Statistical learning - Vinciotti (2022)

Stefano Cretti

telegram: @StefanoCretti

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

March 10, 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 $\underline{X}$)

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 (mean squared error):

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

Furthermore, since we know that:

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

Thus we get:

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

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** $(\underline{x}_i, y_i)$, $i=1,\dots,n$, where $\underline{x}_i=(x_{i1},\dots,x_{ip})^t$ 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:

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

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**.
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 the former 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:

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

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:

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

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 refers to the error that is introduced by approximating a problem, which may be extremely complicated, by a much simpler model. If an estimated model performs well on the training data but it does not perform well on new data, the estimated model has high bias. 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 more flexible models are generally more influenced by a change of dataset.
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.

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

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.

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).

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 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 $f(\underline{X})$. Many loss functions can be used; the choice of a specific loss function determines which function is considered to be the true rule $f(\underline{X})$.

3.3 Regression setting

In a regression setting, the loss function which is generally used is the **squared error loss function**, 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:

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

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.

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, since the law of iterated expectations states that $E[X] = E[E[X|Y]]$ (analogously to profile likelihood, you fix one variable and variate the other), 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

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

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 EPE is

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

□

In a regression setting you could use other loss functions, such as

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

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

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

  or focussing more on the conditional mean

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

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

3.3.1 K nearest neighbour for regression

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

$$\hat{f}(\underline{x}) = \text{ Average } (y_i|\underline{x}_i \in N_K(\underline{x}))$$

Where:

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

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.

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

$$L(Y, \hat{Y}(\underline{X})) = \text{TODO Add punnet square}$$

Where:

- Y is the true class
- $\hat{Y}(\underline{X}$ 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 $\hat{Y}$, which is the function of $\underline{X}$ such that it minimizes the **expected 0-1 loss**, defined as:

$$E[L(Y, \hat{Y}(\underline{X}))]$$

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

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

(Which is the weighted sum of $\underline{x}$ belonging to each possible class). Then, considering the binary case, if the classifier predicts $\underline{x}$ to belong to class 0 ($\hat{y} = 0$), we have

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

On the other hand, if the classifier predicts $\underline{x}$ to belong to to class 1 ($\hat{y} = 1$), 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 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:

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

We talk about **perfect separation** of the classes 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. More often than not, 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

$$L(Y, \hat{Y}(\underline{x})) = \text{TODO add punnet square}$$

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; using a 0-1 loss function results in a model which always classifies patients as healty, therefore it is 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

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

which is the classification threshold that takes into account different error weights.

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

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

Meanwhile, for a classification, you create a **confusion matrix**, estimate the probabilities via some method, substitute the values into the following formulas

$$\begin{aligned} \text{True positive rate (TPR)} &= \frac{\text{TP}}{\text{TP } + \text{ FN}} && = \text{ sensitivity} \\ \text{False positive rate (FPR)} &= \frac{\text{FP}}{\text{FN } + \text{ FP}} && = 1 - \frac{\text{TN}}{\text{TN } + \text{ FP}} = 1 - \text{ specificity} \\ \text{Error rate} &= \frac{\text{FP } + \text{ FN}}{m} \end{aligned}$$

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).

Different models have different ROC curves. The best possible curve would touch the top left corner, 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 in 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 an 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:

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

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). Repeat for each class.

4.1.1 Bias-Variance trade off in K nearest neighbour classification

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).

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:

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

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; therefore

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

which corresponds to the following probability mass function

$$f(y) = p^y(1-p)^{1-y},\ y = 0, 1$$

which has for expected value

$$E[Y] = 0 \cdot (1-p) + 1 \cdot p = p$$

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 go from 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 $\mathbb{R}$.

Therefore we must find a way to normalize the regression in the range $(0, 1)$.

4.2.2 Characteristics of the logistic regression

Logistic regression solves the problem of normalizing the regression in a range $(0, 1)$ 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

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

and it has the following properties:

- Its codomain corresponds to the interval $(0, 1)$
- It has sigmoid shape with intercept in $y = 0.5$
- In the binomial case, if $\beta 1 > 0$ then the function is increasing, else it is decreasing.

The g function is the inverse of the conversion from linear regression to logistic function.
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.

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

The predictor for this measure is called **log-odds** or **logit** (it is the g function from before) 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 formal definition of a logistic regression model (assuming the random variable to be bernoulli distributed) then becomes:

$$\begin{gathered}
Y|\underline{X} = \underline{x} \sim Bernoulli(p(1|\underline{x}))\\
\log\left(\frac{p(1|\underline{x})}{1-p(1|\underline{x})}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p
\end{gathered}$$

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 it induces a multiple factor of e^{β_j} for the odds).

4.2.3 Example of logistic regression

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

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 logaritms 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** (the exponential of the previous function), 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.

4.2.4 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, contrarely 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 $\beta_0, \beta_1, \ldots, \beta_p$. To do so we need to compute the maximum likelihood starting from the fact that Y is binary, thus

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

The likelihood conditional to the observations hence is:

$$L(\underline{\beta}) = \prod_{i=1}^{n} f(y_i|\underline{x}_i) = \prod_{i=1}^{n} p(1|\underline{x}_i)^{y_i}(1-p(1|\underline{x}_i))^{1-y_i}$$

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):

$$
\begin{aligned}
\frac{\partial l(\beta)}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j}\left(\sum_{i=1}^{n}\left[y_i \underline{\beta}^t \underline{x}_i - \log\left(1+e^{\underline{\beta}^t \underline{x}_i}\right)\right]\right) \quad j = 0, \ldots, p\\
&\text{but } \frac{\partial \underline{\beta}^t \underline{x}}{\partial \beta_j} = \frac{\partial}{\partial \beta_j}(\beta_0 x_{i0} + \cdots + \beta_p x_{ip}) = x_{ij} \text{ since the other terms simplify}\\
&= \sum_{i=1}^{n}\left[y_i x_{ij} - \frac{e^{\underline{\beta}^t \underline{x}_i}}{1+e^{\underline{\beta}^t \underline{x}_i}} x_{ij}\right]\\
&\text{but } \frac{e^{\underline{\beta}^t \underline{x}_i}}{1+e^{\underline{\beta}^t \underline{x}_i}} = p(1|\underline{x}) \text{ therefore}\\
&= \sum_{i=1}^{n} [y_i x_{ij} - p(1|\underline{x}_i) x_{ij}]
\end{aligned}
$$

By setting this derivative to zero you obtain $p+1$ equations in $p+1$ 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

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