(x)$ as $f(x)$).

3.3 Regression setting

In a regression setting, the loss function which is generally used is the **squared error loss function** (also referred as the risk of estimating Y using $\hat{f}(\underline{X})$), which is defined as:

$$L(Y, f(\underline{X})) = (Y - f(\underline{X}))^2$$

When using this loss function, the criterion for choosing the model becomes finding f such that it minimizes the **expected prediction error** (EPE), which is the expected value of the squared error loss function:

$$\begin{aligned} \text{EPE}(f) &= E_{Y,\underline{X}}[(Y - f(\underline{X}))^2] \\ &= \iint (y - f(\underline{x}))^2 g_{Y,\underline{X}}(y, \underline{x})\, dy\, d\underline{x} \qquad \textbf{if } Y \textbf{ is continuous} \end{aligned}$$

This error is an expectation since you usually do not know the distribution of Y and $\underline{X}$. If Y is continuous you can rewrite this expectation as the double integral times the joint density function $g_{Y,\underline{X}}(y, \underline{x})$.

Theorem 1 (Minimum of the expected prediction error)**.** *$EPE(f) = E_{Y,\underline{X}}[(Y - f(\underline{X}))^2]$ has a **minimum** when $f(\underline{x}) = E[Y|\underline{X} = \underline{x}]$. $f(\underline{x})$ is called regression function.*

Proof. (Sketch) By definition we know that

$$\text{EPE}(f) = E_{Y,\underline{X}}[(Y - f(\underline{X}))^2]$$

Then, because of the law of *iterated expectations* we get

$$E_{Y,\underline{X}}[(Y - f(\underline{X}))^2] = E_{\underline{X}}[E_{Y|\underline{X}}[(Y - f(\underline{X}))^2]|\underline{X}]$$

If we then consider a specific value of $\underline{X}$, meaning $\underline{X} = \underline{x}$, the inner expectation becomes

$$E[(Y - f(\underline{x}))^2|\underline{X} = \underline{x}]$$

But since $\underline{x}$ is fixed, $f(\underline{x})$ is a constant, hence this expectation can be written as a function in some parameter a

$$g(a) = E_Y[(Y - a)^2]$$

Then we want to find the (constant) value of a that minimizes $g(a)$

$$\begin{aligned} g(a) &= E_Y[(Y - a)^2] \\ &= E[Y^2 - 2aY + a^2] && \text{compute the square} \\ &= E[Y^2] - 2aE[Y] + E[a^2] && \text{separate expectations} \\ &= E[Y^2] - 2aE[Y] + a^2 && \text{pull out } a^2 \text{ since constant} \end{aligned}$$

Then, setting the derivative to zero we get

$$\frac{dg}{da} = -2E[Y] + 2a \overset{!}{=} 0 \implies \hat{a} = E[Y]$$

Notice that if you plug $\hat{a}$ into $g(a) = E_Y[(Y-a)^2]$ you get

$$g(\hat{a}) = E_Y[(Y - E[Y])^2] = Var(Y)$$

Since $a = E[Y]$, the function that minimizes the $g(a)$ function and consequently the EPE is

$$f(\underline{x}) = E[Y|\underline{X} = \underline{x}]$$

□

In a regression setting you could use other loss functions, such as the *absolute error loss* function

$$L(Y|f(\underline{X})) = |Y - f(\underline{X})|$$

which is used in **least absolute deviation** (LAD) regression. In this case $f(\underline{x})$ **is the median of** Y **given** $\underline{x}$, therefore you have a model which is more robust to outliers.

We will choose the squared error loss and this motivates how the methods are developed. For example:

- **Linear regression** (parametric regression) can be expressed as

  $Y|\underline{X} = \underline{x} \sim N(\underline{x}^t\underline{\beta}, \sigma^2)$ Meaning that $\underline{x}$ is composed of

  or focusing more on the conditional mean

  $$E[Y|\underline{X} = \underline{x}] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

- **K-nearest neighbours regression**: one of the most used non-parametric regression

3.3.1 K-nearest neighbours regression (KNN)

In the scope of regression, K-nearest neighbours method is defined as

$$\hat{f}(\underline{x}) = \text{Average}(y_i|\underline{x_i} \in N_K(\underline{x})) = \frac{1}{K}\sum_{x_i \in N_{\underline{x}}} y_i \tag{3.1}$$

Where:

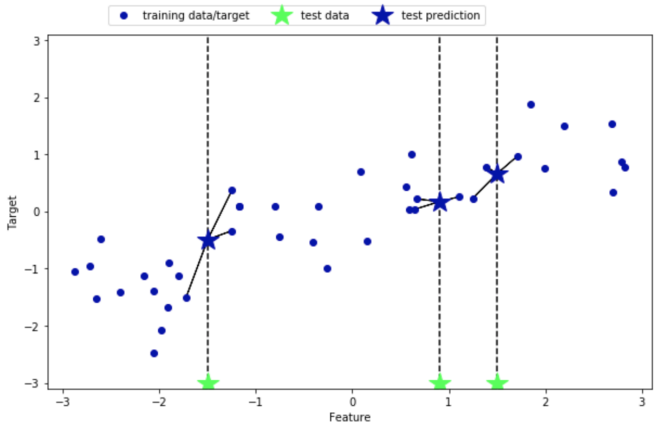
- Average is the sample mean
- $N_K(\underline{x})$ is the neighbourhood of K points closest to $\underline{x}$
- K is an integer

This method makes two approximations:

- It uses sample mean to approximate population mean
- It uses a neighbourhood of $\underline{x}$ rather than only $\underline{x}$

This method works well if you have a high amount of data and the number of parameters (p) is small. This simple method is rarely used, but it has inspired the development of more sophisticated kernel methods.

Figure 3.1: The training data are shown in blue dots, test data and predictions of test data are star-shaped. The predicted value on test data is given making the average of the neighbours' values



3.4 Classification setting

In a classification setting the response Y *is categorical*, with K categories. A **classifier** is deciding which class to assign to a new observation $\underline{x}$. In this case $\hat{Y}$ is the predicted class.

An example of loss function that can be used in classificiation setting is the 0-1 **loss function** which penalizes classifying a datapoint in the wrong class. Assume for instance the binary case, meaning $K = 2$ (but you could generalize for any number of classes); in that case the 0-1 loss function can be represented as:

True value, Prediction	**0**	**1**
0	0	1
1	1	0

Table 3.1: This is the table representing the possible values of the 0-1 loss function: $L(Y, \hat{Y}(\underline{X}))$. Y is the true class, while instead $\hat{Y}(\underline{X})$ is the predicted class.

Notice that this function penalizes the same way all the misclassified observations (it assigns them the value 1).

Using the 0-1 loss function, the criterion for choosing the model becomes finding the classifying function $\hat{Y}$, dependent on $\underline{X}$, that **minimizes the expected 0-1 loss function** when applied to every possible value $\underline{x}$. In practice, we want to find the function $\hat{Y}$ so that the expected value $E[L(Y, \hat{Y}(\underline{X}))]$ is the minor.

To find $\hat{Y}$, let us consider just one $\underline{x}$, therefore the problem becomes minimizing

$$E[L(Y, \hat{Y}(\underline{x}))] = \sum_{k=1}^{k} L(k, \hat{Y}(\underline{x}))\ p(k|\underline{x})$$

[This equation means that $E[L(Y, \hat{Y}(\underline{x}))]$, is equal to the sum of all the possible values of the loss function (calculated for each class with respect to the predicted class) multiplied by the probability of that class given $\underline{x}$]. Then, considering the binary case (with two possible classes 0 and 1), if the classifier predicts $\underline{x}$ to belong to class 0 ($\hat{y} = 0$), we have

$$\begin{aligned} E[L(Y,0)] &= L(0,0)p(0|\underline{x}) + L(1,0)p(1|\underline{x}) \\ &= 0 \cdot p(0|\underline{x}) + 1 \cdot p(1|\underline{x}) \\ &= p(1|\underline{x}) \end{aligned}$$

Logically, $E[L(Y,0)]$ will be the loss value (1) multiplied by the probability of $\underline{x}$ belonging to class 1, Y = 1. On the other hand, if the classifier predicts $\underline{x}$ to belong to the class 1, $p(1|\underline{x})$, we have

$$\begin{aligned} E[L(Y,1)] &= L(0,1)p(0|\underline{x}) + L(1,1)p(1|\underline{x}) \\ &= 1 \cdot p(0|\underline{x}) + 0 \cdot p(1|\underline{x}) \\ &= p(0|\underline{x}) \end{aligned}$$

Predicting $\underline{x}$ to the class that minimizes the expected 1-0 loss results in classifying $\underline{x}$ to class 1 if $p(1|\underline{x}) > p(0|\underline{x})$. Since $p(0|\underline{x}) = 1 - p(1|\underline{x})$, we can write

$$p(1|\underline{x}) > 1 - p(1|\underline{x}) \implies 2p(1|\underline{x}) > 1 \implies p(1|\underline{x}) > 0.5$$

Therefore, in the binary case, $\underline{x}$ is predicted to belong to the class with more than 0.5 probability (which is intuitive).

In general (k classes), the 0-1 loss is minimized by the **bayes classifier**, which states to assign $\underline{x}$ to class j where

$$j = \underset{i \in classes}{\operatorname{argmax}}\, p(Y = j|\underline{X} = \underline{x})$$

(Which means assigning $\underline{x}$ to the class with the highest probability). This approach tells us how to set the problem, but it does not give any information on the probabilities, which have to be estimated by the single method.

To rewrite the binary case in another way:

$$\text{assign } \underline{x} \text{ to class } \begin{cases} 1 \text{ if } p(1|\underline{x}) > 0.5 \\ 0 \text{ if } p(1|\underline{x}) < 0.5 \end{cases}$$

The line formed by the points with probability $p(1|\underline{x}) = 0.5$ is called **bayes decision boundary** or **decision surface**.

The **bayes error rate** (BER) is the probability of committing an error when classifying observations. It is defined as:

$$\text{BER } = 1 - E_{\underline{X}} \left(\max_j p(j|\underline{x}) \right)$$

(Since the class with the highest probability is the one considered by us to classify observations, the probability of getting an error is equal to the total probability subtracted with the probability used to assign the class).

Figure 3.2: Bayes decision boundary in green. In the simplest case, when we have two classes with equivalent probability, $p(1|\underline{x}) = p(0|\underline{x} = 0.5)$

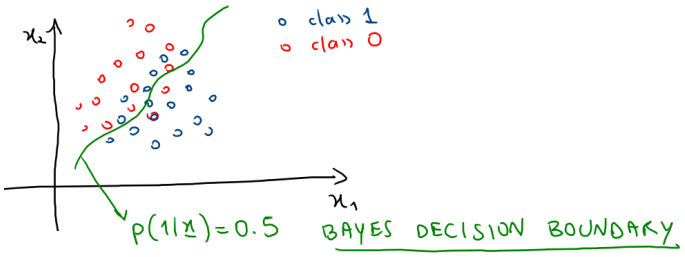
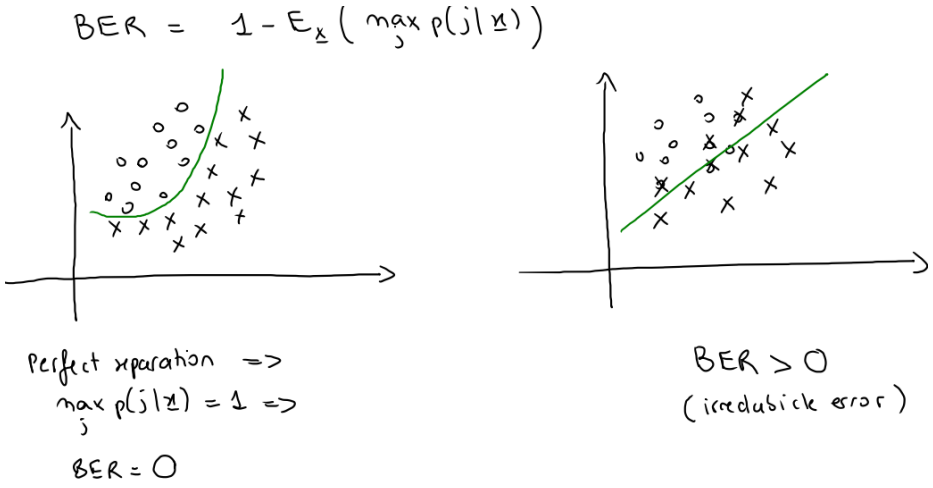


Figure 3.3: The figure on the left represents the best possible scenario, as the BES value is 0, and the decision surface separate completely the elements belonging to the two classes. The graph on the left instead shows an intermadiate situation, where $BER \neq 0$



We talk about **perfect separation** of the classes (figure 3.3) when

$$\max_j p(j|\underline{x}) = 1 \implies \text{BER} = 0$$

In this case it is possible to classify any $\underline{x}$ without error. Generally, **BER** > 0, therefore you have some irreducible error due to a partial overlap of the classes.

It is possible to use a more generic loss function, called **misclassification loss function**, which can be represented (if $K = 2$) as:

Table 3.2: This is the table representing the possible values of $L(Y, \hat{Y}(\underline{X}))$. c_0 and c_1 represent two different cost values

True value, Predictions	**0**	**1**
0	0	c_0
1	c_1	0

This loss function is analogous to the 0-1 loss function, but it assigns **different weights to different errors**, therefore the misclassifications have different costs:

- c_0 is the misclassification cost of misclassifying a class 0 to 1
- c_1 is the misclassification cost of misclassifying a class 1 to 0

Just like the 0-1 loss function, misclassification loss function can be generalized for K classes. This loss function is generally better than the 0-1 loss function since***in real situations it is likely for misclassifications to have different relevance*** (credit risk evaluation, medical context and others). Moreover, this loss function allows you to adjust for unbalanced classes. Consider for example a rare disease; consider the 1 value to be indicative of the ill state, and 0 of the healthy state. We would obtain in all the cases

$$\begin{aligned} E[L(Y,0)] &= L(1,0)p(1|\underline{x}) && \text{prediction for the 0 (healthy) class} \\ <<< \; E[L(Y,1)] &= L(0,1)p(0|\underline{x}) && \dots \text{ for the 1 (ill) class} \end{aligned}$$

since $p(1|\underline{x})$ is really low, and consequently, using a 0-1 loss function results in a model which always classifies patients as healty. The model would be almost always correct, but it does not perform well when trying to determine which patients are affected by the disease. When using misclassification loss function, you can instead increase the weight of the misclassification "classify a patient as healthy when they are not", which may lead to worse results in terms of identifying healthy patients, but way better results in terms of identifying diseased patients.

Repeating the same steps used for 0-1 loss function, we can determine the threshold for the misclassification function (for $k = 2$ but it can be generalized)

$$E[L(Y,0)] = c_1 p(1|\underline{x}) \text{ and } E[L(Y,1)] = c_0 p(0|\underline{x})$$

Therefore we assign $\underline{x}$ to class 1 if

$$c_1 p(1|\underline{x}) > c_0 p(0|\underline{x})$$

But since $p(0|\underline{x}) = 1 - p(1|\underline{x})$, we can instead write

$$\begin{aligned} c_1 p(1|\underline{x}) &> c_0 - c_0 p(1|\underline{x})) \\ p(1|\underline{x}) &> \frac{c_0}{c_0 + c_1} \end{aligned}$$

which is the **classification threshold**.

3.5 Model accuracy

In practice, we would use our chosen method to estimate $f(\underline{x}) = E[Y|\underline{X} = \underline{x}]$ (regression) or $p(1|\underline{x})$ (classification) from training data $(\underline{x}_i, y_i),\ i = 1, \ldots, n$. The **accuracy** of the model is typically measured on some **test data** $(\underline{x}_i^{(t)}, y_i^{(t)}),\ i = 1, \ldots, m$.

Therefore, a regression that uses MSE has for accuracy

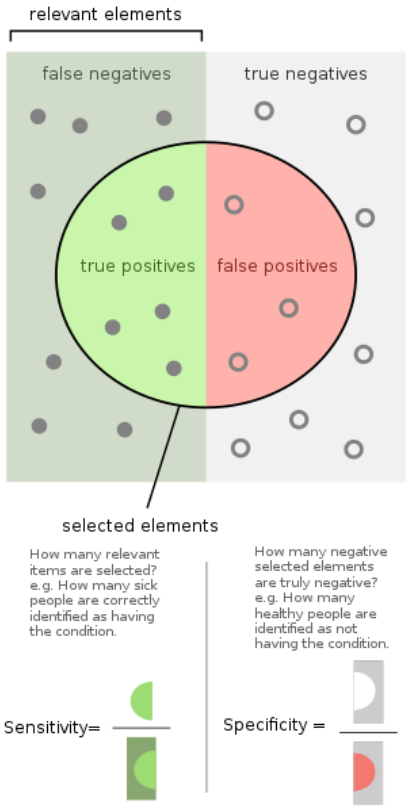
$$\text{MSE } = \frac{1}{m} \sum_{i=1}^{m} \left(y_i^{(t)} - \hat{f}\left(\underline{x}_i^{(t)}\right)\right)^2$$

True value, Predictions	**0**	**1**
0	TN	FP
1	FN	TP

Table 3.3: Confusion matrix

Meanwhile, for a classification, you create a **confusion matrix** (table 3.3), estimate the probabilities via some method, substitute the values into the following formulas to obtain

Figure 3.4: graphical explanation for dummy guys



sensitivity and specificity (figure 3.4).

$$\textbf{True positive rate (TPR)} \quad = \frac{\text{TP}}{\text{TP} \ + \ \text{FN}} = \ \textbf{sensitivity}$$

$$\textbf{False positive rate (FPR)} \quad = \frac{\text{FP}}{\text{FN} \ + \ \text{FP}} = 1 - \frac{\text{TN}}{\text{TN} \ + \ \text{FP}} = 1 - \ \textbf{specificity}$$

$$\textbf{Error rate} \quad = \frac{\text{FP} \ + \ \text{FN}}{m}$$

Notice that this last formula only works when using a 1-0 loss function; if you are using a misclassification loss function, the formula is similar but weighted:

$$\text{Misclassification cost} = \frac{c_0 \ \text{FP} + c_1 \ \text{FN}}{m}$$

Since defining the exact costs of the misclassifications might prove difficult, a possible approach is using the **Receiver Operating Characteristic (ROC) curve**. Basically you plot all the pairs (FPR, TPR) you obtain by varying the classification threshold from 0 to 1; you will then obtain

a curve from which you can decide which value to use as a threshold (the one you see fit for your needs of specificity and sensibility).

Figure 3.5: Image taken from Wikipedia. Receiver Operating Characteristic (ROC) curve.

Different models have different ROC curves. The best possible curve would touch the top left corner (figure 3.5), meaning that you never misclassify any observation and therefore the model is deterministic. The worst possibile curve corresponds to the diagonal (from bottom left to top right corner), meaning that you have a $1/2$ chance of misclassifying the observation (the same as tossing a coin to decide). The **Area Under the Curve (AUC)** is the area between the curve and the main diagonal. Its value goes from 0 in the worst case to 0.5 in the best one. For these reasons, you want a curve that is as high as possible (and thus has the biggest possible AUC). When choosing between two models, if the ROC curve of one of the models is always above the ROC of the other model, the former one is the strictly better one; if the two ROC curves intersect, you choose the model whose curve is the highest in the region you are most interested in (based on sensibility and specificity).

Chapter 4

Statistical decision theory II

Just like in the regression setting, there are parametric and non-parametric methods in a classification setting too, these methods being:

- **Logistic regression** (parametric classification)
- **K nearest neighbour** (non-parametric classification)

4.1 K nearest neighbour for classification

K-nearest neighbour is a non-parametric method for classification, meaning it allows to estimate the probability of a testing observation belonging to each of the classes. Formally this can be written as:

$$P(Y = j|\underline{X} = \underline{x}_0), \quad \text{for } j = 1, \ldots, C$$

Where:

- $\underline{x}_0$ is the test observation
- j is a specific class
- C is the number of classes

K is a positive integer and represents the number of training data points closest to the observation data point to consider when classifying. K is called **tuning parameter**, meaning that it can be tweaked in order to change how the model works; notice that the model is still non-parametric since K does not depend on the training data points.

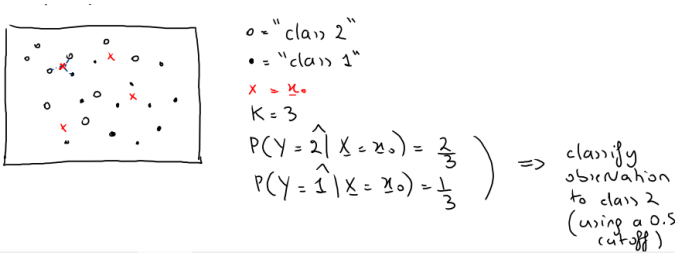
In order to use this method you have to:

1. **Identify the K training observations** that are closest (according to some distance, typically we consider the Eucledian distance) to $\underline{x}_0$. These observations are then indexed with $\underline{N}_0$.
2. **Compute the probability of the observation to belong to each of the classes** using the following formula:

$$\hat{P(Y = j|\underline{X} = \underline{x}_0)} = \frac{1}{k} \sum_{i \in \underline{N}_0} I(\hat{y_i} = j), \quad \text{where } j = 1, \ldots, C$$

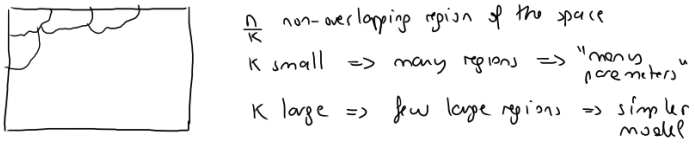
Put into words, the equation above means: compute the probability that, given a test observation $\underline{x}_0$, it belongs to a specific class j by dividing the number of neighbours belonging to that class by the number of neighbours (K). The I is an indicator variable that equals 1 if $y_i \neq \hat{y}_i$, while instead it is equal to 0 if $y_i = \hat{y}_i$. The indicator is used for each $i \in \underline{N_0}$ observation. Repeat for each class.

Figure 4.1: Classification process with K-neighbour method



4.1.1 Bias-Variance trade off in K nearest neighbour classification

Figure 4.2: Tuning the K parameter



The tuning parameter K relates to the complexity of the model and it is the determinant to balance the bias and the variance of the model. Usually the value of K is chosen using some training data. Intuitively (NOT formally), the model space could be considered as an area of dimension n which is divided in regions containing k training points, therefore you have $\frac{n}{k}$ regions, hence:

- if **K is small**, then there are many small regions in the space and the model is complex. For this reason, if we change $\underline{x}_0$ with another value, the decision surface position changes quite a lot; if K is too small, the model could be overfitting and thus perform poorly with new test data points (low bias, high variance).
- if **K is large**, then there are few big regions in the space and the model is simple. For this reason, if we change $\underline{x}_0$ with another value, the decision surface position will barely change; if K is too big, the model will give similar results for many different values of $\underline{x}_0$ (high bias, low variance).

The graph shown in figure 4.4 represents the behaviour of different error rates (on test data, the constant error, and on training data) depending on the complexity of the model. As it can be seen ...

An increase in model complexity is always associated with a lowering of the error rate on the training data, while it is not generally associated with a decrease of the error rate performed on the

Figure 4.3: Complex and smooth decision surfaces, that respectively belong to complex models and simpler models

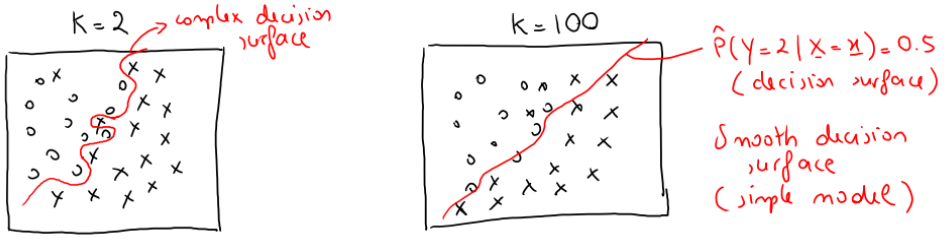
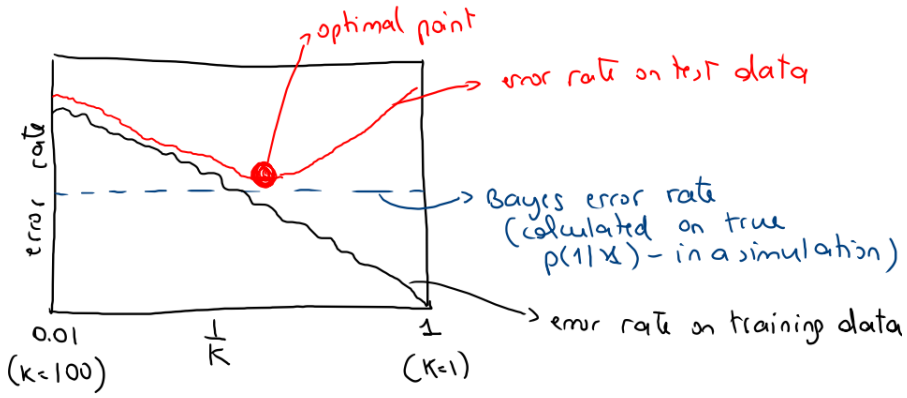


Figure 4.4: Error rate on test data (red), error rate on training data (black), Bayes|constant error rate



test data. The latter in fact decreases until a certain moment, when it starts to increase making a U-shaped curvature. The best case scenario would be the model where both the error rates are at their minimum.

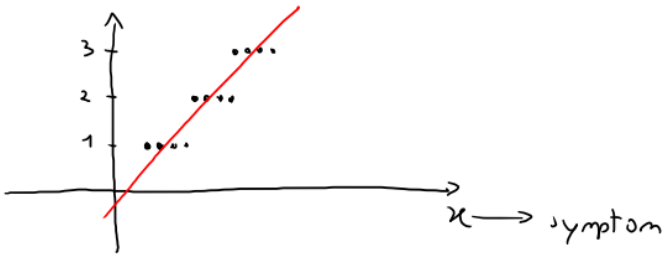
4.2 Logistic regression

4.2.1 Why we cannot use linear regression in classification setting

Example 1: We want to predict the medical condition of a patient in the emergency room based on their symptoms. Assume that Y takes only 3 possible values:

$$Y = \begin{cases} 1 \text{ if stroke} \\ 2 \text{ if overdose} \\ 3 \text{ if seizures} \end{cases}$$

Figure 4.5: Image representing a possible linear regression model for distinguishing classes



A linear regression may seem to work, but you have some **problems**:

- If the order of the data points changes, the model changes.
- We are arbitrarely assigning an order to not necessarely ordered classes.
- We are imposing an arbitrary distance between the categories.

Example 2: Assume that Y from the previous example only takes 2 values:

$$Y = \begin{cases} 0 \text{ if stroke} \\ 1 \text{ if overdose} \end{cases}$$

In this case Y is a random variable described by a Bernoulli distribution with some probability p of taking the value 1, only two possible values are admitted; therefore

$$Y = \begin{cases} 1 & p \\ 0 & 1-p \end{cases}$$

The Bernoulli random variable, that can assume either 0 or 1 value, has this particular expected value:

$$E[Y] = 0 \cdot (1-p) + 1 \cdot p = p \qquad \text{where } p \text{ represents the probability of getting a 1}$$

Given these premises, you can write:

$$E[Y|\underline{X} = \underline{x}] = p(1|\underline{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Or considering just the Bernoulli case:

$$E[Y|\hat{\underline{X}} = \underline{x}] = \hat{\beta}_0 + \hat{\beta}_1 x_1$$

But notice that:

- $p(1|\underline{x}) \in (0,1)$, so the probability should span between 0 to 1
- $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \in (-\infty, +\infty)$, yet using linear regression the probability spans over the entirety of $\mathbb{R}$.

Any time a straight line is fit to a binary response that is coded as 0 or 1, in principle we can always predict $p(1|\underline{x}) < 0$ for some values of X and $p(1|\underline{x})) > 1$ for others. Therefore we must find a way to normalize the regression in the range $(0,1)$.

4.2.2 Characteristics of the logistic regression

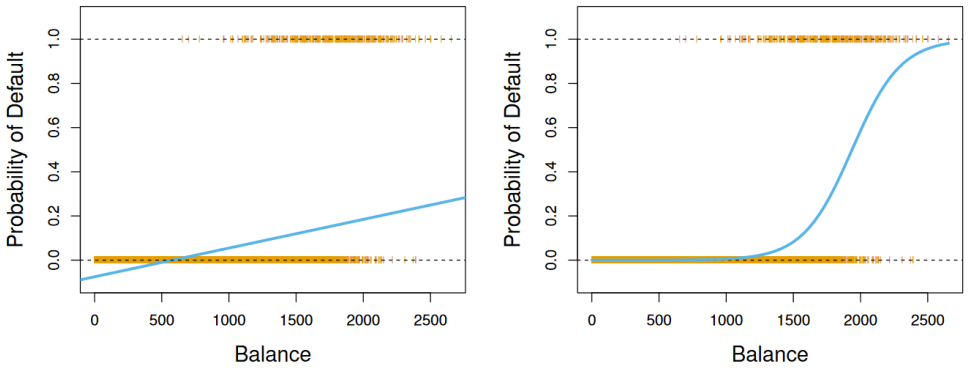
We have to model $p(1|\underline{x})$ using a function that gives outputs between 0 and 1 for each value of X. **Logistic regression** solves the problem by utilizing a function of the probability $p(1|\underline{x})$, namely g, such that

$$g(p(1|\underline{x})) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \underline{\beta}^t \underline{x}$$

Logistic regression uses the **logistic function** (also named sigmoid function or activation function), which is defined as

$$p(1|\underline{x}) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

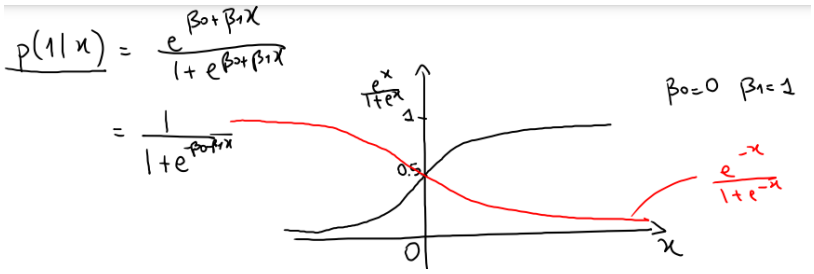
Figure 4.6: Linear regression (left) **vs** Logistic regression (right) for classification problems. Notice on the left the fact that values under 0 for certain values of $\underline{x}$ are possible.



and it has the following properties (figure 4.6):

- Its codomain corresponds to the interval $(0, 1)$
- Taken any $\underline{x}$, the logistic function classifies it to the "1" class if $p(1|\underline{x}) > 0.5$, otherwise to the "0" class.

Figure 4.7: Linear regression (left) **vs** Logistic regression (right) for classification problems. Notice on the left the fact that values under 0 for certain values of $\underline{x}$ are possible.



Considering the ***binomial*** case ($p(1|\underline{x}) = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$) with $\beta_0 = 0$ and $\beta_1 = 1$, as shown in figure 4.7, the curve is sigmoid shape with intercept in $y = 0.5$. It increases when the value of $\beta_1 > 0$.

In the binomial case, the logistic function can be written as:

$$
\begin{aligned}
p(1|\underline{x}) &= \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}} && \text{binomial case for simplicity}\\
&= \frac{e^{\beta_0+\beta_1 x_1}}{e^0+e^{\beta_0+\beta_1 x_1}} && \text{reorganizing for clarity}\\
&= \frac{1}{e^{\beta_0+\beta_1 x_1}} \cdot \frac{e^{\beta_0+\beta_1 x_1}}{\frac{e^0}{e^{\beta_0+\beta_1 x_1}}+1} && \text{by collecting } e^{\beta_0+\beta_1 x_1} \text{ from the denominator}\\
&= \frac{1}{\frac{e^0}{e^{\beta_0+\beta_1 x_1}}+1} && \text{simplifying}\\
&= \frac{1}{1+e^{-\beta_0-\beta_1 x_1}} && \text{since } \frac{a^x}{a^y} = a^{x-y}
\end{aligned}
$$

Let us define a quantity called **odds**, which is the ratio between positive and negative outcomes (the fraction of the probability of obtaining result 1 over the one corresponding to the obtaining class 0).

$$
\begin{aligned}
\text{Odds} &= \frac{p(1|\underline{x})}{1-p(1|\underline{x})} && \text{by definition}\\
&= \frac{\frac{e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}}}{1-\frac{e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}}} && \text{since } p(1|\underline{x}) = \underline{\beta}^t \underline{x}\\
&= \frac{\frac{e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}}}{\frac{1+e^{\underline{\beta}^t \underline{x}}-e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}}} && \text{minimum common denominator}\\
&= \frac{\frac{e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}}}{\frac{1}{1+e^{\underline{\beta}^t \underline{x}}}} && \text{simplifying}\\
&= \frac{e^{\underline{\beta}^t \underline{x}}}{1+e^{\underline{\beta}^t \underline{x}}} \cdot \frac{1+e^{\underline{\beta}^t \underline{x}}}{1} && \text{reorganizing}\\
&= e^{\underline{\beta}^t \underline{x}} && \text{simplifying}
\end{aligned}
$$

As it can be seen from the formula, Odds $= \frac{p(1|\underline{x})}{1-p(1|\underline{x})} = e^{\underline{\beta}^t \underline{x}}$ can take any value between $[0, +\text{inf}]$, indicating very low and high probabilities of positive outcome respectively (the bigger is the value of $p(1|\underline{x})$, the lower is the value of $1-p(1|\underline{x})$).

Example: on average 1 in 5 people with an odds of $\frac{1}{4}$ will default, since $p(X) = 0.2$ implies an odds of $\frac{0.2}{1-0.2} = \frac{1}{4}$.

The predictor for this measure is called **log-odds** or **logit** and it is obtained applying a logarithmic tranformation, hence:

$$\log\left(\frac{p(1|\underline{x})}{1-p(1|\underline{x})}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \underline{\beta}^t \underline{x}$$

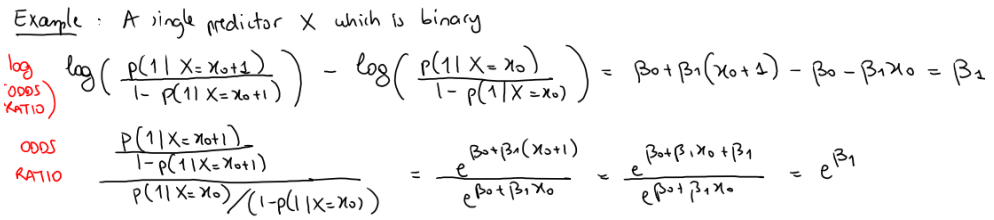
The logistic regression model (assuming the random variable to be Bernoulli distributed $[Y|\underline{X} = \underline{x} \sim Bernoulli(p(1|\underline{x}))]$) is defined as follows:

$$\log\left(\frac{p(1|\underline{x})}{1-p(1|\underline{x})}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

The coefficients can be interpeted as follows: a one-unit increase in x_j (while everithing else is kept constant) induces a change in the log-odds of β_j, while instead it multiplies the odds by e^{β_j} (see the example written below).

Example: Logistic regression for a single predictor X which is binary.

Figure 4.8: Demonstration written by the professor



To compute the change of the log-odds for a one-unit change of X we compute

$$\begin{aligned}
\log(\text{odds ratio}) &= \log\left(\frac{p(1|X=x_1+1)}{1-p(1|X=x_1+1)}\right) - \log\left(\frac{p(1|X=x_1)}{1-p(1|X=x_1)}\right) && \text{log odds in 1 minus log odds in 0}\\
&= \beta_0 + \beta_1(x_1+1) - \beta_0 - \beta_1 x_1 && \text{by definition of logistic regression}\\
&= \beta_1 && \text{simplifying}
\end{aligned}$$

This quantity is called **log(odds ratio)** since the subtraction of two logarithms in the same base is equal to the logarithm of the ratio of the arguments. Notice that, as expected, a one unit increase of the value of x_1 results in an increment of β_1.

If we instead consider the same increment for the **odds ratio**, we get:

$$\begin{aligned}
\text{odds ratio} &= \frac{\frac{p(1|X=x_0+1)}{1-p(1|X=x_0+1)}}{\frac{p(1|X=x_0)}{1-p(1|X=x_0)}} && \text{odds in 1 divided by odds in 0}\\
&= \frac{e^{\beta_0+\beta_1(x_0+1)}}{e^{\beta_0+\beta_1 x_0}} && \text{using the fact that } \log\left(\frac{p(1|\underline{x})}{1-p(1|\underline{x})}\right) = \underline{\beta}^t\underline{x}\\
&= \frac{e^{\beta_0+\beta_1 x_0+\beta_1}}{e^{\beta_0+\beta_1 x_0}} && \text{rearranging}\\
&= e^{\beta_1} && \text{since } \frac{x^a}{x^b} = x^{a-b}
\end{aligned}$$

Therefore we have an increase of e^{β_1} as expected.

4.2.3 Parameter estimation in logistic regression

When estimating parameters for non-linear regressions you often get non-closed form expressions, therefore the maximum likelihood(s) for the parameter(s) must be maximised numerically. Notice

that, contrarily to linear regression, you cannot use the least square method.

For instance, give some training data $(\underline{x}_i, y_i),\ i = 1, \ldots, n$, we want to estimate the best values of $\beta_0, \beta_1, \ldots, \beta_p$ to fit the values of y. To do so we need to compute the maximum likelihood given that Y is binary, thus

$$Y|\underline{X} = \underline{x} \sim Bernoulli(p(1|\underline{x})),\ f(y|\underline{x}) = p(1|\underline{x})^y(1 - p(1|\underline{x}))^{1-y}$$

To put into words what is written here, for dummies like me, the probability that a point, obtained given some $\underline{x}$, is classified to the 0 class, is equal to $p(1|\underline{x})^0(1-p(1|\underline{x}))^{1-0} = 1*(1-p(1|\underline{x}))$. Coherently, the probability that a point is classified to the 1 class, is equal to $p(1|\underline{x})^1(1-p(1|\underline{x}))^{1-1} = p(1|\underline{x})*1$. The probabilities of all the points are multiplied to obtain the likelihood function, written below:

$$L(\underline{\beta}) = \prod_{i=1}^{n} f(y_i|\underline{x}_i) = \prod_{i=1}^{n} p(1|\underline{x}_i)^{y_i}(1 - p(1|\underline{x}_i))^{1-y_i} \qquad \text{given some values of } \beta_0 \text{ and } \beta_1$$

We then compute the log-likelihood and reorganize it in order to make evident its dependence from $\underline{\beta}$:

$$\begin{aligned}
l(\underline{\beta}) &= \sum_{i=1}^{n} [y_i \log(p(1|\underline{x}_i)) + (1 - y_i)\log(1 - p(1|\underline{x}_i))] && \text{applying logarithm} \\
&= \sum_{i=1}^{n} [y_i \log(p(1|\underline{x}_i)) - y_i \log(1 - p(1|\underline{x}_i)) + \log(1 - p(1|\underline{x}_i))] && \text{multiplying} \\
&= \sum_{i=1}^{n} \left[y_i \log\left(\frac{p(1|\underline{x}_i))}{1 - p(1|\underline{x}_i))} \right) + \log(1 - p(1|\underline{x}_i)) \right] && \log(a) - \log(b) = \log\left(\frac{a}{b}\right) \\
&\text{but since } \log\left(\frac{p(1|\underline{x})}{1 - p(1|\underline{x})} \right) = \underline{\beta}^t \underline{x} \text{ and } p(1|\underline{x}) = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}} \\
&= \sum_{i=1}^{n} \left[y_i \underline{\beta}^t \underline{x}_i + \log\left(1 - \frac{e^{\underline{\beta}^t \underline{x}_i}}{1 + e^{\underline{\beta}^t \underline{x}_i}} \right) \right] \\
&= \sum_{i=1}^{n} \left[y_i \underline{\beta}^t \underline{x}_i + \log\left(\frac{1}{1 + e^{\underline{\beta}^t \underline{x}_i}} \right) \right] && \text{using mcm and simplifying} \\
&= \sum_{i=1}^{n} \left[y_i \underline{\beta}^t \underline{x}_i + \log(1) - \log\left(1 + e^{\underline{\beta}^t \underline{x}_i} \right) \right] && \text{dividing the logarithms} \\
&= \sum_{i=1}^{n} \left[y_i \underline{\beta}^t \underline{x}_i - \log\left(1 + e^{\underline{\beta}^t \underline{x}_i} \right) \right] && \text{since } \log((1) = 0
\end{aligned}$$

We then compute the first derivative in $\underline{\beta}$ (notice that since you want to derive in a vector, you

need to take the partial derivatives in each of the elements of the vector):

$$\frac{\partial l(\beta)}{\partial \beta_j} = \frac{\partial}{\partial \beta_j}\left(\sum_{i=1}^{n}\left[y_i \underline{\beta}^t \underline{x}_i - \log\left(1 + e^{\underline{\beta}^t \underline{x}_i}\right)\right]\right) \quad j = 0, \ldots, p$$

but $\frac{\partial \underline{\beta}^t \underline{x}}{\partial \beta_j} = \frac{\partial}{\partial \beta_j}(\beta_0 x_{i0} + \cdots + \beta_p x_{ip}) = x_{ij}$ since the other terms simplify, consequently

$$= \sum_{i=1}^{n}\left[y_i x_{ij} - \frac{1}{1 + e^{\underline{\beta}^t \underline{x}_i}} * e^{\underline{\beta}^t \underline{x}_i} * x_{ij}\right] = \sum_{i=1}^{n}\left[y_i x_{ij} - \frac{e^{\underline{\beta}^t \underline{x}_i}}{1 + e^{\underline{\beta}^t \underline{x}_i}} x_{ij}\right]$$

but $\frac{e^{\underline{\beta}^t \underline{x}_i}}{1 + e^{\underline{\beta}^t \underline{x}_i}} = p(1|\underline{x})$ therefore

$$= \sum_{i=1}^{n}\left[y_i x_{ij} - p(1|\underline{x}_i) x_{ij}\right]$$

By setting this derivative to zero you obtain $p + 1$ equations in $p + 1$ parameters (where p is the number of β parameters), therefore to optimize numerically the function (by solving the equations) you must use an optimization method like **Newton-Raphson**. The *gml* function in R uses an iterative reweighted least-squares algorithm, which is a variation of the Newton-Raphson method.

From the optimization you get $\underline{\hat{\beta}} = \hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, which can be used to estimate $p(1|\underline{x})$, since

$$\hat{p(1}|\underline{x}) = \frac{e^{\underline{\hat{\beta}}^t \underline{x}}}{1 + e^{\underline{\hat{\beta}}^t \underline{x}}}$$

Look to pages $137 - 139$ from "*An introduction to statistical learning with applications in R*" (doi:10.1080/24754269.2021.1980261) for examples.

Chapter 5

Generalized linear models

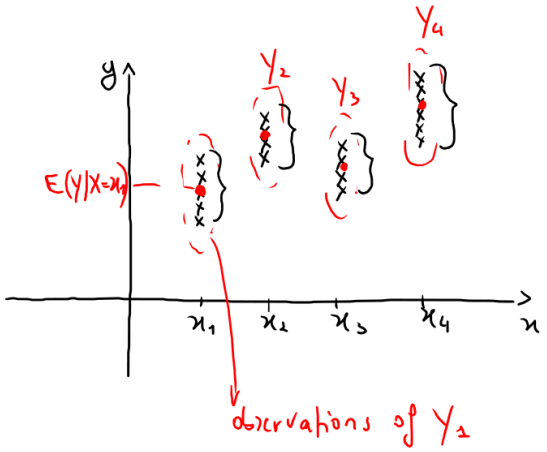
Definition 1. *A regression model provides a function that describes the relationship between one or more independent variables and a response, dependent, or target variable (definition taken from Reference-Site).*

Generalized linear models (GLMs) are a broad family of regression models which includes, for instance, linear, logistic and poisson regressions (and many more, some of them are shown in figure 5.2). The concept was created by Nelder and Wedderburn in 1972, in order to map into a single framework all regressions models. In general regression models are characterized by:

- Y: the random **response variable** you want to predict
- $\underline{X} = (X_1, \ldots, X_p)$: a **vector of random predictors** (generally known)

Notice that, since these models describe the value of Y with respect to the vector $\underline{X}$ (they give the value of $Y|\underline{X} = \underline{x}$), they do not give information on the actual distribution of Y, but rather on the values that it takes conditionally to the values taken by the predictors, which can be written as

Figure 5.1: The values of a specific Y_i are not distributed in the same way as the other Y_i usually



$$Y_i = (Y|\underline{X} = \underline{x}_i)$$

Therefore, if you have n vectors of random predictors, $Y_1, \ldots, Y_n$ are independent random variables but generally they are not identically distributed, as shown in figure 5.1.

5.1 From linear models to generalized linear models

Linear models make some implicit assumptions that must be removed in order to generalize.

Linear models	**Generalized linear models**
$Y_i \sim N(\mu_i, \sigma^2)$	$Y_i \sim f(y_i; \mu_i, \phi)$
$\mu_i = E[Y_i] = E[Y\|\underline{X} = \underline{x}_i] = \eta_i$	$g(\mu_i) = \eta_i$
$\eta_i = \underline{x}_i^t \underline{\beta} = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$	$\eta_i = \underline{x}_i^t \underline{\beta}$

First of all, linear models assume that the each observation (random variable) Y_i is normally distributed, with mean μ_i, meaning that $E[Y|\underline{X} = \underline{x}_i] = \mu_i$, and $\sigma^2 = Var(Y_i) = Var(Y|\underline{X} = \underline{x}_i)$. It also assumes that the variance is constant, therefore *homoskedasticity* is taken as a fundamental condition. The values of each mean μ_i is calculated exactly through this calculation: $\mu_i = x_i^t \beta = \nu_i$. The β coefficients are calculated through tecniques like the Least Square method and the Maximum Likelihood Estimation.

On the other hand, Generalized Linear Models do not make any assumptions on the shape of the distribution from which it comes Y_i: *Y_i is distributed according to some function $f(y_i)$ with the parameters μ_i and ϕ.* ϕ is called **special parameter** and it may or may not be present in the model. The **linear predictor** is the function used to predict the outcome of the random variable Y_i knowing some values for the predictor $\underline{X}_i$; in both cases the linear predictor is defined as $\eta_i = \underline{x}_i^t \underline{\beta} = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$, where p is the number of parameters and the coefficients $\beta_0, \ldots, \beta_p$ are the weights for the predictors.
In generalized linear models μ_i and η_i are related by a **link function**, which is a function g of μ_i such that $g(\mu_i) = \eta_i$. Contrarily to what we would think, the link function doesn't represent a tranformation

A generalized linear model is then obtained by choosing

- The **systematic component**: represents the "correlation" between the predictors and the outcome
- The **link function**: a function that acts over the the systematic component, it "delinearizes" the correlation between predictors and outcomes.
- The **random component**: it is the distribution of the observations. They could be distributed in different ways, including Poisson, negative binomial excetera.

A linear regression model is for example obtained considering:

- link function: 1, since systemic component ($\beta^t \underline{x}$) remains linear
- random component: normal distribution, since all Y_i are normally distributed

A series of possibly makable models are reported in the 5.2.

Figure 5.2: Here are represented for each line a type of regression model obtained as a generalized linear model

NAME	LINK FUNCTION	DISTRIBUTION
GENERAL LINEAR MODEL	IDENTITY $b0 + b1X$	NORMAL
LOGISTIC REGRESSION	LOGIT $\frac{e^{b0+b1X}}{1+e^{b0+b1X}}$	BINOMIAL
POISSON	LOG e^{b0+b1X}	POISSON
GAMMA	INVERSE $\frac{1}{b0+b1X}$	GAMMA

5.2 Exponential dispersion family

The function f, describing the distribution of the random variable Y_i, needs to be a member of the **exponential dispersion family (EDF)**. A density belonging to the EDF can be written as

$$f(y_i; \theta_i, \phi) = \exp\left\{ \frac{y_i \theta_i - b(\theta_i)}{a_i(\phi)} + c(y_i; \phi) \right\} \tag{5.1}$$

where:

- $f(y_i, \theta_i, \phi)$ is the density function that has to be tested. θ_i and ϕ are some parameters. The values of θ_i are called *canonical parameters*.
- a_i, b and c are some other functions.

It is obvious that, if a density function could be related to this family, it is possible to derive from it the functions a_i, b and c, and the parameter θ_i. Now we'll try to understand if some important density functions belong to the EDF family.

5.2.1 Example: normal distribution is a member of the EDF

We want to demonstrate that the normal distribution is a member of the EDF.

$$
\begin{aligned}
f(y_i;\mu_i,\sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2}\frac{(y_i-\mu_i)^2}{\sigma^2}\right\} && \text{definition of normal distribution}\\
&= \exp\left\{-\frac{1}{2}\frac{(y_i-\mu_i)^2}{\sigma^2}+\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)\right\} && \text{pull coefficient inside the exponential}\\
&= \exp\left\{-\frac{1}{2}\frac{(y_i-\mu_i)^2}{\sigma^2}-\frac{1}{2}\log\left(2\pi\sigma^2\right)\right\} && \text{rearrange using log properties}\\
&= \exp\left\{-\frac{1}{2}\frac{y_i^2-2y_i\mu_i+\mu_i^2}{\sigma^2}-\frac{1}{2}\log\left(2\pi\sigma^2\right)\right\} && \text{compute the square}\\
&= \exp\left\{\frac{y_i\mu_i+\mu_i^2/2}{\sigma^2}-\frac{1}{2\sigma^2}y_i^2-\frac{1}{2}\log\left(2\pi\sigma^2\right)\right\} && \text{group terms containing } \mu_i
\end{aligned}
$$

The normal distribution therefore belongs to the EDF by setting:

$$
\begin{aligned}
\theta_i &= \mu_i,\\
\phi &= \sigma^2,\\
b(\theta_i) &= \theta_i^2/2,\\
a_i(\phi) &= \phi,\\
c(y_i;\phi) &= -\frac{1}{2\phi}y_i^2-\frac{1}{2}\log(2\pi\phi)
\end{aligned}
$$

5.2.2 Example: binomial distribution is a member of the EDF

We want to demonstrate that the binomial distribution is a member of the EDF.

$$
\begin{aligned}
f(y_i;n_i,\pi_i) &= \binom{n_i}{y_i}\pi_i^{y_i}(1-\pi_i)^{n_i-y_i} && \text{definition of binomial distribution}\\
&= \exp\left\{\log\left(\binom{n_i}{y_i}\pi_i^{y_i}(1-\pi_i)^{n_i-y_i}\right)\right\} && \text{exponentiate to compare with EDF}\\
&= \exp\left\{y_i\log\pi_i+(n_i-y_i)\log(1-\pi_i)+\log\binom{n_i}{y_i}\right\} && \text{split log and simplify}\\
&= \exp\left\{\frac{y_i\log\left(\frac{\pi_i}{1-\pi_i}\right)+n_i\log(1-\pi_i)}{1}+\log\binom{n_i}{y_i}\right\} && \text{group terms containing } \pi_i
\end{aligned}
$$

The binomial distribution therefore belongs to the EDF by setting:

$$\begin{aligned}
\theta_i &= \log\left(\frac{\pi}{1-\pi}\right) \textbf{(logit)} \\
\phi &= 1 \\
b(\theta_i) &= -n_i \log(1-\pi_i) \\
&= -n_i \log\left(1 - \frac{e^{\theta_i}}{1+e^{\theta_i}}\right) \qquad \text{substitution of } \pi_i \text{ because of the calculation in 5.2} \\
&= n_i \log\left(1+e^{\theta_i}\right) \\
a_i(\phi) &= \phi \\
c(y_i;\phi) &= \log\binom{n_i}{y_i}
\end{aligned}$$

We replaced $\pi_i = \frac{e^{\theta_i}}{1+e^{\theta_i}}$ since:

$$\begin{aligned}
e^{\theta_i} &= \frac{\pi_i}{1-\pi_i} \\
&\Rightarrow (1-\pi_i)e^{\theta_i} = \pi_i \\
&\Rightarrow e^{\theta_i} - \pi_i e^{\theta_i} = \pi_i \\
&\Rightarrow \pi_i = \frac{e^{\theta_i}}{1+e^{\theta_i}}
\end{aligned} \tag{5.2}$$

For the binomial distribution ($Y_i \sim Bin(n_i, \pi_i)$) it is possble to write the relation between θ_i to μ_i.

$$E[Y_i] = \mu_i = n_i\pi_i = n_i\frac{e^{\theta_i}}{1+e^{\theta_i}} \qquad \text{Link between } \mu_i \text{ and } \theta_i$$

Consequently, it is possible to rewrite the value of θ_i as follows

$$\pi_i = \frac{\mu_i}{n_i} \implies \theta_i = \log\left(\frac{\pi_i}{1-\pi_i}\right) = \log\left(\frac{\mu_i/n_i}{1-\mu_i/n_i}\right) = \log\left(\frac{\mu_i}{n_i-\mu_i}\right)$$

Notice that **heteroskedasticity** in naturally embedded in the model since the variance depends on μ_i:

$$Var(Y_i) = n_i\pi_i(1-\pi_i) = n_i\frac{\mu_i}{n_i}\left(1-\frac{\mu_i}{n_i}\right) = \mu_i\left(\frac{n_i-\mu_i}{n_i}\right)$$

This hold true for all GLMs but linear models which is the only homoskedastic model of the group. Most datasets are not homoskedastic.

5.3 Connection between θ_i and μ_i, $Var(Y_i)$

In general, one can show that, for any function in the EDF,

$$\mu_i = E[Y_i] = b'(\theta_i)$$

Where b' is the first derivative of the function b.

$$\begin{aligned} Var(Y_i) &= b''(\theta_i) \cdot a_i(\phi) \\ &= V(\mu_i) \cdot a_i(\phi) \end{aligned}$$

Where b'' is the second derivative of b and $V(\mu_i)$ is the *variance function* (basically just a way to rewrite underlining the dependence on μ).

5.3.1 Example: connection of θ_i , μ_i and $Var(Y_i)$ for the binomial model

We know, for the binomial function, that

$$b(\theta_i) = n_i \log(1 + e^{\theta_i})$$

The first derivative in θ_i then becomes

$$b'(\theta_i) = \frac{n_i e^{\theta_i}}{1 + e^{\theta_i}} = n_i \pi_i$$

But it is also known that

$$\mu_i = n_i \pi_i$$

Therefore, as expected from the general formula

$$\mu_i = b'(\theta_i)$$

We know that, for the binomial model

$$a_i(\phi) = 1 \qquad \text{the function dependent on } \phi \text{ is equal to 1}$$

We then compute the second derivative of $b(\theta_i)$ in θ_i:

$$\begin{aligned} b''(\theta_i) &= n_i e^{\theta_i} \left[- \left(\frac{1}{1 + e^{\theta_i}} \right)^2 e^{\theta_i} \right] + n_i \frac{e^{\theta_i}}{1 + e^{\theta_i}} && \text{by deriving } b'(\theta_i) \\ &= -n_i \left(\frac{e^{\theta_i}}{1 + e^{\theta_i}} \right)^2 + n_i \frac{e^{\theta_i}}{1 + e^{\theta_i}} && \text{pulling both } e^{\theta_i} \text{ inside the square} \\ &= -n_i \pi_i^2 + n_i \pi_i && \text{since } \frac{e^{\theta_i}}{1 + e^{\theta_i}} = \pi_i \\ &= n_i \pi_i (1 - \pi_i) && \text{collecting } n_i \pi_i \end{aligned}$$

Therefore, as expected, we get

$$Var(Y_i) = b''(\theta_i) a_i(\phi) = n_i \pi_i (1 - \pi_i) * 1$$

5.4 Choice of link function

The choice of link function is not unique and can depend on the function f which was chosen. To recall the meaning of link function, refer to section 5.1.

5.4.1 Example: link function for a Bernoulli function

Assume that Y_i is Bernoulli distributed with probability π_i ($Y_i \sim Ber(\pi_i)$). Then:

- $\mu_i = \pi_i \in (0,1)$ (since $n_i = 1$)
- $Var(Y_i) = \pi_i(1-\pi_i) = \mu_i(1-\mu_i)$
- $\eta_i = \underline{x}_i^t \underline{\beta}$ with $\eta_i \in (-\infty, +\infty)$

We must find a function g such that $g(\mu_i) = \eta_i$ which, taking into account that $\mu_i \in (0,1)$ and $\eta_i \in (-\infty, +\infty)$, means:

$$\begin{aligned} g &: (0,1) \to (-\infty, +\infty) \qquad \longrightarrow \text{ link function} \\ g^{-1} &: (-\infty, +\infty) \to (0,1) \end{aligned}$$

where g represents the link function to be defined. These conditions are satisfied by any cumulative density functions (CDF), as long as it is continuous (since it must be invertible). We therefore have infinitely many functions g that satify these conditions, but the three most common choices are:

1. **Standard normal CDF**: Choose $g^{-1} = \Phi$, with Φ being the standard normal Cumulative Distribution Function (CDF), hence

   $$g = \Phi^{-1} : (0,1) \to (-\infty, +\infty)$$

   Where g is the inverse of the CDF. This type of regression ($f \sim$ Bernoulli $, g = \Phi^{-1}$) is called **probit regression**.

2. **Logistic CDF**: The logistic continous probability distribution is shown in figure 5.3. Its cumulative distribution function is the logistic function, which we already encountered in section 4.2, and that is shown in subfigure 5.4. The formula for the general probablity density function is shown below.

   $$f(y; \mu, \sigma^2) = \frac{\exp(^{(y-\mu)}/_{\sigma})}{\sigma^2(1+\exp\left(^{(y-\mu)}/_{\sigma}\right)^2}, \quad \text{with } -\infty < y < +\infty$$

   We take a particular case, setting $\mu = 0$, $\sigma^2 = 1$ we get the following sigmoid function:

   $$g^{-1} = f(y; 0, 1) = \frac{e^y}{1+e^y}$$

   Which is the sigmoid function. The link function then becomes the **logit link**

   $$\frac{e^\eta}{1+e^\eta} = \mu \implies \eta = \log\left(\frac{\mu}{1-\mu}\right)$$

   This regression ($f \sim$ Bernoulli $, g =$ logit) is the **logistic regression**.

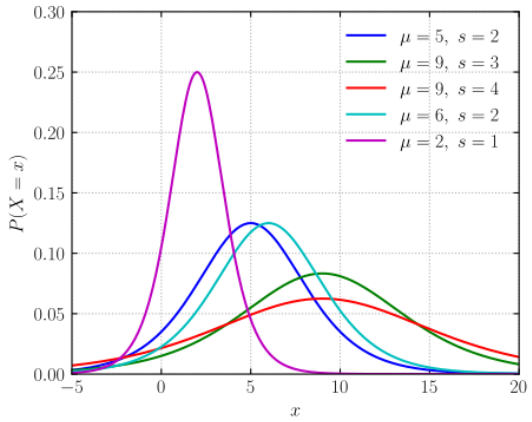


Figure 5.3: Logistic probability density functions AKA logistic distributions

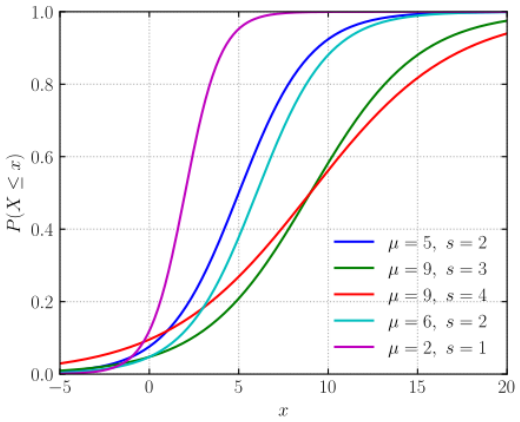
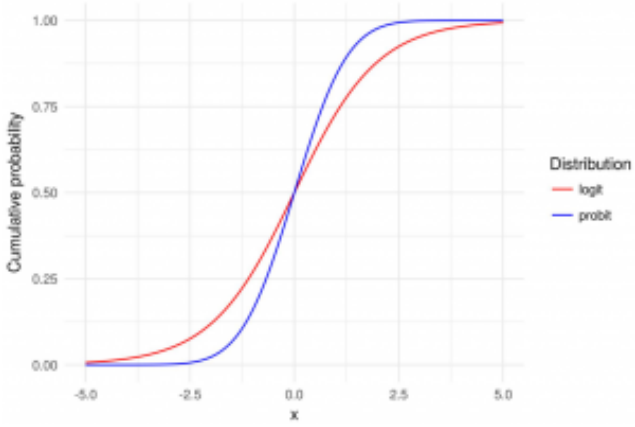


Figure 5.4: Logistic cumulative distributions AKA logistic functions

Figure 5.5: Logistic distribution and function

Figure 5.6: Logistic regression models are also called logit models, while probit regression models are also called probit models. Logit models are used to model Logistic distribution while probit models are used to model the cumulative standard normal distribution, as already said



3. **Poisson CDF**: Assume that we want to understand if a plate is populated by some bacteria. They are distributed accordingly to the Z_i random variable, Poisson distributed with parameter λ_i ($Z_i \sim$ Poisson (λ_i)). y_i takes the following values

$$y_i = \begin{cases} 0 & \text{if } z_i = 0 \text{ (absent)} \\ 1 & \text{if } z_i > 0 \text{ (present)} \end{cases}$$

We would like to describe how y_i changes with time ($x = 0, 1, 2, \dots$), therefore we need a function that links μ_i and η_i. Since we know that

$$\eta_i = \log(\lambda_i)$$

and that

$$\begin{aligned}\mu_i &= E[Y_i] \\ &= 0 \cdot P(Y_i = 0) + 1 \cdot P(Y_i = 1) \\ &= P(z_i > 0) \\ &= 1 - P(z_i = 0) \\ &= 1 - e^{-\lambda_i}\end{aligned}$$

we can link μ_i and η_i passing through λ_i ($\mu_i \leftrightarrow \lambda_i \leftrightarrow \eta_i$). Firstly we rearrange the link between μ_i and λ_i:

$$\mu_i = 1 - e^{-\lambda_i} \implies e^{-\lambda_i} = 1 - \mu_i \implies \lambda_i = -\log(1 - \mu_i)$$

Then, plugging the λ_i value in the link formula between λ_i and η_i ($\log(\lambda_i) = \eta_i$) we get

$$\eta_i = \log(-\log(1 - \mu_i))$$

This link is called **complementary log-log link**.

5.4.2 Canonical link

The logit link is also called **canonical link** since

$$\eta_i = \log\left(\frac{\mu_i}{1 - \mu_i}\right) = \theta_i$$

In general a canonical link is a function g such that

$$\eta_i = g(\mu_i) = \theta_i$$

To put the concept in words, a link function is said to be canonical when the relationship between μ_i and ν_i is equal to that between μ_i and θ_i.

These link functions have some advantages:

- Clear interpretation of the parameters ($\underline{\beta}$)
- Some theoretical reasons (Fisher scoring and iteratively reweighted least squares algorithms are equivalent, zero sum residuals)
- It follows directly from the EDF formulation of the distribution, in fact

$$\begin{aligned}\mu_i &= b'(\theta_i) \\ \theta_i &= (b')^{-1}(\mu_i)\end{aligned}$$

Since $(b')^{-1}(\mu_i)$ is the canonical function, it also represents the link function.

Despite these advantages the logit link is not always the correct one to use.

5.5 Residuals in generalized linear models

As already seen, a linear model can be express either as:

$$y_i = \eta_i + \varepsilon_i \text{ (signal (deterministic linear predictor) + noise representation)}$$

Or as:

$$Y \sim N(\eta_i, \sigma^2)$$

From the first expression we get the usual definition or residuals in linear models:

$$e_i = y_i - \hat{\eta}_i$$

Since for generalized linear models we can only use the formulation $Y \sim N(\eta_i, \sigma^2)$ (meaning that we cannot simply write the model as value plus noise), consequently, we need a new way to define the residuals. There are many options, but the most commonly used ones are

- **Pearson residuals**
- **Deviance residuals**

5.5.1 Pearson residuals

Pearson residuals are defined as follows:

$$r_i^p = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{Var}(y_i)/\hat{\phi}}}$$

Where y_i represents the observed value of y and μ_i is the predicted mean. If we apply this definition, for instance, to a Poisson distribution ($\phi = 1$), we get

$$r_i^p = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}} \text{ with } y_i \in (0, +\infty)$$

Beacause of the interval on which the values of y_i are defined, the residuals may not be symmetric (in fact, the most negative value on the nominator would be $-\hat{\mu}_i$, while instead the most positive will be possibly $+\infty$), but the assumption of gaussianity is usually satisfied with big values of μ.

5.5.2 Deviance residuals

The **deviance** of an observation i is defined as:

$$d_i = 2[l_i(y_i; y_i) - l_i(\hat{\mu}_i; y_i)]$$

Where:

- $\hat{\mu}_i = e^{\underline{x}_i^t \hat{\underline{\beta}}}$
- $l_i(y_i; y_i)$ is the log-likeihood of the saturated model (most fitting non-parametric model, therefore the highest possible likelihood)
- $l_i(\hat{\mu}_i; y_i)$ is the log-likelihood of the model using the estimated parameters

- $d_i \geq 0$, since always $l_i(y_i; y_i) \geq l_i(\hat{\mu}_i; y_i)$

The overall deviance of the model is defined as the sum of the deviance of each observation

$$\text{deviance} := \sum_{i=1}^{n} d_i$$

The lower the deviance, the closer the model to a perfect fit.

The **deviance residuals** can be then defined as:

$$r_i^D = \sqrt{d_i} \cdot \text{sign}(y_i - \hat{\mu}_i) \quad \text{where sign}(y_i - \hat{\mu}_i) = \begin{cases} 1 & y_i > \hat{\mu}_i \\ -1 & y_i < \hat{\mu}_i \\ 0 & y_i = \hat{\mu}_i \end{cases}$$

Chapter 6

Simulating data from a GLM in R

Simulating data is often used to test the performance of your model.

6.1 Example: simulating Poisson data

In general, in a Possion regression you have:

$$Y_i = (Y|\underline{X} = \underline{x}_i) \sim \text{Poisson}(\mu_i)$$

$$\mu_i = E[Y_i] \in (0, +\infty)$$

$$\eta_i = \underline{x}_i^t \underline{\beta} \in (-\infty, \infty)$$

If you then decide to use the canonical link, which is the log function:

$$\eta_i = \log(\mu_i) \implies \mu_i = e^{\eta_i}$$

Therefore you can generalize this Poisson regression as

$$Y_i \sim \text{Poisson}(e^{\underline{x}_i^t \underline{\beta}})$$

Since $\eta_i = e^{\underline{x}_i^t \underline{\beta}}$. To then simulate data to test this model you must:

1. Assume a number of predictors p (in our case $p = 1$ for plotting convenience)
2. Fix some values for $\underline{\beta}$ (in our case notice that $\eta_i = \beta_0 + \beta_i x_i$)
3. Generate n random values of x_i (in this case $n = 100$)
4. Compute the corresponding values of eta_i (in this case, we have 100 values of η)
5. Compute the values of μ by using the correct link function, and consequently the values of y_i
6. Try several methosds to find a model correctly fitting the simulated data

In **R** this translates to:

```r
# Set the seeds
set.seed(123)

x <- rnorm(100, 0, 2)
# Does not matter the distribution, in this case we are getting random values following a
# normal distribution with 0 mean and 2 standard deviation

beta0 <- 1  # Fixed, intercept
beta1 <- 2  # Fixed

eta <- beta0 + beta1 * x  # generated the principal component. #TODO non ricordo il nome preciso
# Eta assumes all the mean values of the linear model

plot(x, eta) # where eta represents the mean values calculated through the components of the mod
# of course it is the points of eta stay on a line

# since we consider the log function as the link function (eta = log(mu))
# consequently mu = exp(eta)
mu <- exp(eta) #I find the real mean values for each Yi,
# that have to be still calculated

y <- rpois(100, lambda=mu)  # generates y values accordingly to poisson distributions
# formulated using the values of mu calculated before.

# I put myself in a situation where I know data (x, y) but I don't know where
# they come from. I use statistical techniques to fit my model on the data
# generated hoping to get the model used to obtain the data

plot(x, y, type = "p")  # See that is not linear, could be log(y)?
plot(x, log(y))  # Since we obtain a linear trend,
# we expect eta to be in the form beta0 + beta1x

# Try fitting a line (not the best approach)
z <- log(y)
summary(z) # typical Poisson data contain several 0s, which
# give -infinites

#Transform the data
z <- log(y+1)  # Sudo-count transformation (commonly used)
summary(z)
plot(x, log(y+1))

modA <- lm(formula=z~x)  # Fit a line in z, since we found a linear behaviour
summary(modA)  # Estimated parameters are far from the chosen ones. The estimates of the
# betas are completely different

plot(modA)  # Notice systematic trends in Residual vs
# Fitted analysis: it is suggested the presence of a
```

```
# quadratic term in the residuals. A good pattern would be a random
# distribution around 0. QQ plot is ok. Scale-location checks for homoskedasticity, it is not ok
# From the Residual vs Leverage plot it seems that there is a systematic trend of the
# residuals, while instead they should be distributed above and below

# add a quadratic term
modB <- lm(z~x+I(x^2))  # Adding quadratic term, take x, produces x^2 and puts it in the formula
summary(modB)  # Not that far from starting data
plot(modB)  # Not the best but seems decent

# Proper way to try and fit data y[1:100]
y[1:100]

# Inspect the data and notice that the response in positive
#discrete You could then try using generalized linear models, and in particular Poisson regressi
# Poisson seems reasonable, which is the simplest to fit this kind of data.
# Negative binomial is another option, that we will not consider in this case.

modC <- glm(formula=y~x, family="poisson")  # no transformation.
# Default is gaussian, change to Poisson, it takes the canonical link
summary(modC)d
# Residuals are not the same in case of generalized linear models
# Really near 1 and 2, the real values
# Note: Because of how glm works, you need decently big values for n. In linear
# regression models the beta hat vector has a gaussian distribution, because there
# is an assumpion of gaussianity in the model. In the case of other regression
# models, as the poisson regression, gaussianity is not respected if n is not
# large. For this reason, the parameters obtained are an approximation.
# The dispersion parameter phi, that we found in the general linear models section, is set to 1
# Note: Fisher Scoring gives info on optimization attempts number. The program starts with some
# values of beta, and then tries other values.

plot(modC)  # Seems decent
# note the presence of another type of error. scale-location shows some values to
# go up and down, although we know this is the correct model
# Note: Some regular-ish patterns may be seen when working with discrete data

# Use the good model to make predictions
p1<-predict(modC)
# Calculates for each x the value of eta
# predict(modC) == coefficients(modC)[1] + coefficients(modC)[2] * x
summary(p1)
# we compare the estimated values of eta with those obtained through
# log(y), which are the non-parametric eta hat which goes over the y points.
# If the model fits well, in theory the two should be near
plot(p1, log(y))
abline(0,1)
```

```
p2<-predict(modC,type="response")  # Estimates values of mu hat.
# R will output the exp(linear correlation) values for each x,
# instead of the values of eta. mu obviously represent
# the best prediction of y

summary(p2)
plot(p2,y)  # Plot predicted vs simulated values of y
abline(0,1)  # Line that should be approximated by the
# datapoints. p2 predicts y from x, and since it gives values
# comparable to y, y can be predicted by using x
```

6.2 Example: simulating binomial data

You can do the same with other distributions. This is the example for a binomial distribution.

```
# Example 2, same xs and beta, and eta, change mu. Have to think to the link
# function
x <- rnorm(100, 0, 2)
beta0 <- 1
beta1 <- -2
eta <- beta0 + beta1 * x
mu <- exp(eta) / (1 + exp(eta))  # Use logit as the link function
summary(mu)

plot(eta, mu) #it is visible the "transformation" to a sigmoid line
# of the linear function eta
plot(x, mu)

# The two plots are different, although they are really similar now.
# For example, changing the value of beta1 to a negative value, we
# would obtain a reversed curve in the case of plot(x, mu), while instead
# the graph would remain unchanged in the case of plot(eta, mu)

y <- rbinom(100, size=1, prob=mu) # Compute Bernoulli y values
plot(x, y) # only 0 and 1 values were generated
table(y)

modD <- glm(y~x, family="binomial")  # Create the model, default
# is logit

p4<-predict(modD) # Compute linear predictors
summary(p4)
p5<-predict(modD,type="response") # Compute the fitted means, which
# are the probabilities
summary(p5)
```

Chapter 7

Discriminant analysis

In a classification setting, the Y response variable is a categorical variable that can indicate to any class j of the k possible classes. The aim of classification is estimating the probability of an observation $\underline{x}$ belonging to any class j, namely

$$P(Y = j|\underline{X} = \underline{x}),\ j = 1, \dots, k$$

Importantly, compared to other methods, discriminant analysis methods are usually flexible to any nymber of classes $(2, \dots, k)$.

Given the observation $\underline{x}$, what is the probability that it belongs to a specific class j. In discriminant analysis (LDA, QDA, naive Bayes...), contrarily to other methods already seen in the classification setting section (3.4) methods, the values of $P(Y|\underline{X} = \underline{x})$ are not obtained directly. Instead, the focus is on estimating:

$$f(\underline{x}|Y = j) = f_j(\underline{x})$$

which basically corresponds to the opposite conditional probability. With discriminant analysis methods, first you need to assume a distribution of x, then you estimate the conditional probability density mentioned above (probability density of $\underline{x}$ given a certain class j). To obtain the other probability, you exploit the Bayes theorem:

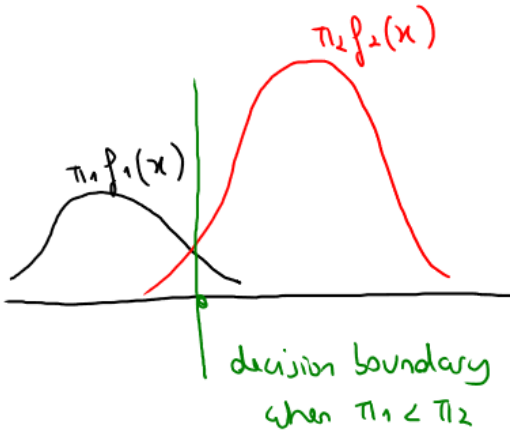
$$\underbrace{P(Y = j|\underline{X} = \underline{x})}_{\text{posterior probability}} = \frac{f(\underline{x}|Y = j)P(Y = j)}{f(\underline{x})} \qquad \text{since } P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$= \frac{f(\underline{x}|Y = j)p(Y = j)}{\sum_{i=1}^{k} f(\underline{x}|Y = i)P(Y = i)} \qquad \text{since you have a discrete number of classes}$$

$$= \frac{f_j(\underline{x}) \overbrace{\pi_j}^{a\ priori\ \text{prob.}}}{\sum_{i=1}^{k} f_i(\underline{x}) \underbrace{\pi_i}_{a\ priori\ \text{prob.}}} \qquad \text{just rewriting in a clearer manner}$$

According to Bayes classifier, we choose j that maximises $P(Y = j|\underline{X} = \underline{x})$, but since the denominator does not depend on j, we can just consider the numerator; the problem then becomes assigning $\underline{x}$ to the class j which maximises $f_i(\underline{x})\pi_j$ (see figure 7.1).

Figure 7.1: Evaluation process of the class of an observation $\underline{x}$, considering only two possible classes, take the class which has higher value $f_i(\underline{x})\pi_j$.



7.1 Estimating the *a priori* probability

In order to estimate $\pi_j,\ j = 1, \ldots, k$, there are two options:

- $\hat{\pi}_j = 1/k$, to be used when all k classes are equally likely.
- $\hat{\pi}_j = n_j/n$ with n_j being the number of observations belonging to class j and with $n = n_1 + \cdots + n_k$, which means assigning to each class a weight dependent on its empirical frequence.

Instead, $f_j(\underline{x})$ is a p-dimensional density which requires some assumptions; the chosen assumptions define which method is used, LDA, QDA or naive Bayes.

7.2 Discriminant analysis methods

7.3 Linear discriminant analysis

7.3.1 LDA with one predictor variable

LDA (assuming $p = 1$ to make it simple) makes two assumptions:

- **Gaussianity**, meaning that all classes are gaussian, hence:

$$f_j(x) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_j}\right)^2},\ x \in \mathbb{R}$$

  Where:

  - μ_j is the mean
  - σ_j^2 is the variance of X conditional on $Y = j$

This assumption does not work if the predictor is discrete.

- **Constant variance**, meaning that different groups, to be distinguished by using LDA, have to have **different means and the same variance**:

$$\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2 = \sigma^2$$

Writing these two assumptions with a single expression we get:

$$(X|Y = j) \sim N(\mu_j, \sigma^2) \tag{7.1}$$

7.3.2 Linear decision boundary (one predictor)

Assume we want to classify an observation x using the LDA method (it is correct to use x instead of $\underline{x}$ since in this case we have only a single parameter) Starting from the Bayes formula we have:

$$\begin{aligned} P(Y = j|\underline{X} = \underline{x}) &= \frac{f_j(\underline{x})\pi_j}{\sum_{i=1}^k f_i(\underline{x})\pi_i} && \text{rearranged for discriminant analysis} \\ &= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma}\right)^2}\pi_j}{\sum_{i=1}^k \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2}\pi_i} && \text{because of the assumptions written in formula 7.1} \\ &= \frac{e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma}\right)^2}\pi_j}{\sum_{i=1}^k e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2}\pi_i} && \text{simplifying} \end{aligned}$$

We then assign x to the class j which maximises this function, but since the denominator does not depend on j, it corresponds to maximising the numerator, namely

$$e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2}\pi_j$$

or equivalently the logarithm of it

$$\log(\pi_j) - \frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2 = \log(\pi_j) - \frac{x^2}{2\sigma^2} + \frac{x\mu_j}{\sigma} - \frac{\mu_j^2}{2\sigma^2}$$

But since $-\frac{x^2}{2\sigma^2}$ does not depend on j (and because of that we consider it as a constant) we can simplify to

$$\underbrace{\log(\pi_j) - \frac{\mu_j^2}{2\sigma^2}}_{\text{intercept}} + x \overbrace{\frac{\mu_j}{\sigma}}^{\text{first coefficient } \beta_1}$$

We thus assign x to class j that maximises

$$\delta_j(x) = \left(\log(\pi_j) - \frac{\mu_j^2}{2\sigma^2}\right) + x\left(\frac{\mu_j}{\sigma}\right)$$

and if we define:

$$(\beta_0)_j = \left(\log(\pi_j) - \frac{\mu_j^2}{2\sigma^2}\right)$$
$$(\beta_1)_j = \left(\frac{\mu_j}{\sigma}\right)$$

We notice that the expression is in linear form:

$$\delta_j(x) = (\beta_0)_j + x(\beta_1)_j$$

In the case of two classes, the *decision boundary* is given by:

$$\delta_1(x) = \delta_2(x)$$

Which is linear.

7.3.3 Estimating parameters (one predictor)

The parameters to use in the LDA method (namely $\pi_1, \ldots, \pi_k$, $\mu_1, \ldots, \mu_k$ and σ^2) must be estimated. The estimators for these parameters are:

$$\hat{\pi}_j = \frac{n_j}{n} \qquad \text{empirical frequency of class } j$$
$$\hat{\mu}_j = \sum_{i:y=j} \frac{x_i}{n_j} \qquad \text{sample mean of class } j$$
$$\hat{\sigma^2} = \frac{\sum_{j=1}^k (n_j - 1) S_j^2}{\sum_{j=1}^k (n_j - 1)} = \frac{\sum_{j=1}^k \sum_{i:y=j} (x_i - \hat{\mu}_j)^2}{n - k} \qquad \text{pooled variance, } S_j \text{ is the sample variance}$$

The method assumes homoskedasticity; for this reason, rather than using the empirical variance of the individual group, a weighted average of the variances of all groups is used. This measure in called pooled variance.

Having defined these estimators, x is classified plugging into the following equation the values of the estimate

$$\hat{\delta_j}(x) = \log(\hat{\pi}_j) - \frac{\hat{\mu}_j^2}{2\hat{\sigma^2}} + x\frac{\hat{\mu}_j}{\hat{\sigma}},\ j = 1, \ldots, k$$

The class assigned is the one producing the highest value of $\delta(x)$.

7.3.4 Multivariate LDA

We talk about multivariate LDA when $p \geq 1$, meaning that you do not have a single predictor variable, as in the example written above, but rather a vector of predictors $\underline{X} = (X_1, \ldots, X_p)$. In this model, the probability of an observation belonging to a certain class j is **multivariate normal** distributed, meaning

$$(\underline{X}|Y = j) \sim N_p(\underline{\mu_j}, \Sigma)$$

Where:

- p is the number of predictors

Figure 7.2: Multivariate normal distribution. It is shown the case with only two variables

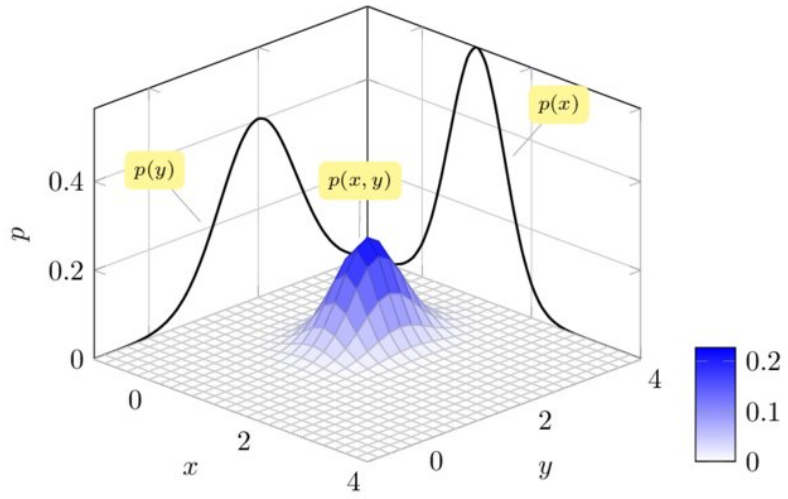
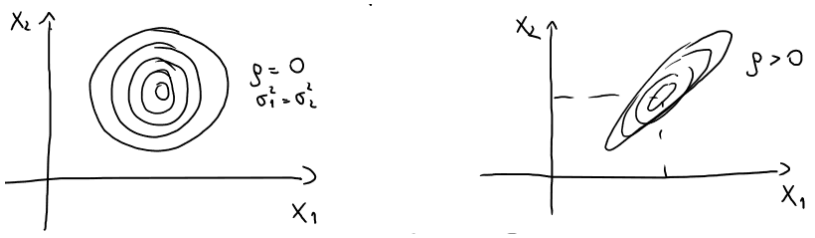


Figure 7.3: Multivariate normal distribution without correlation / with correlation



- $\underline{\mu_j} = \begin{pmatrix} \mu_{1j} \\ \vdots \\ \mu_{pj} \end{pmatrix}$ is a vector of mean values (which is class dependent)
- $\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \sigma_{p,1} & \cdots & \cdots & \sigma_p^2 \end{pmatrix}$ is the covariance matrix. This model assumes that each of the predictors is normal distributed and that the covariance matrix is the same for all groups (meaning $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_k = \Sigma$).

Remember that:

- $\sigma_{i,e} = Cov(X_i, X_e)$
- $\sigma_e^2 = Cov(X_e, X_e) = Var(X_e)$

7.3.5 Linear decision boundary (multiple predictors)

The fact that the decision boundary for LDA is linear can be demonstrated.

$$
\begin{aligned}
P(Y=j|\underline{X}=\underline{x}) &= \frac{\pi_j f_j(\underline{x})}{\sum_{i=1}^k \pi_i f_i(\underline{x})} && \text{from Bayes formula} \\
&= \frac{\pi_j \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma^{-1}(\underline{x}-\underline{\mu_j})\right\}}{\sum_{i=1}^k \pi_i \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma^{-1}(\underline{x}-\underline{\mu_j})\right\}} && f_j(\underline{x}) \sim \text{ multiv. normal} \\
&= \frac{\pi_j \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma^{-1}(\underline{x}-\underline{\mu_j})\right\}}{\sum_{i=1}^k \pi_i \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma^{-1}(\underline{x}-\underline{\mu_j})\right\}} && \text{simplifying}
\end{aligned}
$$

We then assign observation $\underline{x}$ to class j which maximises the logarithm of the numerator (since the denominator does not depend on j)

$$
\begin{aligned}
\delta_j(\underline{x}) &= \log(\pi_j) - \frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma^{-1}(\underline{x}-\underline{\mu_j}) && \text{function to maximise} \\
&= \log(\pi_j) - \frac{1}{2}\underline{x}^t\Sigma^{-1}\underline{x} + \underline{x}^t\Sigma^{-1}\underline{\mu_j} - \frac{1}{2}\underline{\mu_j}^t\Sigma^{-1}\underline{\mu_j} && \text{multiplying} \\
&\propto \left(\log(\pi_j) - \frac{1}{2}\underline{\mu_j}^t\Sigma^{-1}\underline{\mu_j}\right) + \underline{x}^t\left(\Sigma^{-1}\underline{\mu_j}\right) && \text{ignoring terms not depending on } j \\
&\propto (\beta_0)_j + \underline{x}^t(\underline{\beta})_j && \text{with } (\underline{\beta})_j = (\beta_{1j} \ \cdots \ \beta_{pj})^t \\
&= \beta_{0j} + x_i\beta_{1j} + x_2\beta_{2j} + \cdots + x_p\beta_{pj} && \text{which is linear}
\end{aligned}
$$

7.3.6 Estimating parameters (multiple predictors)

The estimated parameters are analogous to the single predictor case:

$$
\begin{aligned}
\hat{\pi}_j &= \frac{n_j}{n} && \text{empirical frequency of class } j \\
\underline{\hat{\mu}_j} &= \frac{1}{n}\sum_{i:y=j} \underline{x_i} && \text{vector of sample means in class } j \\
\hat{\Sigma} &= \frac{1}{n-k}\sum_{j=1}^k \sum_{i:y=j} (\underline{x_i}-\underline{\hat{\mu}_j})(\underline{x_i}-\underline{\hat{\mu}_j})^t && \text{pooled covariance matrix}
\end{aligned}
$$

The class probabilities are obtained when these estimates go into $\delta_j(\underline{x})$ leading to

$$
P(Y=\hat{j}|\underline{X}=\underline{x}) = \frac{e^{\hat{\delta_j}(\underline{x})}}{\sum_{i=1}^k e^{\hat{\delta_i}(\underline{x})}}
$$

Since before we used the logarithm function to define $\hat{\delta_j}$, as written below:

$$\hat{\delta_j}(\underline{x}) = \log(\hat{\pi}_j) - \frac{1}{2}\underline{\hat{\mu}_j}^t\hat{\Sigma}^{-1}\underline{\hat{\mu}_j} + \underline{x}^t\Sigma^{-1}\underline{\hat{\mu}_j}$$

7.3.7 LDA vs logistic regression

Assume you want to compare LDA and logistic regression. Take $k = 2$ (two possible classes); you can then write the log-odds as:

$$\text{log-odds}_{LR} = \log\left(\frac{p(1|x)}{1-p(1|x)}\right) \qquad = \log\left(\frac{p(1|x)}{p(0|x)}\right) = \delta_1(x) - \delta_0(x) \qquad = \beta_0 + \beta_1 x$$

$$\text{log-odds}_{LDA} = \log\left(\frac{p(1|x)}{1-p(1|x)}\right) \qquad = \log\left(\frac{p(1|x)}{p(0|x)}\right) = \delta_1(x) - \delta_0(x) \qquad = \alpha_0 + \alpha_1 x$$

Both of them generate linear expressions

The main difference, assuming that $f_i(\underline{x})$ can be approximated by a multivariate normal, is the estimation of parameters:

- LR: $\underline{\hat{\beta}}$ is estimated directly by Maximum Likelihood
- LDA: $\underline{\hat{\alpha}}$ are estimated indirectly via $\underline{\hat{\mu_j}}$, $\hat{\Sigma}$

Logistic regression optimization procedure is more unstable when the classes are well separated (probability close to 0 and 1).

7.4 Quadratic discriminant analysis

Quadratic discriminant analysis (QDA) is similar to LDA in the sense that it is still defined as a multinormal distribution, but the assumption of homoskedasticity falls; this means that you assume all predictor variables to be normal distributed but each of them can have its own covariance matrix. Importantly, QDA assumes still gaussianity. Therefore we can write QDA as

$$(\underline{X}|Y = j) \sim N_p(\underline{\mu_j}, \Sigma_j)$$

7.4.1 Quadratic boundary

For QDA it can be shown that the decision boundary is quadratic. The calculations shown below prove it

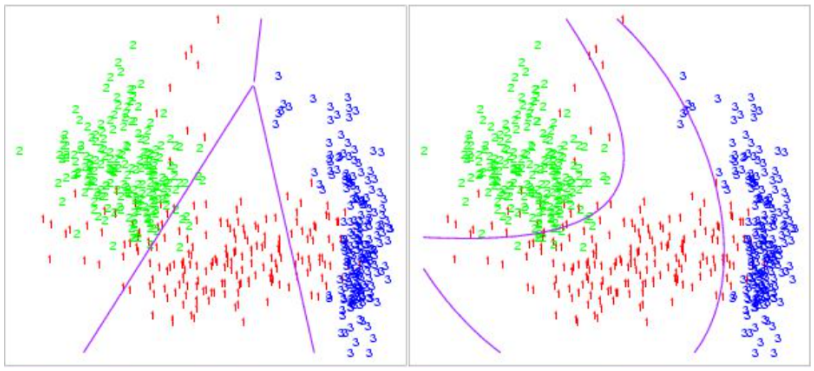
$$\begin{aligned}
P(Y = j|\underline{X} = \underline{x}) &= \frac{\pi_j f_j(\underline{x})}{\sum_{i=1}^k \pi_i f_i(\underline{x})} && \text{from Bayes formula} \\
&= \frac{\pi_j \frac{1}{(2\pi)^{p/2}|\Sigma_j|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma_j^{-1}(\underline{x}-\underline{\mu_j})\right\}}{\sum_{i=1}^k \pi_i \frac{1}{(2\pi)^{p/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma_i^{-1}(\underline{x}-\underline{\mu_j})\right\}} && \text{from the assumptions} \\
&= \frac{\pi_j|\Sigma_j|^{-1/2} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma_j^{-1}(\underline{x}-\underline{\mu_j})\right\}}{\sum_{i=1}^k \pi_i|\Sigma_i|^{-1/2} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma_i^{-1}(\underline{x}-\underline{\mu_j})\right\}} && \text{simplifying}
\end{aligned}$$

Then assign observation $\underline{x}$ to class j which maximises the logarithm of the numerator (since the denominator does not depend on j).

$$\begin{aligned}
\delta_i(\underline{x}) &= \log(\pi_j) - \frac{1}{2}\log(|\Sigma_j|) - \frac{1}{2}(\underline{x}-\underline{\mu_j})^t\Sigma_j^{-1}(\underline{x}-\underline{\mu_j}) && \text{function to maximise} \\
&= \log(\pi_j) - \frac{1}{2}\log(|\Sigma_j|) - \frac{1}{2}\underline{x}^t\Sigma_j^{-1}\underline{x} + \underline{x}^t\Sigma_j^{-1}\underline{\mu_j} - \frac{1}{2}\underline{\mu_j}^t\Sigma_j^{-1}\underline{\mu_j} && \text{multiplying} \\
&= \overbrace{-\frac{1}{2}\underline{x}^t\Sigma_j^{-1}\underline{x}}^{\text{quadratic term}} + \overbrace{\underline{x}^t\Sigma_j^{-1}\underline{\mu_j}}^{\text{linear term}} + \overbrace{\log(\pi_j) - \frac{1}{2}\log(|\Sigma_j|)}^{\text{intercept}} && \text{which is quadratic in } \underline{x}
\end{aligned}$$

7.4.2 LDA vs QDA

Figure 7.4: QDA vs LDA



Obviously, lines generated through QDA are always curved, contrarily to LDA; then, they are more prone to overfit the data. QDA has more parameters than LDA, in fact in the first case the number of parameters is kxp parameters

namely

$$\begin{aligned}
&\text{QDA: } pk + k\frac{p(p+1)}{2} && \underline{\mu_1}, \ldots, \underline{\mu_k}, \Sigma_1, \ldots, \Sigma_k \\
&\text{LDA: } pk + \frac{p(p+1)}{2} && \underline{\mu_1}, \ldots, \underline{\mu_k}, \Sigma_{p\times p}
\end{aligned}$$

For example, if we take $k = 2$ and $p = 50$, then LDA has 1375 parameters, while QDA has 2650. Because of the greater amount of parameters, the variability/balance equilibrium has to be evaluated. In case of equivalent results, it is important to take the simpler model.