Data Analysis

Miriam Amendola

January 4, 2024

Contents

Chapter 1

Fundamentals

1.1 Linear Algebra

Matrix dot product: Assume we have $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, then $C = AB \in \mathbb{R}^{m \times p}$ is defined as:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Orthogonality, we say that the matrix U is **orthogonal** if $U^T U = I$.

Similarity, we say that two matrices B and C are **similar** if there exists a matrix P such that $B = P^{-1} C P$. When two matrices are similar they have the same eigenvalues.

Eigen-decomposition: From the **spectral theorem** we know that if A is a real square symmetric matrix then it can be decomposed as $A = U \Lambda U^T$ where U is a matrix obtained by stacking the eigenvectors of A while Λ is a diagonal matrix with the eigenvalues of A. In other words, if A is a real square symmetric matrix then it is **similar** to the diagonal matrix Λ.

In general, the relationship between the eigenvectors and the eigenvalues is the following: $Au^{(k)} = \lambda_k u^{(k)}$ where u is an eigenvector and λ is the corresponding eigenvalue.

1.2 Law of Large Numbers

1.3 Bayes's Rule

1.4 Theorem of the total expectation

1.5 Tower Property

Chapter 2

Estimation Theory

2.1 Introduction to Data Analysis

2.1.1 Data Analysis

Data Analysis is the process of extracting information from data. Although data and information are often used interchangeably, they are not the same. Data is something that should cointain information, but it is not information itself. In order to extract information from data, we need to build a learning system.

An important thing to know is that when the learning system is *good*, then by increasing the amount of data we have, the available information cannot decrease. In practice, most of the times we do not have a good system, so it can happen that the data fed to the system is misleading and the information we get from the analysis decreases.

The two main families of problems in data analysis are *estimation* and *classification*. The main difference is that in the estimation problem, we have a set of data and we want to estimate a real value, while in the classi- fication problem the output is contained in a finite set.

The input of the learning system can be anything (a vector, a matrix, a graph, a sequence, etc.), but the output is always a real number or a finite set of values. In general, we do not care about the dimension of the output, because we can always repeat the problem as many times as the dimension of the output.

Another thing to make clear is that in statistical learning, we do not have a temporal correlation between the variables. When we say that two variables depend on each other, we just mean that they are correlated, without implying the causality.

2.2 Bayes Estimation

Assume we have a random variable Y and we want to approximate it with a single deterministic value. To do that, we need to find the number that, on

average, is the closest to the values of the random variable. In other words, we want to find the number z^* that minimizes the expected value of the squared *distance* between Y and z:

$$z^* : \arg\min_{z \in \mathbb{R}} \mathbb{E}\left[(Y-z)^2\right]$$

The quantity we want to minimize is called *mean squared error* (MSE) and in this case is a function of the value z. We can find the minimum of this function by computing its derivative and setting it to zero:

2.3 Maximum Likelihood Estimation

2.4 Exercises

2.4.1 Maximum Likelihood

2.4.2 MMSE

Exercise 1. We have a random variable $Y \sim N(0, \sigma_y^2)$ and data

$$X_i = Y_i + W_i \qquad i = 1, \ldots, n$$

where variables W_i are *i.i.d* and also independent with respect to Y and distributed according to $W \sim N(0, \sigma_w^2)$.

Compute the *minimum mean square error estimator*.

The MMSE is the posterior mean of Y given X:

$$\hat{Y} = \mathbb{E}\left[Y|X\right]$$

To compute the posterior mean, we need to compute the posterior distribution of Y given X. We can do that either by definition or by applying Bayes' Theorem:

$$f(y|x) = \frac{\pi(y)\ell(x|y)}{p(x)}$$

We can infer that:

$$f(x_i|y) = \ell(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{\frac{-(x-y)^2}{2\sigma_w^2}}$$

and consequently

$$\ell(x|y) = \prod_{i=1}^{N} \ell(x_i, y)$$

While we can say that given $Y = y$ then X_i is distributed according to W_i that is shifted by an amount of y, because $X_i = Y_i + W_i$ and $f(w|y) = f(w)$, we cannot compute $f(y|x)$ observing that $Y_i = X_i - W_i$ because X_i and W_i are not independent on each other but they are dependent because $X_i = Y_i + W_i$. Thus we need to apply Bayes' Theorem to find out $\ell(y|x)$.

Observe that since we are computing x *given* y, then we're actually trying to understand the **generative mechanism** that produces the data x given the true value y. Suppose for example that we're sampling measurements x of the temperature in a room. We're actually measuring the true value of the temperature and some noise w added to it. We can try to estimate the generative mechanism to understand what is the distribution of temperature given the true value.

Recall that, in the Bayesian setting, $f(y|x)$ is called *posterior* function and $\pi(y)$ is called the *prior* function. The posterior function does not tell us what is the generative mechanism of the data but it estimates how y is hidden from the data. Also, we will observe that the **MMSE** estimator is obtained by combination of likelihood and prior information.

In order to compute $f(y|x)$ we can apply the Bayesian Theorem:

$$f(y|x) = \frac{\pi(y)f(x|y)}{p(x)}$$

Observe that since $f(y|x)$ is a probability density function with respect to y, the term $p(x)$ must be a constant with respect to y. We can express $p(x)$ as the joint or *marginal* distribution of x and y because

$$p(x) = \int \pi(y')\ell(x|y')dy$$

Because if we apply to the previous expression the integral to both members, we obtain:

$$\int f(y|x) = \frac{\int \pi(y)\ell(x|y)dy}{p(x)} = 1???$$

So, given that $p(x)$ is a constant with respect to y, we arrive at the fundamental proposition that:

$$f(y|x) \propto \pi(y)\ell(x|y)$$

It is propotional because I divide by $p(x)$ that is a factor that is completely determined by y. If $p(x)$ is not a constant given y then $f(y|x)$ is not a probability density function.

Some remarks:

- In this part of the course we're assuming that we already know the model, so $\pi(y)$ and $\ell(x|y)$ are known.
- If our data follows the model, then no other algorithm (not even deep learning) can outperform the estimators derived by our models
- We do not create a generative mechanism but we "pretend" to know it.

To do a practical example, we can consider a weather forecasting system, with input data that is temperature, humidity and pressure and output data that is if tomorrow will be sunny or rainy. The *posterior* function (likelihood function) tells us if given that is sunny or rainy what could be the probabile values of the sensor data. The *prior* function tells us on average if it is rainy or sunny.

Now we're going to calculate $f(y|x)$. We've said that apart from a constant $p(x)$, that is constant with respect to y but a function of x, our *posterior* function is:

$$f(y|x) \propto \pi(y)\ell(x|y)$$

We know that:

$$\ell(x|y) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{\frac{-(x_i-y)^2}{2\sigma_w^2}}$$

and that

$$\pi(y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{y^2}{2\sigma_y^2}}$$

So we can write:

$$f(y|x) \propto \pi(y)\ell(x|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \frac{1}{(2\pi\sigma_w^2)^{\frac{N}{2}}} e^{-\frac{y^2}{2\sigma_y^2}} e^{-\frac{1}{2\sigma_w^2}\sum_{i=1}^{N}(x_i-y)^2}$$

We can ignore the constants:

$$f(y|x) \propto e^{-\frac{y^2}{2\sigma_y^2} - \frac{1}{2\sigma_w^2}\sum_{i=1}^{N} x_i^2 - \frac{N}{2}\frac{y^2}{2\sigma_w^2} + \frac{y}{\sigma_w^2}\sum_{i=1}^{N} x_i}$$

Since the second term of the exponential is costant with respect to y and it can be written as a product, it can be ignored because we're considering the fact that is proportional. After ignoring the constant, we can observe that in the following expression the first term depends on the *prior* function while the second term depends on the *posterior* function.

$$f(y|x) \propto e^{-\frac{y^2}{2\sigma_y^2} - \frac{N}{2}\frac{y^2}{2\sigma_w^2} + \frac{y}{\sigma_w^2}\sum_{i=1}^{N} x_i}$$

We can factor the terms that depend on y^2:

$$f(y|x) \propto e^{-\frac{y^2}{2}\left(\frac{1}{\sigma_y^2} + \frac{1}{\frac{\sigma_w^2}{N}}\right) + \frac{y}{\sigma_w^2}\sum_{i=1}^{N} x_i}$$

and then by defining:

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_y^2} + \frac{1}{\frac{\sigma_w^2}{N}}$$

We get:

$$f(y|x) \propto e^{-\frac{1}{2\sigma^2}y^2 + \frac{\sum x_i}{\sigma_w^2}y}$$

Now we try to complete the square by adding and subtracting a function $g(x)$:

$$f(y|x) \propto e^{\frac{-(y-\mu)^2}{2\sigma^2}}$$

where:

$$\mu = \frac{\sigma^2}{\sigma_w^2}\sum_{i=1}^{N} x_i$$

So we obtained that the posterior mean is μ:

$$\hat{Y} = \mathbb{E}[Y|X] = \frac{\sigma_y^2}{N\sigma_y^2 + \sigma_w^2}\sum_{i=1}^{N} x_i$$

Chapter 3

Regression

3.1 Model Based Regression

In this section we will study model-based regression, which means the models of the data and the error are given and we are only interested in finding the *regression function*, also called *optimal estimator*.

Suppose we have $Y \in \mathbb{R}$ (that is our parameter of interest) and data $X \in \mathbb{R}^d$, both random variables.

Definition 1. The optimal regression function is the mean of the posterior distribution of the target variable given the data, i.e. the **MMSE**:

$$r(X) = \mathbb{E}[Y \mid X]$$

Exercise 2. Compute the regression function for the multiple linear regression model:

$$Y = \beta_0 + \sum_{i=1}^{d} \beta_i X_i + \mathcal{E}$$

where Y is the response variable, β_0 is the intercept term, β_i are the coefficients of the regressors X_i and $\mathcal{E}$ is the error term.

The common assumption we do in multiple linear regression is that $\mathcal{E}$ is a zero mean random variable with variance σ^2.

Let us rewrite the model in matrix form:

$$Y = \beta^T X + \mathcal{E}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_d)^T$ is the *parameters* vector and $X = (1, X_1, \ldots, X_d)^T$ is the *predictors* vector.[1]

Now we can compute the regression function:

$$r(X) = \mathbb{E}[Y \mid X] = \mathbb{E}\left[\beta^T X + \mathcal{E} \mid X\right]$$

[1] We can write this both as $\beta^T X$ or $X^T \beta$ because the final product is a scalar thus the order of the product doesn't matter, but we will use the first notation.

By applying the linearity properties of the expectation:

$$r(X) = \beta^T \mathbb{E}[X \mid X] + \mathbb{E}[\mathcal{E} \mid X]$$

The first expectation $\mathbb{E}[X \mid X]$ is X because we are conditioning on X and X is a constant given X, while the second expectation $\mathbb{E}[\mathcal{E} \mid X]$ is zero because in this setting we need to assume that $\mathcal{E}$ is conditionally zero mean given X.

In conclusion, we found that the optimal regression function for the multiple linear regression model is:

$$r(X) = \beta^T X$$

Question

Why is the error term conditionally zero mean given X?

Exercise 3. Find the relationship between β and X and Y. In particular, we want to find an expression for β that depends on some moments of X, Y and XY.

Assume we are working with the multiple linear regression model:

$$Y = X^T \beta + \mathcal{E}$$

where $Y \in \mathbb{R}$, $X \in \mathbb{R}^{(d+1)\times 1}$, $\beta \in \mathbb{R}^{(d+1)\times 1}$ and $\mathcal{E} \in \mathbb{R}$.

First we multiply on the left both members by X:

$$\underset{(d+1)\times 1}{XY} = \underset{(d+1)\times 1}{XX^T\beta} + \underset{(d+1)\times 1}{X\mathcal{E}}$$

Then we take expectation:

$$\mathbb{E}[XY] = \mathbb{E}\left[XX^T\right]\beta + \mathbb{E}[X\varepsilon]$$

And we compute each term separately.

For the first term, we can observe that the parameter vector is a constant with respect to X so we can take it out of the expectation. To compute the expectation of the matrix $X^T X$, let's consider a generic entry:

$$[XX^T]_{ij} = X_i X_j \to \mathbb{E}\left[[XX^T]_{ij}\right] = \mathbb{E}[X_i X_j]$$

Since the expected value between $X_i X_j$ is a correlation, XX^T is a quantity that describes the correlation between the data.

Definition 2. The matrix $R_X \triangleq \mathbb{E}\left[XX^T\right]$ is the **(auto)-correlation matrix**.

The same reasoning can be applied to $\mathbb{E}[XY]$, where we get a corretlation between the data and the response variabile.

Definition 3. The matrix $R_{XY} \triangleq \mathbb{E}[XY]$ is the **(cross)-correlation matrix**.

As for the second term, since X and $\mathcal{E}$ are not independent, we apply the tower property:

$$\mathbb{E}[X\mathcal{E}] = \mathbb{E}_X[\mathbb{E}[X\mathcal{E}|X]] = \mathbb{E}_X[X\mathbb{E}[\mathcal{E}|X]] = 0$$

since in the inner expectation X is a constant, it can be taken out of the expectation and since we assumed that the error term is zero mean given X, we find out that the whole expectation is zero.

We can rewrite the equation 3 as:

$$R_{XY} = R_X\beta$$

And then solve for β:[2]

$$\beta = R_X^{-1}R_{XY}$$

In conclusion we found out that in a model-based linear regression problem β cannot be arbitrary, because it is uniquely determined by the correlation matrices of X and XY.

3.1.1 Connection between model-based and supervised

What is changing with respect to the other part of the course?

In the other part of the course we've worked in a **supervised** setting, meaning that we had a training set. We still had a model because we *assumed* that the relationship between Y and X was linear, but we wanted to learn the parameters β of the model from the data, making it a *parametric regression* problem.

Recall that the prediction phase it's our most important goal and it is the goal we would have if we knew the model. Since we don't have the model, in the *parametric* setting, we need to assume a model and learn the parameters β. In that case, learning β is instrumental to *predict* Y.

In our case, we are in a model-based setting, which means we do not have a training set. As we showed in the previous exercise, in our case, the parameters β are not to be learned from the data, but they are uniquely determined by the correlation matrices of X and XY.

Now the question is ***how can we learn the model***?

In a supervised approach we use a **training set** $\tilde{X}$ (not a *dataset*).

$$\tilde{X} = \begin{bmatrix} \tilde{X}_{11} & \dots & \tilde{X}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{X}_{(d+1)1} & \dots & \tilde{X}_{(d+1)n} \end{bmatrix}$$

That we can write compactly as:

$$\tilde{X} = \left[\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_n\right]$$

Where each $\tilde{X}$ is a vector with dimensions $(d+1) \times 1$.

And the corrisponding values of the response variable, $\tilde{Y}$:

$$\tilde{Y} = \left[\tilde{Y}_1, \tilde{Y}_2, \dots, \tilde{Y}_n\right]$$

[2] By assuming that R_x is invertible.

Exercise 4. Propose an estimator $\hat{\beta}$ based on the training set $(\tilde{X}, \tilde{Y})$, knowing that $\beta = R_x^{-1} R_{xy}$ and $Y = \beta^T X$.

We can find the estimator by replacing the correlation matrices with their empirical counterparts:

$$\hat{\beta} = \left(\hat{R_X}\right)^{-1} \hat{R}_{xy}$$

where the correlation matrices are defined as:

$$[\hat{R}_X]_{ij} = \frac{1}{n} \sum_{m=1}^{n} \tilde{X}_{im} \tilde{X}_{jm} \qquad [\hat{R}_{XY}]_i = \frac{1}{n} \sum_{m=1}^{n} \tilde{X}_{im} \tilde{Y}_m$$

We can rewrite the estimators by using the matrix notation:

$$[\hat{R}_X]_{ij} = \frac{1}{n} \left[\tilde{X} \tilde{X}^T\right]_{ij} \qquad [\hat{R}_{XY}]_i = \frac{1}{n} \left[\tilde{X} \tilde{Y}^T\right]_i$$

By replacing the correlation matrices in the estimator and simplyfing n we get:

$$\hat{\beta} = \left(\hat{R}_X\right)^{-1} \hat{R}_{XY} = \left(\tilde{X} \tilde{X}^T\right)^{-1} \tilde{X} \tilde{Y}^T$$

The **empirical estimator** of β we found is the same as the one we obtained in the supervised approach. In that case, we minimized the RSS, which, if divided by n, is the empirical risk.

$$RSS = \frac{1}{n} \sum_{m=1}^{n} \left(\tilde{Y}_m - \beta^T \tilde{X}_m\right)^2 \xrightarrow{n \to +\infty} \mathbb{E}\left[(Y - \beta^T X)^2\right]$$

> **Warning**
>
> Sentire la registrazione da qui in poi.

The two things are connected only in this case.

We took the optimal risk (that is the expectation) and we replaced it with the empirical risk. I am trying to do Bayes empirically. The difference is that we need a model. In MMSE we have a **general solution** that works for every model, in the **supervised approach** we need to impose or assume a model.

Optimal Bayes is a benchmark performance. There is some optimal thing to do, we cannot do it. I impose one model and replace the *optimal risk* with the *empirical risk*.

3.1.2 Benchmark performance

Both in the model-based and supervised case, the final goal of the supervised regression is to make a prediction.

In model-based regression we want to find the expression of Y which allows us to compute Y knowing the distribution of X and the parameters. In supervised regression, we want to make a prediction by exploiting the information contained inside the training set pairs, which must reveal the link between X and Y.

Assume we have a training set $T_n = \{(X_i, Y_i)\}_{i=1}^n$ and we want to make a prediction for a new observation (X_0, Y_0). Making a prediction means that we want to predict Y_0, that is not observed, given the observation X_0. It is important to remark that we are working under the assumption that X_0 and Y_0 share the same distribution of X_i and Y_i.

In the model-based approach, the optimal estimator is the regression function $r(X_0)$, which is given by the MMSE. In the supervised approach, the estimator is the regression function $\hat{r}(X_0)$, which is given by the minimization of the empirical risk. This function will be a **suboptimal** estimator, meaning that it will not reach the performance of the MMSE.

Theorem 1. The error of the suboptimal regression function $\hat{r}(X_0)$ is given by:

$$\mathbb{E}\left[(\hat{r}(X_0) - Y_0)^2\right] = \text{MMSE} + \mathbb{E}\left[(\hat{r}(X_0) - r(X_0))\right]$$

where $\text{MMSE} = \mathbb{E}\left[(r(X_0) - Y_0)^2\right]$ is the error of the optimal estimator.

To prove this result, we first need to introduce the **orthogonality principle**.

Theorem 2. The **orthogonality principle** states that the error of the optimal estimator is orthogonal to any (integrable) function of the data $g(X_0)$:

$$\mathbb{E}\left[g(X_0)\left(r(X_0) - Y_0\right)\right] = 0$$

Proof. To prove the theorem 1, we need to compare the error of the suboptimal estimator with the error of the optimal estimator. The error of $\hat{r}(X_0)$ is given by:

$$\mathbb{E}\left[(\hat{r}(X_0) - Y_0)^2\right]$$

In order to compare it with the optimal estimator, we need to add and subtract $r(X_0)$ from the error term:

$$\mathbb{E}\left[(\hat{r}(X_0) - Y_0)^2\right] = \mathbb{E}\left[([\hat{r}(X_0) - r(X_0)] + [r(X_0) - Y_0])^2\right]$$

By expanding the square we get:

$$\mathbb{E}\left[([\hat{r}(X_0) - r(X_0)]^2 + [r(X_0) - Y_0]^2 - 2[\hat{r}(X_0) - r(X_0)][r(X_0) - Y_0])^2\right]$$

And applying the linearity property:

$$\mathbb{E}\left[[\hat{r}(X_0) - r(X_0)]^2\right] + \mathbb{E}\left[[r(X_0) - Y_0]^2\right] - 2\mathbb{E}\left[[\hat{r}(X_0) - r(X_0)][r(X_0) - Y_0]\right]$$

The first term quantifies how much our regression function is deviated from the optimal regression function, the second term is the MMSE, the third term need further discussion.

Since in the expectation there are two random variables X_0 and Y_0 and they are dependent, we cannot split the expectation. In order to get X_0 fixed, we would need to have a conditional expectation.

We apply the *tower property*:

$$\mathbb{E}\left[[\hat{r}(X_0) - r(X_0)][r(X_0) - Y_0]\right] = \mathbb{E}_{X_0}\left[\mathbb{E}_{Y_0}\left[\underbrace{[\hat{r}(X_0) - r(X_0)]}_{g(X_0)}[r(X_0 - Y_0)] \mid X_0\right]\right] =$$

$$= \mathbb{E}_{X_0}\left[\mathbb{E}_{Y_0}\left[g(X_0)\left(r(X_0) - \underbrace{\mathbb{E}\left[Y_0 \mid X_0\right]}_{r(X_0)}\right)\right]\right] = 0$$

The last equality is true because of the **orthogonality principle**.

So we can rewrite the error of the $\hat{r}(X_0)$ as:

$$\boxed{\mathbb{E}\left[(\hat{r}(X_0) - Y_0)^2\right] = \text{MMSE} + \mathbb{E}\left[(\hat{r}(X_0) - r(X_0))\right]}$$

□

In conclusion, we can say that the error of the suboptimal estimator is the sum of the error of the optimal estimator and a **penalty term**, which is the squared difference between the arbitrary function and the optimal estimator. If our arbitrary function is the optimal estimator, the penalty term is zero, but if it differs, the penalty term increases.

This result has been very useful to discover *non-parametric approaches*. Before this result the non-parametric approaches tried to approximate the optimal regression function, but now we can try to find an arbitrary function that minimizes the penalty term. The only problem is that we don't know the optimal regression function, so we cannot compute easily the penalty term. As for the MMSE, it can be approximated by simulation.

3.2 Supervised Parametric Regression

3.2.1 Simple Linear Regression

Let's consider a simple linear regression model:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

where $\mathbb{E}[\varepsilon] = 0$. Assume we want to estimate the parameters β_0 and β_1 and find the estimated model:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

Suppose we have a training set $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^n$. We can write:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i \qquad i = 1, \ldots, n$$

Ordinary Least Squares The Ordinary Least Squares method let us estimate the parameters $\hat{\beta}_0$, $\hat{\beta}_1$ by minimizing the residual sum of squares, which is defined as following:

$$\text{RSS}(\beta_0, \beta_1) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

Under the assumption that the error term $\underline{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$:

- $\mathbb{E}[\varepsilon_i] = 0, \forall i$
- $\text{Var}[\underline{\varepsilon}] = \sigma^2 I_n$ (homoscedasticity)

The **least square estimates** of the parameters are:

$$\underline{\beta}^T = \left(\hat{\beta}_0, \hat{\beta}_1\right) = \arg\min_{(\beta_0, \beta_1)} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

If the error terms are independent and identically distributed as normals $\varepsilon_i \sim N(0, \sigma^2)$, then the *least squares estimates* are equal to the ones we obtain by *maximum likelihood estimation.*

Maximum Likelihood Estimation for linear regression Now we want to find the maximum likelihood estimates for the parameters $\hat{\beta}_0$ and $\hat{\beta}_1$. In order to solve the maximization problem, we need to compute the gradient with respect to the parameters $\underline{\beta}$ and set it to zero:

$$\nabla_{\underline{\beta}} RSS(\beta_0, \beta_1) = 0$$

$$\begin{cases} \frac{\partial RSS}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0 \\ \frac{\partial RSS}{\partial \beta_1} = -2 \sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i) = 0 \end{cases}$$

$$\begin{cases} n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

Let's define the following quantities:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

From the first equation we can find β_0:

$$\beta_0 = \overline{y} - \beta_1 \overline{x}$$

Substituting this value into the second equation we can find β_1:

$$\overline{y} \sum_{i=1}^{n} x_i - \beta_1 \overline{x} \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \leftrightarrow \beta_1 = \frac{\sum_{i=1}^{n} x_i y_i - \overline{y} \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i^2 - \overline{x} \sum_{i=1}^{n} x_i}$$

The least squares estimates are the following:

$$\begin{cases} \hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \\ \hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \end{cases}$$

The **Gauss-Markov** theorem states that the least squares estimates are the **best linear unbiased estimates** (BLUE) of the parameters β_0 and β_1.

This means that the least squares estimates are unbiased and have the *same* smallest variance:

$$\sigma^2 = \mathrm{RSE}^2 = \frac{\mathrm{RSS}(\hat{\beta}_0, \hat{\beta}_1)}{n-2}$$

By assuming that the errors ε_i are normally distributed, we can compute the confidence intervals for the parameters β_0 and β_1: Let's call B_0 and B_1 the random variables that represent the estimates of β_0 and β_1 respectively. Since we know the distribution of the errors ε_i, we can compute the distribution of B_0 and B_1:

$$B_0 \sim N(\beta_0, SE^2(B_0))$$

$$B_1 \sim N(\beta_1, SE^2(B_1))$$

The following quantity:

$$\frac{B_1 - \beta_1}{SE(B_1)} \sim t_{n-1}$$

is called **t-statistic** and it is used to compute the confidence interval for β_1. It is distributed a (Student) t-distribution with $n-2$ degrees of freedom.

3.2.2 Assesing the accuracy of the model

In order to understand if the inffered model is good enough, we should use an error metric. If we're working with a continuous target variable, we can use the *RSE* or **residual standard error**. The residual standard error measures how much of the variability is not explained by our model.

However, the residual standard error has the limitation that it depends on scale by the order of Y. If Y is a large number, then the *RSE* will also be a large number.

This means that it is not always clear what constitutes a good RSE. An alternative measure of fit is the $\mathbf{R^2}$ **statistic**.

This statistic represents the proportion of variance explained and it is independent of the scale of Y. The definition of *R-squared* is the following:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

Where **TSS** is the *total sum of squares* and it is defined as following:

$$TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2$$

This quantity measures the total variance in the response Y and represents the amount of variability inherent in the response before the regression is performed. In contrast, RSS measures the amount of variability that is left unexplained after performing the regression.

The difference between this two quantities measures the amount of variability in the response that is explained or removed by performing the regression and R^2 measures the proportion of variability in Y that can be explained using X. When the R^2 statistic is close to 1 it means that a large proportion of the variability in the response has been explained by the regression. When the R^2 statistic is close to 0 it means that the regression did not explain much of the variability in the response and this might occur because the linear model is wrong or the inherent error is high or both.

In the simple regression setting:

$$R^2 = r^2$$

Where r is the Pearson's correlation index between X and Y:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

The correlation index is a measure of the linear relationship between X and Y. The R^2 statistic extends the concept of correlation between multiple predictors and the response.

3.2.3 Multiple Linear Regression

Suppose we have an input vector $X^T = (X_1, \ldots, X_p)$ and we want to predict an output Y. The linear regression model has the form:

$$Y = f(X) + \varepsilon = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \varepsilon$$

We can interpret the model as the following: β_j is the average increase in Y when X_j increases by one unit holding all others constants.

Note that despite being called **linear regression**, X_j can come from different sources, such as:

- transformations of the input function, e.g. log
- polynomial fit, e.g. $X_2 = X_1^2$
- dummy coding of the levels of qualitative inputs
- interaction between variables $X_3 = X_1 \cdot X_2$

Least squares estimates

Suppose we have an input vector $X^T = (X_1, \ldots, X_p)$ and we want to predict an output Y. We need a regresison model, for example:

$$Y = f(X) + \varepsilon$$

Under the linear assumption, we have:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \varepsilon$$

Where $\underline{X} = (X_1, \ldots, X_P)^T$ is the input vector and $\underline{\beta}$ is the parameter vector that we want to estimate. Given the training data $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^n$ we pick $\underline{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ to minimize the residual sum of squares.

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Since we can write our linear regression model in matrix form:

$$\underline{Y} = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 x_{11} x_{12} \ldots x_{1p} \\ 1 \ldots \\ \vdots \\ 1 x_{n1} x_{n2} \ldots x_{np} \end{bmatrix}$$

da finire

$$\underline{Y}\,\underline{\underline{X}}\beta + \underline{\varepsilon}$$

So the estimated response on the traning set will be:

$$\hat{Y} = \underline{\underline{X}}\hat{\beta}$$

Now we can rewrite the formula of the *RSS* in matrix form:

$$\mathrm{RSS}(\underline{\beta}) = \sum_{i=1}^{n} \left(y_i - (\underline{\underline{X}}\beta)_i \right)^2 = (\underline{Y} - \underline{\underline{X}}\beta)^T (\underline{Y} - \underline{\underline{X}}\beta)$$

Now we need to compute the gradient of the residual sum of squares with respect to the parameters. Let's recall first the following rules:

1. $\nabla_x a^t x = \nabla_x x^T a = a$
2. $\nabla_x x^T A x = A^t x + A x$

The gradient of the RSS will be:

$$\begin{aligned} \nabla_\beta \mathrm{RSS}(\beta) &= \nabla_\beta \left((Y - X\beta)^T (Y - X\beta) \right) = \\ &= \nabla_\beta \left(Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta \right) = \\ &= \nabla_\beta \left(Y^T Y - 2\beta^T X^T Y + \beta^T X^T X\beta \right) = \\ &= -2X^T Y + (X^T X)^T \beta + (X^T X)^T \beta = \\ &= X^T Y + (X^T X)^T \beta = 0 \Leftrightarrow \hat{\beta} = (X^T X)^{-1} X^T Y \end{aligned}$$

The *least squares estimation* of the parameters are

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Note that in this computation we also found that:

$$X^TY + (X^TX)^T\beta = X^T(Y - X\hat{\beta}) = 0 \Rightarrow e \perp X$$

To confirm that this solution is actually a point of minimum, we need to check the **hessian matrix**, defined as following:

$$H^(_{RSS}\beta) = \frac{\partial^2 RSS(\beta)}{\partial\beta\partial\beta^T} = 2X^TX$$

Since this matrix is positively defined, then the estimator is always a point of minimum.

Now we can substitute the parameters into $\hat{Y}$

$$\underline{\hat{Y}} = \underline{X\hat{\beta}} = \underbrace{\underline{X}(X^TX)^{-1}X^T}_{H}Y = HY$$

We've found that $\hat{Y}$ depends directly on the original response and the constant of proportionality is a matrix H called *hat matrix*.

A geometrical view of least squares Let's consider least squares regression with two predictors. The outcome vector y is orthogonally projected onto the hyperplane spanned by the input vectors x_1 and x_2. The projecttion $\hat{y}$ represents the vector of the least squares predictions.

Variance of least squares estimator Assuming input vector x_i are non-random, and errors ε_i are iid with $\mathbb{E}[\varepsilon_i] = 0$ and $\text{var}\varepsilon_i = \sigma^2$, then the **variance-covariance matrix** of $\underline{\hat{\beta}}$ is the following:

$$\text{var}(\hat{\beta}) = (X^TX)^{-1}\sigma^2$$

Also, as shown in the *Gauss-Markow Theorem* $\underline{\hat{\beta}}$ is the best linear unbiased estimator of $\underline{\beta}$. If we don't have σ^2, we can compute and unbiased estimate of the variance parameter with the following:

$$\hat{\sigma}^2 = \frac{1}{n-p-1}RSS = \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Analysis of the regression model

While discussing the following tests, we will assume that **the linear model is the correct population model and** $\varepsilon_i \sim^{iid} N(0, \sigma^2)$. Under these assumptions:

$$\hat{\beta} \sim N_{p+1}\left(\beta, (X^TX)^{-1}\sigma^2\right)$$

$$(n-p-1)\hat{\sigma}^2 \sim \sigma^2\chi^2_{n-p-1}$$

and $\hat{\beta}$ and $\hat{\sigma}^2$ are statistically independent.

In order to answer the question *is a particular X_j predictor important?* We could use the following hypotesis test:

$$H_0 : \beta_j = 0 \qquad H_a : \beta_j \neq 0$$

To run this test, we calculate the t-statistic:

$$t_j = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}$$

where v_j is the j-th diagonal element of $(X^T X)^{-1}$

Again, according to the Gauss-Markov theorem, the estimator $\underline{\hat{\beta}}$ is unbiased and has the smallest variance among all the linear unbiased estimators. The variance-covariance matrix of $\underline{\hat{\beta}}$ is the following:

$$\text{Var}\left[\hat{\beta}\right] = (X^T X)^{-1}\sigma^2$$

Variable Selection The F-statistic is one of the techniques adopted to select the predictors that are associated with the response. The task of determining the predictors is called *variable selection*.

Variable selection is used to improve the **prediction accuracy**, because the least squares estimates have low bias but large variance. The variance is reduced by fitting a model that only contains the predictors that are actually associated with the response.

Interpretation is another reason to perform variable selection. We want to identify a smaller subset of predictors with the strongest relationship with the response.

Other than the F-statistic, there are other *statistics* to perform variable selection, such as Mallow's C_p, Akaike information criterion (AIC), Bayesian information criterion (BIC), adjusted R^2.

There are also *algorithms* to perform variable selection, such as *best subsets selection*, *forward selection*, *backward selection*, *stepwise selection*. This involves fitting all the possible subset of the predictors and then choose between them based on some criterion. This is the most computationally expensive method.

Forward selection which starts with the *null model* (only intercept term). At each step fit p simple linear regressions and add to the model the variable that results in the lowest RSS, then add to that model the variable that results in the lowest RSS for the new two-variable model, and so on, until some stopping rule is satisfied.

Backward selection which starts with all the variables in the model, then removes the variable with the largest p-value, that is the variable that is the least statistically significant. The new $(p-1)$-variable model is fit, and the variable with the largest p-value is removed. This procedure continues until a stopping rule is reached. This technique cannot be used if the number of predictors is greater than the number of samples, while forward selection can always be used.

Mixed selection which starts with no variable in the model and add the variable that provides the best fit. However, if at any point the p-value for one of the variables in the model rises about a certain threshold, we remove it from the model. We continue to perform both the forward and backward steps until all variables in the model have a sufficiently low p-value and all variables outside the model would have a large p-value if added to the model.

An important thing to note is that while an R^2 value close to 1 indicates that the model explains a large portion of the variance in the response variable, it does not indicate that the model will predict future observations with great accuracy. In fact, R^2 will always increase when more variables are added to the model, even if only weakly associated with the response. This is because the R^2 statistics is computed on the training data and indicates a better fit on those data.

When fitting a model, it can be useful to plot the data if it is possible.

Confidence and prediction intervals

After fitting the multiple regression model, it is possible to predict Y on the basis of a set of values for the predictors $X_1, X_2, \ldots, X_p$. However, there is some uncertainty associated with this prediction which concernes coefficient estimates, the model bias and the irreducible error.

The inaccuracy in the coefficient estimates is related to the reducible error.

$$E(y_0 - \hat{f}(x_0))^2 = \text{bias}(\hat{f}(x_0))^2 + \text{Var}\,[(]\,\hat{f}(x_0)) + \text{Var}\,[\varepsilon]$$

Qualitative predictors

Interactions

Non-linear effects of predictors

Diagnostics

3.3 Supervised Non Parametric Regression

In this lecture we will consider a **supervised** and **non-parametric** or **distribution-free** setting, meaning that we don't know what are the models that define the data.

The training set is defined as a set of n pairs made by the features X_i and the labels Y_i:

$$T_n \triangleq \{(X_i, Y_i)\}_{i=1}^n$$

where $X_i \in \mathbb{R}$ and $Y_i \in \mathbb{R}$. The pairs (X_i, Y_i) are *i.i.d.* samples.

Supervised problem Since we are focusing on *supervised* non-parametric approaches, the function $\hat{r}(X_0)$ will depend on the training set, and we will call it **estimated regression function**:

$$r_n(X_0) \triangleq r(X_0; T_n)$$

Since $r_n(X_0)$ is a **family** of functions, for different realizations of the training set, we get a different function. When deriving the error term, we were considering $\hat{r}(X_0)$ only as a deterministic function of X_0. Now if we take into account the training set, since it is a set of pairs of random variables, the expectation is not well defined.

In order to solve this problem, we would need to condition the expectation on the training set, by computing the error for a fixed realization of T_n:

$$\mathbb{E}\left[(r_n(X_0) - Y_0)^2 \mid T_n\right] = \text{MMSE} + \mathbb{E}\left[(r_n(X_0) - r(X_0)) \mid T_n\right]$$

Since the MMSE does not depend on the training set, we can ignore the conditioning.

Under the assumption that (X_0, Y_0) is independent on the training set T_n, when applying the *tower property*, we obtain:

$$\mathbb{E}\left[(r_n(X_0) - Y_0)^2\right] = \text{MMSE} + \mathbb{E}\left[(r_n(X_0) - r(X_0))\right]$$

The first expression is a **random error term**, because the expression depends on the random variable T_n, while the second expression is a **deterministic error term**, because we compute the expectation with respect to all the possible realizations of the training set.

In practical terms, if I had to implement these two expressions in MATLAB, for the first expression I would need compute it on a single training set, while for the second expression I would need to average over all the possible realizations of the training sets.

Consistency means that the estimation error $\mathbb{E}\left[(r_n(X_0) - Y_0)^2\right]$ will converge to the MMSE as n goes to infinity, meaning that the penalty term will go to zero. We can distinguish between two types of consistency:

- **weak consistency**
- **strong consistency**

Asymptotic methods From the Law of Large Numbers, we know that we can estimate the expected value of a distribution by using the arithmetic mean and we also know that the arithmetic mean is only an approximation of the true mean, which converges to the true mean only with $n \to \infty$.

Even if we know that the arithmetic mean is an approximation, we still use it because it is a universal method, meaning that it does not depend on the specific problem. We can apply the same logic to the regression problem.

An asymptotic method should guarantee that if we collected infinite information, we would get the optimal regression function. So, ideally, we want that when n gets large, the penalty term goes to zero.

Non-parametric regression

Let's consider a general regression problem:

$$Y_0 = r(X_0) + \mathcal{E}$$

Where we didn't make any assumptions on our model, but we simply defined our error as:

$$\mathcal{E} = Y_0 - r(X_0)$$

We can see that here it always holds that $\mathbb{E}[\mathcal{E}|X] = 0$. But this happens only because $r(X_0)$ is the optimal regression function by definition.

To make a comparison, when we consider linear regression, we are trying to solve the following problem:

$$Y_0 = \beta^T X_0 + \mathcal{E}$$

and we have **to make the assumption that** $\mathbb{E}[\mathcal{E}|X] = 0$ because we are making an assumption on the *shape* of the optimal function.

Another remark: when considering a regression problem, we are trying to find Y_0 given X_0, that can vary in a continuous set. When we take one sample from our dataset:

$$Y_i = r(X_i) + \mathcal{E}_i$$

this is going to be a single point on the plot.

Exercise Find a method that allow us to approximate the optimal regression function:

$$r_n(X_0) \approx r(X_0) = \mathbb{E}[Y_0 \mid X_0]$$

If we had no information on the data and we only wanted to find the mean of Y_0, we could use the arithmetic mean:

$$\mathbb{E}[Y_0] = \frac{1}{n} \sum Y_i$$

What changes when we take into account an approximation? Given our training samples:

$$\{X_i\} = 0, 1, 0, 1, 0, 1 \dots$$
$$\{Y_i\} = y_1, y_2, y_3, y_4, y_5, y_6 \dots$$

In order to estimate the mean of the labels given $X = 0$, we would compute the arithmetic mean only over the samples were the data is zero:

$$\mathbb{E}[Y \mid X = 0] \approx \frac{y_1 + y_3 + y_5 + y_7}{4}$$

And to find the conditional mean, we would repeat the same thing for every possible X_i in our training set.

Then our method would be the following: we select all the Y_i where the X_i that are equals, compute the conditional mean and move the value of X_i. But this approach has a critical problem, which is that we won't ever find two same values of X_i in the training set because the probability of draw the exact X_i is always zero.

The solution will involve taking into account *points that are close to each other*. We continue the discussion about the supervised, non-parametric case, in which we have our training set T_n. Now we observe the feature x_0 and we want to predict its label y_0. Note that we likely don't already have a label y_0 for our point x_0, because we are assuming that X_i is a continuous random variable. We could also assume that X_i is a discrete variable but it is better to consider the general case.

Since our assumption that X_i is a continuous random variable, we know that the event $X_i = x_0$ is not an impossible event but it is highly unlikely because:

$$\Pr[X_i = x_0] = 0$$

Note that although the probability is zero, the event is not impossible, which means that we can observe x_0. Let's consider the definition of probability by applying a strong law of large numbers:

$$\Pr[X_i = x_0] = \lim_{n \to +\infty} \frac{\#\{X_i = x_0\}}{n} = 0$$

We can see that the probability of the event $X_i = x_0$ is zero in two cases: the first is that the number of occurences of the event is actually zero and the second case is that the number of the events grows **sublinearly** or slower than the number of samples.

Note that the previous equation is valid for any realization of the random variable, which means that we will have *almost surely* the same limit for each realization.

From what we said, it is reasonable to assume that $x_0 \notin T_n$.

Our goal is to approximate the following conditional probability:

$$\mathbb{E}[Y_0 \mid X_0 = x_0]$$

From the definition of conditional probability for discrete variables we know that in order to compute this conditional probability we need to take all of the values equal to x_0 and average the labels of these points. For continuous variables, this definition does not hold because it is highly unlikely to find a value equal to x_0.

The solution, at least in the approach that we will follow, will consist in a relaxation of the condition $X_0 = x_0$. In particular, we will take the values that

lie in a **neighbourhood** of x_0, of size h, that we will call $I_h(x_0)$. We can write it formally using an *indicator function*:

$$r_n^{(NK)}(x_0) = \frac{\sum_{i=1}^n Y_i \quad \mathbb{I}\{\|X_i - x_0\| \leq h\}}{\sum_{i=1}^n \mathbb{I}\{\|X_i - x_0\| \leq h\}}$$

Note that we used the *euclidean norm* because we are referring to a generic number of dimension d. This estimator is called **naive kernel estimator**. The main problem of this approach is that we don't know how many points will fall into the neighbourhood.

If we wanted a fixed number of points in the neighbourhood, we would need to use another estimator, called the **nearest neighbour estimator**. With this estimator, we fix k, that is the number of points that we want in the neighbourhood and we take the k closest neighbours of x_0 as points that constituites the neighbourhood.

To formally define this estimator, we need to introduce the *sorted list notation*:

$$X_{(1)}(x_0), \ldots, X_{(n)}(x_0) \text{ s.t. } \|X_{(1)}(x_0) - x_0\| \leq \|X_{(2)}(x_0) - x_0\| \leq \ldots \|X_{(n)}(x_0) - x_0\|$$

Where $X_{(1)}(x_0)$ is the closest sample to x_0, $X_{(2)}(x_0)$ is the second closest sample and so on. We can use the follwing notation to denote also the labels:

$$\begin{pmatrix} X_{(1)}(x_0) \\ Y_{(1)}(x_0) \end{pmatrix}, \ldots, \begin{pmatrix} X_{(n)}(x_0) \\ Y_{(n)}(x_0) \end{pmatrix}$$

Note that $Y_{(n)}(x_0)$ is not the n-th closest label, but the label associated to the n-th closest sample to x_0. Now we can define the nearest neighbour estimator:

$$r_n^{(NN)}(x_0) = \frac{1}{k} \sum_{i=1}^{k} Y_(i)(x_0)$$

For a good approximation of the conditional probability we need two things: many samples in the neighbourhood of x_0 and we need that the neighbourhood is small. The first condition is a consequence of the **LLN**.

When the number of sample grows, the number of points in the neighbourhood should grow too. We can justify this reasoning in this way:

$$\frac{\#\{X_i \in I_h(x_0)\}}{n} \overset{n \to +\infty}{\approx} \Pr\left[X_i \in I_n(X_0)\right] = p > 0$$

When the number of samples grows, we can approximate the number of points that lie in the neighbourhood over n as the probability that we will find a point in the interval. Since this probability is a positive number p, it means that the number of points grows linearly with the number of samples.

The second condition states that the neighbourhood should be enough small, this is also called **principle of locality**. This is the reason why these methods are also called **local methods**.

There seem to be an apparent contradiction, which is that if we reduce the size of the neighbourhood $h \to 0$, then the number of points in neighbourhood

reduces to, thus violating the first condition. We may think that we can consider h as a very small fixed number. In this case we are not converging to the real expectation value so we lose the locality princple.

The only way to guarantee both conditions is to scale the size of the neighbourhood h along with increasing the number of samples. But this has to happen under certain conditions that we will now introduce. Fristly, we try to approximate the probability that a point lies in $I_h(x_0)$:

$$\Pr\left[X_i \in I_h(X_0)\right] \approx f_X(x_0) 2h_n$$

because we can assume hat if h is very small, then the area under curve of x_0 is given by the area of the rectangle with base equal to the size of the neighbourhood $2h$ and height equal to the pdf of x_0.

We also know that we can approximate the probability by the definition given by the frequentist approach:

$$\Pr\left[X_i \in I_n(X_0)\right] \approx \frac{\#X_i \in I_h(x_0)}{n} \approx f_X(x_0) 2h_n$$

From this, we can derive that:

$$\text{no. of points } \propto h_n n$$

From this relation, in order to guarantee the the principle of locality we want that for $n \to +\infty$, $h_n \to 0$, while to guarantee LLN we need that $h_n n \to +\infty$, which are two reasonable requirements.

If h_n scales with law $\frac{1}{n^p}$ with $p > 1$, then we won't converge, while if h_n scales with law $\frac{1}{\sqrt{n}}$ then we will converge.

We can conclude that for this estimator, h_n should scale sublinearly while n grows linearly.

Note that if we have d dimensions, since we need to compute the volume of the hypercube that approximates the area under the curve, the number of points given by:

$$\text{no. of points } \propto f_x(x_0) 2h_n^d$$

This means that if we increase d, n should grow exponentially to match the growth rate of h, and this is also known as the *curse of dimensionality*.

As for the K-NN estimator, it can be proved that it is **weakly consistent**.

For this estimator, the number of points is given by the parameter k_n, so the conditions are reversed with respect to the naive estimator. For this estimator, we need to guarantee that $k_n \to +\infty$ in order to satisfy the LLN requirement and $\frac{k_n}{n} \to 0$ in order to satisfy the locality requirement.

3.4 Exercise

Chapter 4

Linear Methods for Regression

Chapter 5

Classification

5.1 Introduction

This lecture is about classification, also called decision making or hypothesis testing. In order to study classification, we will follow a path similar to the one on regression, that is we will find, at first, the best that we can do if we already have the model, then we will examine the case where the parameter is not random and finally we will study the supervised approach.

In classification, I have data X and Θ that is the parameter I am interested in and it is discrete, which means that:

$$\Theta \in \{\theta_1, \theta_2, \ldots, \theta_H\}$$

where each θ_i is called a **class** or an **hypothesis**.

In some contexts, the parameter Θ is said to be **categorical**, which means that each class is a category. Since we are interested in the probability of each class, it's equivalent to consider them as a category or as a discrete number. However, there is a slight difference in the two terminologies, because if the parameter is discrete, then its classes will be numbers like $1, 2, \ldots, H$, while if the parameter is categorical, then its classes will be categories such as cat, dog, bird, et cetera.

In regression we quantify the error using as a metric the distance between the true value and the predicted value; thus it makes no sense to use categories in regression. In classification we don't care about the distance between categories, because the concept is not properly defined. We have only two possibilities that are: we belong to the category or not. Also, to quantify the error instead of using the MSE, we use the probability of error.

It is important to stress that in the classification problem, during the prediction we are not instered in *estimating* a class, but we are interested in make a choice about the class.

The performance metric will be the error probability:

$$\Pr\left[\hat{\Theta}(X) \neq \Theta\right]$$

In telecommunication we used maximum error probability **MAP**.

5.2 Model-Based Classification

5.2.1 Bayesian approach

Let us first use the bayesian approach to find the best performances. We define the **posterior distribution** of the hypothesis as the following p.m.f:

$$\Pr[\Theta = \theta \mid X]$$

Also we define the **prior distribution** as the following p.m.f:

$$\pi(\theta) = \Pr[\Theta = \theta]$$

Note that the most complete information about the parameter Θ is given by the posterior distribution. We cannot have more information than the one give by this distribution.

Usually, the prior distribution is not very informative, we can assume that it is given by one over the number of hypothesis for θ and zero for all the other possibile hypothesis.

[PLOT]

It is intuitive and in this case correct to find that in order to minimize the probability of making a mistake, we decide for the parameter that has the highest probability according to the posterior distribution, that is also the way to maximize the probability to make the right choice.

Remember that in the regression case we wanted to minimze the MSE by finding the value that maximized the likelihood. ...

Recall that the *prior* is embedded in the posterior through Bayes' rule.

We can formally show that this choice maximizes the probability of making the right choice and that this minimizes the error probability, but we won't do that.

We are now interested in answering the question *does the system learn correctly if we have many data?*

Bayes' update In regression, we have the prior $\pi(\theta)$ and the likelihood $\ell(X \mid \theta)$ which is usually a vector and it is the generative mechanism of our data. Let us call $\mu(\theta \mid X)$ the posterior distribution that in Bayesian statistic is usually also called **belief**. From Bayes' rule, we know the expression of the posterior:

$$\mu(\theta \mid X) = \frac{\pi(\theta)\ell(x \mid \theta)}{\sum_{\theta'} \pi(\theta')\ell(x \mid \theta')}$$

Where the denominator is a normalizing constant called **marginal distribution**.

We can verify the expression of the marginal distribution by computing the sum of the marginal distribution over all the possible θ' and checking if it equals 1.

When x is frozen, the marginal distribution is just a normalizing constant, so the belief is proportional to prior times the likelihood:

$$\mu(\theta \mid X) \propto \pi(\theta)\ell(x \mid \theta)$$

This expression is also called Bayes update.

What can I expect from a good learning system? Assume that I observe $X = [X_1, \ldots, X_n]$ that are i.i.d according to $\ell(X_i \mid \theta_0)$. Ideally, I'd want that, if the true hypothesis is θ_0, $\mu(\theta_0 \mid X) \to 1$ when $n \to +\infty$. This means that not only I want that the probability of the error is zero but also that my system has the highest posterior distribution possible.

Proof. Let us compute the ratio between the posterior distribution if the parameter is θ_0 and the posterior distribution if the parameter is a certain $\theta \neq \theta_0$.

$$\frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)} = \frac{\pi(\theta_0)\prod_{i=1}^{N}\ell(X_i \mid \theta_0)}{m(x)}\frac{m(x)}{\pi(\theta)\prod_{i=1}^{N}\ell(X_i \mid \theta)} = \frac{\pi(\theta_0)}{\pi(\theta)}\frac{\prod_{i=1}^{N}\ell(X_i \mid \theta_0)}{\prod_{i=1}^{N}\ell(X_i \mid \theta)}$$

By applying the log to both members:

$$\log\left(\frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)}\right) = \log\left(\frac{\pi(\theta_0)}{\pi(\theta)}\right) + \log\left(\frac{\prod_{i=1}^{N}\ell(X_i \mid \theta_0)}{\prod_{i=1}^{N}\ell(X_i \mid \theta)}\right)$$

Observe that by applying logarithm property, the product becomes a sum. Then we divide both members by n:

$$\frac{1}{n}\log\left(\frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)}\right) = \frac{1}{n}\log\left(\frac{\pi(\theta_0)}{\pi(\theta)}\right) + \frac{1}{n}\sum_{i=1}^{N}\log\left(\frac{\ell(X_i \mid \theta_0)}{\ell(X_i \mid \theta)}\right)$$

When n approaches infinity, the first term goes to zero because the ratio of the priors is a finite number, while the second term converges (according to the law of large numbers) to the expected value **computed under the true hypotesis** of the logarithm of the ratio of the likelihoods.

$$\frac{1}{n}\log\left(\frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)}\right) \xrightarrow{n\to+\infty} \mathbb{E}_{\ell(\cdot\mid\theta_0)}\left[\frac{\ell(X_i \mid \theta_0)}{\ell(X_i \mid \theta)}\right]$$

□

This expectation is called **Kullback-Leibler divergence**.

$$\mathbb{E}_{\ell(\cdot\mid\theta_0)}\left[\frac{\ell(X_i \mid \theta_0)}{\ell(X_i \mid \theta)}\right] \triangleq D_{KL}(\theta_0 \mid\mid \theta)$$

In general, the Kullback-Leibler divergence explains how different are two distributions. It is a non-negative number that is zero if and only if the two distributions are equal.The greater the distributions are different, the greater the divergence. If p and q are two pmfs, then:

$$D_{KL}(p \mid\mid q) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)} = \mathbb{E}_p\left[\log\frac{p(x)}{q(x)}\right]$$

If p and q are two pdfs, then:

$$D_{KL}(p \,||\, q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Now we're going to prove that, given that the ratio of belief converges to the Kullback-Leibler divergence, our system learns the correct hypothesis.

Proof. We've shown that the quantity for $n \to +\infty$ converges to the Kullback-Leibler divergence that is a non-negative quantity.

$$\frac{1}{N} \log \frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)} \to D_{KL}(\theta_0 \,||\, \theta) > 0$$

Under the assumption of **identifiability**, that is the assumption that $\ell(x \mid \theta_0) \neq \ell(x \mid \theta)$, if we multiply both members by N, we get that D_{KL} goes to $+\infty$. But this means that also the first member should diverge to infinity and we know that a logarithm will diverge only if its argument diverges:

$$\frac{\mu(\theta_0 \mid X)}{\mu(\theta \mid X)} \to +\infty$$

This quantity can diverge either if the numerator goes to infinity or the denominator goes to zero. Since the posterior distribution is a probability, it is a finite number, so we have that $\mu(\theta \mid X) \to 0$ and this implies, given the identifiability property that $\mu(\theta_0 \mid X) \to 1$. □

With respect to the previous lesson, we are going to find what happens if our data is generated by an arbitrary function f, that we are going to assume is not the likelihood function of the true hypothesis.

As in the previous proof, we are going to compute the ratio between the belief of θ given x and the belief of a θ' given x.

$$\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} = \frac{\pi(\theta)\ell(x \mid \theta)}{\pi(\theta'\ell(x \mid \theta'))} = \frac{\pi(\theta)}{\pi(\theta')} \prod_{i=1}^{N} \frac{\ell(x_i \mid \theta)}{\ell(x_i \mid \theta')}$$

Now we apply the logarithm to both members and we divide both members by n:

$$\frac{1}{n} \log \frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} = \frac{1}{n} \log \frac{\pi(\theta)}{\pi(\theta')} + \frac{1}{n} \sum_{i=1}^{n} \log \frac{\ell(x_i \mid \theta)}{\ell(x_i \mid \theta')}$$

Now we observe that the ratio of the prior functions goes to zero when n goes to $+\infty$ because it is a constant divided by n. This has also another concrete meaning, that is if we have many data, the prior information we had at the start is ignored. We observed the same behaviour during the MSE analysis, where when n was large the prior information was unuseful. [CORREGGERE].

Even though we have a strong bias at the start, with infinite data we discard the prior information. The only exception to this is when the one of the hypotesis has the prior function equal to zero. In that case, we cannot remove the initial bias mathematically and it also makes sense in the concrete, because

Then we got:

$$\frac{1}{n}\log\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} = \frac{1}{n}\sum_{i=1}^{n} Z_i$$

where each Z_i is independent and identically distributed to the ratio of the likelihoods. When n goes to $+\infty$, according to the law of large numbers we know that the term on the right converges to the expected value of Z

$$\frac{1}{n}\log\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} = \mathbb{E}_Z\left[\log\frac{\ell(x \mid \theta)}{\ell(x \mid \theta')}\right]$$

Each Z_i is generated by the true generative mechanism that is the function $f(X)$. Thus we can write:

$$\frac{1}{n}\log\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} = \mathbb{E}_f\left[\log\frac{\ell(x \mid \theta)}{\ell(x \mid \theta')}\right]$$

Now we can multiply and divide the argument of the logarithm by $f(X)$ to obtain an expression that depends on the Kullback-Leibler divergence.

$$\mathbb{E}_f\left[\log\frac{\ell(x \mid \theta)}{\ell(x \mid \theta')}\frac{f(X)}{f(X)}\right] = \mathbb{E}_f\left[\log\frac{f(x)}{\ell(x \mid \theta')}\right] - \mathbb{E}_f\left[\log\frac{f(x)}{\ell(x \mid \theta)}\right]$$

So we found that:

$$\frac{1}{n}\log\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} \xrightarrow{n\to+\infty} D_{KL}\left(f||\ell(\cdot \mid \theta')\right) - D_{KL}\left(f||\ell(\cdot \mid \theta)\right)$$

let's rewrite the this expression by using a slightly changed notation:

$$\frac{1}{n}\log\frac{\mu(\theta \mid x)}{\mu(\theta' \mid x)} \xrightarrow{n\to+\infty} D_{KL}\left(f||\theta'\right) - D_{KL}\left(f||\theta\right)$$

In order to find *which hypothesis will the system learn*, we need to recall that the chain of implications in the previous proof, required the ratio of beliefs to converge to a positive value (when n approaches $+\infty$):

$$\frac{1}{n}\frac{\mu(\theta^* \mid X)}{\mu(\theta' \mid X)} \to D_{KL}\left(f||\theta'\right) - D_{KL}\left(f||\theta\right) > 0$$

Now, we consider a θ^* that isn't an arbitrary hypothesis, but we assume there exists:

$$\theta^* : D_{KL}\left(f||\theta'\right) > D_{KL}\left(f||\theta^*\right) \qquad \forall\theta' \neq \theta^*$$

If this condition is satisfied, then the chain of implications used in the last lecture holds again and we can use it to prove that $\mu(\theta^* \mid X) \to 1$.

Note that θ^* is the hypothesis that has the smalled distance (in terms of the Kullback-Leibler divergence) from the true distribution.

In our proof, there is still a singularity case but it is removed under the assumption of unidentifiability. [WHICH ONE?]

[INSERT PLOT] In this example, our system will learn the hypothesis θ_3.

From this proof, we found out that our model approximates the truth when our model is the closest in terms of Kullback-Leibler divergence to the true model.

Note that this is the general case, and the previous proof was a special case of this. It is simple to show that if $f(x) = \ell(x \mid \theta_0)$, the minimum will be the true hypothesis θ_0.

Exercise Having defined

$$\mu_n(\theta) \triangleq \mu(\theta \mid x_1, \dots, x_n)$$

Can you find a relationship between $\mu_n(\theta)$ and $\mu_{n-1}(\theta)$? We know that:

$$\mu_n(\theta) \propto \pi(\theta) \prod_{i=1}^{n} \ell(x_i \mid \theta) = \pi(\theta) \prod_{i=1}^{n-1} \ell(x_i \mid \theta) \ell(x_n \mid \theta)$$

and that

$$\mu_{n-1}(\theta) \propto \pi(\theta) \prod_{i=1}^{n-1} \ell(x_i \mid \theta)$$

So we can infer that:

$$\mu_n(\theta) = \mu_{n-1}(\theta) \ell(x_n \mid \theta)$$

This is none other that *Bayes' rule* where our past belief $\mu_{n-1}(\theta)$ is the current prior. This property, called **sequential property**, explains why we talked about *Bayes' updates*: by doing successive updates we starts from the prior and step by step we update our belief with new evidence given from the likelihood.

This property holds only thanks to the hypothesis of **independendence** of the data. With an independent model, all the knowledge we have at a certain step is contained in the current belief. This property is really important, due to computation reasons, because it simplifies the calculations and allows to use Bayes updates in a streaming setting. In some ways it can be considered as a form of online learning.

5.2.2 Neyman-Pearson Criterion

Now we want to find what happens when our parameter Θ is deterministic. First of all, let us clarify that in the Bayes problem we encountered before, Θ was random, even though we worked with fixed value of theta. This is because the same reasoning applies to any choice of the hypothesis.

Suppose we have only 2 classes, that are $\theta \in \{-1, 1\}$. *In this case, is our performance metric still the error probability?* No, because we have two different error probabilities. It is important to stress that the only indicators we will need to explain classification are the ones containes in the ROC curve:

$$\alpha = \Pr[\text{choose } 1 \mid -1 \text{ is true}] \qquad \text{type 1 error probability}$$

$$\beta = \Pr[\text{choose } -1 \mid 1 \text{ is true}] \qquad \text{type 2 error probability}$$

By contrast, in the Bayesian setting, our error probability was given, according to the theorem of the total probability, by the arithmetic average of the two errors:

$$P_{err} = \alpha\pi(-1) + \beta\pi(+1) = \alpha\pi(-1) + \beta[1 - \pi(-1)]$$

Now we want to find *what is the optimal strategy to use?* Note that we cannot minimize both errors, because if we lower one error, the other one increases. For example, in target detection we want to minimize the false negatives. The optimal strategy is the **Neyman-Pearson Criterion**, which requires to **minimize**

β over all possible strategies that ensure that $\alpha \leq \alpha^*$. The solution of this criterion is to compare the likelihood ratio to a threshold γ, which depends on α^*:

$$\frac{\ell(x \mid +1)}{\ell(x \mid -1)} \underset{+1}{\overset{-1}{\lessgtr}} \gamma_{\alpha^*}$$

Let use recall the ROC curve that is the following: [PLOT]

The straight line is the silliest decisor, which uses a *randomized rule*, for example, if $\alpha = 0.5$ and $\beta = 0.5$, it flips a coin and decides.

Next lecture we will discuss about the relationship betweem Neyman-Pearson criterion and the MAP rule and also how to increase the performance of a decisor.

5.3 Supervised Classification

In our previous lessons, we have seen model-based classification, both the bayesian setting and the Neyman-Person case. In this lesson, we will see how to perform supervised classification. The setting is similar to the one of the regression. We have our classes

$$\Theta \in \{\theta_1, \theta_2, \ldots, \theta_H\}$$

and our features $X \in \mathbb{R}^d$. As in the regression case, we will have to define a risk function based on the model-based theory (the MMSE) and then use the empirical version of the risk to perform the supervised classification.

In the regression case, we have seen that the MMSE is

$$\text{MMSE} = \min_f \mathbb{E}\left[(Y - f(X))^2\right]$$

We now replace $\mu(\theta \mid X)$ with $\hat{\mu}_\beta(\theta \mid X) \in f$. f will be a parametric class, which has β as a paramater. We will then have to pick a function of this class which will always be a probability mass function for each value of f.

Assume that now we have picked a model. Our goal is to find a function $\hat{\mu}$ which error probability is comparable to the one of μ. But there is a problem that we have to replace the error probability with something that we can use in optimization theory. We can use the Kullback-Leibler divergence.

So now our goal is to find:

$$\beta^* = \arg\min_{\beta \in \mathbb{R}^M} \overline{D}_{KL}\left(\mu \,||\, \hat{\mu}_\beta\right)$$

By definition of Kullback-Leibler divergence, we have that:

$$\mathrm{D}_{KL}\left(\mu \,||\, \hat{\mu}_\beta\right) = \mathbb{E}_\mu\left[\log \frac{\mu(\theta \mid x)}{\hat{\mu}_\beta(\theta \mid x)}\right] = \sum_\theta \mu(\theta \mid x) \log \frac{\mu(\theta \mid x)}{\hat{\mu}_\beta(\theta \mid x)} = f_\beta(X)$$

Since we need a number but we have a function of X, we can take the expectation of $f_\beta(X)$ to get a number. This version of the Kullback-Leibler divergence is called **conditional Kullback-Leibler divergence**.

$$\text{conditional } D_{KL} = \mathbb{E}_X \left[\sum_\theta \mu(\theta \mid x) \log \frac{\mu(\theta \mid x)}{\hat{\mu}_\beta(\theta \mid x)} \right]$$

Note: In information theory, to get the conditional quantities of some measures like the entropy and the Kullback-Leibler divergence, we don't need to apply the conditional probability formula, but we just need to take the expectation of this quantities.

Let's rewrite the conditional Kullback-Leibler divergence in a shorter form:

$$\overline{D}_{KL}(\mu \,||\, \hat{\mu}_\beta) = \mathbb{E}_{X,\Theta} \left[\log \frac{\mu(\Theta \mid X)}{\hat{\mu}_\beta(\Theta \mid X)} \right]$$

To sum up:

1. We want to approximate the optimal posterior mean μ with $\hat{\mu}_\beta$.
2. We found a way to measure the quality of the approximation, the conditional Kullback-Leibler divergence.
3. We need to minimize the Kullback-Leibler divergence and find β^*
4. We do not know the posterior mean but we can show that it is not useful to find β^*.

Assume we have a training set $T_n = \{\theta_i, x_i\}_{i=1}^N$ and our classes are $\Theta = \{1, 2, \ldots, H\}$.

We can define the empirical conditional Kullback-Leibler divergence (our *empirical risk*) as:

$$\overline{D}_{KL} = \frac{1}{N} \sum_{i=1}^{N} \log \frac{\mu(\theta_i \mid x_i)}{\hat{\mu}_\beta(\theta_i \mid x_i)}$$

Now we have a problem. We want to minimize the conditional Kullback-Leibler divergence, but we do not know the posterior mean μ. Let us isolate μ in the non-empirical conditional Kullback-Leibler divergence formula:

$$\overline{D}_{KL} = -\mathbb{E}_{X,\Theta} \left[\log \frac{1}{\mu(\Theta \mid X)} \right] + \mathbb{E}_{X,\Theta} \left[\log \frac{1}{\hat{\mu}_\beta(\Theta \mid X)} \right]$$

Now we define a quantity called **entropy**, that it is a measure of the amount of information in a random variable:

$$\mathcal{H} = \mathbb{E}_X \left[\log \frac{1}{p(X)} \right] \text{ nats}$$

For a discrete random variable:

$$\mathcal{H}(p) = \sum_\theta p(\theta) \log \frac{1}{p(\theta)}$$

We define the **cross-entropy** as:

$$\mathcal{H}(p; q) = \sum_\theta p(\theta) \log \frac{1}{q(\theta)}$$

The Kullback-Leibler divergence is also called **relative entropy**. Observing the decomposition we've written before, we have that the first term is the conditional entropy of the true distribution.

Now we can rewrite a conditional Kullback-Leibler divergence in terms of entropy and cross-entropy:

$$\overline{D}_{KL}(p;q) = \mathcal{H}(p;q) - \mathcal{H}(p)$$

Since we know that the Kullback-Leibler divergence is always positive, we can write:

$$\mathcal{H}(p;q) = \mathcal{H}(p) + \overline{D}_{KL}(p;q)$$

So we found out that the cross-entropy between two distributions p and q is given by the sum of the entropy and the Kullback-Leibler divergence. This result is used to explain the *lower bound of compression*. The number of bits in the compression of some data described by a random variable X is given by the entropy of X. If we know the distribution of X, we can use the cross-entropy to compress X. The number of bits needed to compress X is given by the entropy of X plus the Kullback-Leibler divergence between the true distribution and the one we used to compress X. If we use the right distribution, then the Kullback-Leibler divergence is zero and we can compress X with the same number of bits of the entropy of X.

We found that

$$\overline{D}_{KL} = -\begin{matrix}\text{conditional entropy}\\ \text{of the true distribution}\end{matrix} + \begin{matrix}\text{conditional cross-entropy}\\ \text{between true and}\\ \text{candidate distribution}\end{matrix}$$

Note that the first term does not depend on β but it is a function of the problem. So we can minimize the conditional Kullback-Leibler divergence by minimizing the second term only. Note that without knowing the conditional entropy of the posterior mean we cannot find the value of the minimum, but we can still solve the optimization problem that requires us to find the minimizer. As in the regression part where we had the MMSE, the first term is an unbeatable error and it is an attribute of the problem, depending on the joint distribution of X and Θ.

Now our optimization problem is:

$$\beta^* = \arg\min_{\beta \in \mathbb{R}^M} \frac{1}{N} \sum_{i=1}^{N} \log \frac{1}{\hat{\mu}_\beta(\theta_i \mid x_i)}$$

But now two problem arises:

- *How do we implement this minimation problem?* We will see in the next lecture.
- *How do we choose $\hat{\mu}_\beta$?*

Logistic Regression

To choose $\hat{\mu}_\beta$, there are lots of different standard methods. As an example we will see the **logistic regression**. In particular, we will see the **binary logistic regression**, where we have $\Theta \in \{1, -1\}$.

Let us start by obtaining the decision function from the MAP rule:

$$\frac{\hat{\mu}_\beta(+1 \mid x)}{\hat{\mu}_\beta(-1 \mid x)} \lessgtr 1 \Leftrightarrow \frac{\hat{\mu}_\beta(+1 \mid x)}{1 - \hat{\mu}_\beta(+1 \mid x)} \Leftrightarrow \log \frac{\hat{\mu}_\beta(+1 \mid x)}{1 - \hat{\mu}_\beta(+1 \mid x)} \lessgtr 0$$

By defining:

$$f_\beta(X) \triangleq \log \frac{\hat{\mu}_\beta(+1 \mid x)}{1 - \hat{\mu}_\beta(+1 \mid x)}$$

We derive μ as a function of f:

$$\mu = \frac{1}{1 + e^{-f}}$$

and we can rewrite it for each class:

$$\hat{\mu}_\beta(\theta) = \frac{1}{1 + e^{-\theta f_\beta(X)}}$$

When the problem is linearly separable, we can use **logistic regression**, by imposing $f_\beta(X) = \beta^T X$.

Important note: This theory applies also for the m-ary case and for arbitrary decision functions that are not the MAP rule. The motive for why we use the sigmoid function is not because we need a value from 0 to 1 but because it allows us to write the posterior mean as a function of the decision function, and because it is impractical to use the posterior distribution in a supervised learning context. In any case, it is better to not to be reliant on the sigmoid function while doing binary classification.

5.4 Exercises

5.4.1 Exercise 1

Suppose I have iid features $x = [x_1, x_2]^T$ and labels $\Theta \in \{\theta_0, \theta_1\}$. Let us assume for simplicity that the variance is 1.

Given the vector $m = [m_1 m_2]^T$, under the hypothesis "-1" the features are $x_i \sim N(0, \sigma^2)$, while under the hypothesis "+1" the features are $x_i \sim N(m_i, 1)$, for $i = 1, 2$.

Let us compute the likelihood of the data under the two hypothesis:

$$\ell(x \mid -1) = \frac{1}{2\pi} \exp\left(-\frac{1}{2}x_1^2 - \frac{1}{2}x_2^2\right) = \frac{1}{2\pi} \exp\left(-\frac{\|x\|^2}{2}\right)$$

$$\ell(x \mid -1) = \frac{1}{2\pi} \exp\left(-\frac{\|x - m\|^2}{2}\right)$$

Bayesian case Applying Bayes rule we know that:

$$P(-1 \mid x) \propto \pi(-1)\exp\left(-\frac{\|x\|^2}{2}\right)$$

$$P(+1 \mid x) \propto \pi(+1)\exp\left(-\frac{\|x-m\|^2}{2}\right)$$

Now we can apply the Bayesian test:

$$\frac{P(+1 \mid x)}{P(-1 \mid x)} \gtrless_{-1}^{+1} 1$$

We take the logarithm to simplify:

$$\log \frac{P(+1 \mid x)}{P(-1 \mid x)} \gtrless_{-1}^{+1} 0 \Leftrightarrow \log \frac{\pi(+1)}{\pi(-1)} - \frac{\|x-m\|^2}{2} + \frac{\|x\|^2}{2}$$

Now let us simplify the squared norm:

$$\|x-m\|^2 = (x-m)^T(x-m) =$$
$$= x^Tx + m^Tm - x^Tm - m^Tx = \|x\|^2 + \|m\|^2 - 2m^Tx$$

Note: in case the two features are correlated, we found out that the term is $(x-m)^TC(x-m)$, where C is the variance-covariance matrix.

The final log posterior ratio is:

$$\log \frac{P(+1 \mid x)}{P(-1 \mid x)} = \log \frac{\pi(+1)}{\pi(-1)} + m^Tx - \frac{\|m\|^2}{2}$$

From the Bayesian test statistic we can obtain quickly the log-likelihood ratio between they are equal unless for the ratio of the prior.

$$\log \frac{\ell(x \mid +1)}{\ell(x \mid -1)} = m^Tx - \frac{\|m\|^2}{2}$$

Note: the Bayesian test is a particular application of the Neyman-Pearson test, where the threshold is set. This threshold is the one that minimizes the probability of error.

$$P_{err} = \pi(-1)\alpha + \pi(+1)\beta$$

In the Neyman-Pearson case we can change the threshold, but the Bayesian test picks a particular pair (α, β) that minimizes the probability of error. Altough the Bayesian test should give us the *optimum*, in most practical application we prefer to use the Neyman-Pearson Lemma because we can obtain a decisor with a fixed value of α.

Analytical ROC Curve In this case, to implement the Neyman Pearson decisor, I can use the log-likelihood ratio, because it is a monotonic function of the likelihood ratio, and then I can remove the costants because they can be absorbed in the threshold.

Then the test statistic becomes:

$$m^T x \gtrless_{-1}^{+1} \gamma$$

Since the test statistic is a linear combination of gaussians and so a gaussian itself, we can compute the probabilities of the error in closed form.

$$z = m^T x = m_1 x_1 + m_2 x_2$$

Under the hypothesis "-1" we have $z \sim N(0, \|m\|^2)$, while under the hypothesis "+1" we have $z \sim N(\|m\|, \|m\|^2)$.

Proof. Under the hypothesis "-1" we have that both x_1 and x_2 are gaussian with mean 0 and variance 1.

$$\mathbb{E}[z] = \mathbb{E}[m_1 x_1 + m_2 x_2] = m_1 \mathbb{E}[x_1] + m_2 \mathbb{E}[x_2] = 0$$

For the variance:

$$\mathrm{Var}(z) = \mathrm{Var}(m_1 x_1 + m_2 x_2) = m_1^2 \mathrm{Var}(x_1) + m_2^2 \mathrm{Var}(x_2) = \|m\|^2$$

Under the hypothesis "+1" we have that x_1 and x_2 are gaussian with mean m_1 and m_2 and variance 1.

$$\mathbb{E}[z] = \mathbb{E}[m_1 x_1 + m_2 x_2] = m_1 \mathbb{E}[x_1] + m_2 \mathbb{E}[x_2] = \|m\|^2$$

And the same as before for the variance. □

Now we can compute the probability of error of type I:

$$\alpha = P\left[m^T x > \gamma \mid y = -1\right] = P\left[N(0, \|m\|^2)\right] = Q\left(\frac{\gamma}{\|m\|}\right)$$

And the sensitivity:

$$1 - \beta = P\left[m^T x < \gamma \mid y = +1\right] = P\left[N(\|m\|^2, \|m\|^2)\right] = Q\left(\frac{\gamma - \|m\|^2}{\|m\|}\right)$$

This allow us to find the equation of the ROC curve.

$$\frac{\gamma}{\|m\|} = Q^{-1}(\alpha) \Rightarrow 1 - \beta = Q\left(Q^{-1}(\alpha) - \|m\|\right)$$

Shape of the optimal distribution In the supervised case, we want to learn the posterior distribution $P(y \mid x)$. For the results we mentioned about monotonicity, we can use the log posterior ratio to find the optimal **function** $f^*(x)$.

When doing logistic regression, we imposed a particular shape for the posterior distribution that is:

$$f_\beta(x) = \beta^T x + \beta_0$$

And by equating the log posterior ratio to this function:

$$\log \frac{P(+1 \mid x)}{P(-1 \mid x)} = \log \frac{\pi(+1)}{\pi(-1)} + m^T x - \frac{\|m\|^2}{2} = \beta^T x + \beta_0$$

We can find the optimal parameters β and β_0:

$$\beta = m \qquad \beta_0 = \log \frac{\pi(+1)}{\pi(-1)} - \frac{\|m\|^2}{2}$$

Some notes: we've showed that the optimal function belongs to the family of function we've chosen and the intercept contains the prior.

Chapter 6

Optimization

6.1 Assumptions

Lots of problem in statistical learning can be formulated as optimization problem. Thus, optimization is important because we need to find how solve the optimization problems in a computationally efficient way. Many statistical learning tasks involve the minimization of a *cost* function. This function is also sometimes called *cost*, *loss*, *risk* or *objective* function.

This is the generic form of an *unconstrained minimization problem*:

$$\min_{\beta} J(\beta)$$

Where $\beta \in \mathbb{R}^d$.

[YAPPING]

Example with the regression case.

Example with the logistic regression case.

Another thing is the more we relax the assumptions, the less information we get. *The less we ask, the less we get.* For example, when I work under the assumption that

A **performance guarantee** means that whatever we do the dataset seems to work, meaning that we can apparently obtain some information.

In order to define our problem, let us assume that we have our **random data** $D \in \mathcal{D}$ in form: $D = (X, Y)$. We then choose a **loss function** $Q_\beta(d)$, that defines our actual cost without taking expectation. $\beta \in \mathbb{R}^p$ is our vector of parameters.

For example, if we consider linear regression and we pick our data $d = \{x, y\}$, then the loss function will be $Q_\beta(d) = Q_\beta(x, y) = (y - \beta^T X)^2$.

Then, assuming that the statistical distribution of D is known, our task is to minimize the **loss averaged over** D:

$$J(\beta) = \mathbb{E}\left[Q_\beta(D)\right]$$

Now are going to state the **assumptions** that we require to introduce the gradient descent algorithm. But first some remarks:

- Our *averaged loss function* is $J : \beta \in \mathbb{R}^p \rightarrow j \in \mathbb{R}$.
- $\nabla J(\beta)$ is the gradient of our *averaged loss function* and it is a vector of dimensions $p \times 1$, where each element is $\frac{\partial J(\beta)}{\partial \beta_i}$
- we are going to use the **euclidean norm** as a norm.

We need two assumptions: (1) **Lipschitz continuity** and **convexity**. First, remember that a function is continuous if two points become closer, then the values of the function evaluated in those two points also becomes closer, that is the difference between the two points becomes closer to zero. When we say a function is continuous, we don't know that is the rate with which they become closer. This is why we need to introduce the *Lipschitz-continuity of the gradient*. The definition is the following:

$$\forall \beta_1, \beta_2 \colon \|\nabla J(\beta_2) - \nabla J(\beta_1)\| \leq \delta \|\beta_2 - \beta_1\|$$

That is that the difference of the gradient is upper bounded by the difference of the parameters times a constant. When the gradient is Lipschitz-continuous we usually say *the gradient varies smoothly*.

Note: if a function is Lipschitz-continuous then it is also continuous.

Now let us introduce the convexity. A **convex combination** is the sum of two quantites weighted respectively by the coefficient α and $1 - \alpha$. We say that a function is **convex** if

$$\forall \alpha \in (0, 1) \text{ and } \forall \beta_1 \neq \beta_2$$

we have that:

$$J(\alpha\beta_1 + (1 - \alpha)\beta_2) \leq \alpha J(\beta_1) + (1 - \alpha) J(\beta_2)$$

which means that the function evaluated in the convex combination of the two parameters lies also below the chord that is described by the convex combination of the parameters.

[IMMAGINE]

The most frequent convention in optimization is doing **minimization**, and thus assuming that the cost function is convex. However, if we need to solve a *maximization problem*, then we would need a *concave* function.

Obviously, since the definition of convexity involves a less or equal definition, then the definition of *strict convexity* involves a strict less than sign. A function is **strictly convex** if:

$$\forall \alpha \in (0, 1) \text{ and } \forall \beta_1 \neq \beta_2$$

we have that:

$$J(\alpha\beta_1 + (1 - \alpha)\beta_2) < \alpha J(\beta_1) + (1 - \alpha) J(\beta_2)$$

Then we have another definition, which is the **strong convexity**. A function is **strongly convex** if:

$$\forall \alpha \in (0, 1) \text{ and } \forall \beta_1 \neq \beta_2$$

we have that:

$$J(\alpha\beta_1 + (1-\alpha)\beta_2) \leq \alpha J(\beta_1) + (1-\alpha)J(\beta_2) - \frac{\nu}{2}\alpha(1-\alpha)\|\beta_1 - \beta_2\|^2$$

where ν is a positive strong-convexity constant.

Since the term $\frac{\nu}{2}\alpha(1-\alpha)\|\beta_1 - \beta_2\|^2$ is always positive, then we have that even if the function will be equal to the chord, it will be always above the chord, thus implying the strict convexity.

So we have the following chain of implications:

$$\text{strong convexity} \Rightarrow \text{strict convexity} \Rightarrow \text{convexity}$$

Note: we are assuming $\delta \in \mathbb{R}$ and $\nu \in \mathbb{R}$, but the definition holds only for positive values.

The Hessian matrix, indicated with $\nabla^2 J(\beta)$, is the matrix of second derivatives of a function. This matrix is a matrix of dimensions $p \times p$, where each element is $\frac{\partial^2 J(\beta)}{\partial \beta_i \partial \beta_j}$.

The notation $A > B$ for two symmetric matrices A and B means that $A - B$ is a **positive definite matrix**. If $A \geq B$ then $A - B$ is a **positive semi-definite matrix**. We can also use the symbol $\succ$ and $\succeq$.

When a matrix is positive definite it means that all its eigenvalues are positive. When a matrix is positive semi-definite it means that all its eigenvalues are non-negative.

We can rewrite the definitions of convexity in terms of the Hessian matrix. A function is **convex** if:

$$\nabla^2 J(\beta) \geq 0 \ \forall \beta$$

For example, let us consider the one dimensional case: if the second derivative is positive, then the function is convex. A function is **strictly convex** if:

$$\nabla^2 J(\beta) > 0 \ \forall \beta$$

Finally, a function is **strongly convex** if:

$$\nabla^2 J(\beta) \geq \nu I \ \forall \beta$$

Which means that for all the choice of the paramters β the Hessian matrix is greater or equal than the threshold ν times the identity matrix.

The parabola is the most common example of a strongly convex function. An example of function that is not strongly convex but convex is the exponential function e^{-x}, because we cannot find a ν such that the function is greater or equal than that threasold. This is also because the exponential function is bounded from below by zero.

The MSE cost used in linear regression is strongly convex if the covariance matrix $\mathbb{E}\left[XX^T\right]$ is invertible.

It can be shown that the covariance matrix is positive semi-definite and symmetric.

If the covariance matrix is invertible then the determinant is non-zero and thus each eigenvalue of the matrix is non-zero.

6.2 Gradient Descent

Finding the minimum of a function is not an easy problem because we need to find the point where the derivative is zero; but this alone does not guarantee that we have found the minimum, because it could be a maximum or a saddle point. And even if we ascertain that the point is a point of minimum, we don't know if it is a local or global minimum.

So algorithms like the gradient descent are very useful when our problem has a complex formulation. The basic idea of the gradient descent is to start from a random point and then move in the direction of the negative gradient, which is the direction of the steepest descent. The gradient descent is an iterative algorithm, which means that we need to repeat the same operation multiple times.

Each time we update the parameters in the gradient descent algorithm, we are moving by a step of size μ, which is usually called *step-size*. We can show that if μ is small enough, then the gradient descent algorithm converges to the minimum of the function, the size of μ is determined by δ and ν. But if we choose a step-size that is too large, then the algorithm may not converge.

If we use a **constant step-size**, then we can show that for μ smaller than a certain value (which depends on the Lipschitz constant δ and the strong-convexity constant ν), the gradient descent converges to the exact minimizer. The algorithm converges at a *geometric* rate on the order $O(\rho^i)$ where $\rho \in (0,1)$ is a decreasing function of the step-size μ and also depends on δ and ν.

If use a **decaying step-size**, then if we consider an iteration-dependent step-size $\mu = \mu(i)$, with $\sum_{i=0}^{+\infty} \mu(i) = +\infty$ and $\lim \mu(i) = 0$, the gradient descent converges to the exact minimizer with a *non-geometric* rate.

A typical choice is $\mu(i) = \frac{\tau}{i+1}$ with $\tau > 0$, yielding a convergence rate of $O(\frac{1}{i^{2\nu\tau}})$.

In the previous lecture we've seen that the Gradient Descent algorithm with constant step-size converges *exponentially*[1] to the optimal solution of the problem. With decaying step-size, the convergence is still guaranteed but it is slower.

6.3 Stochastic Gradient Descent

In practice we cannot compute the cost function $\mathbb{E}\left[Q_\beta(D)\right]$ because we do not know the distribution of D. However, by using our dataset $\{d_i\}_{i=1}^n$, we can compute the empirical risk:

$$J_{emp}(\beta) = \frac{1}{n}\sum_{i=1}^{n} Q_\beta(d_i)$$

In practice, we do not use this empirical risk because the computation of the gradient:

- it is not suited to large datasets, since we need to compute the gradient for each iteration of the algorithm

[1] In some books also linearly or geometrically

- it is not suited in the case of online learning, since our dataset changes over time and we want to track the evolution of the data
- it is not suited for distributed application

The solution is using an **instantaneous approximation of the gradient** based on individual samples:

$$\hat{\nabla J_i}(\beta) = \nabla Q_\beta(d_i)$$

We are moving from the expectation of the loss function to the empirical risk by applying the law of large numbers and then we use an approximation of the gradient of this empirical risk based on individual samples. This approximation is called **stochastic gradient**.

The stochastic gradient descent algorithm is the following:

Algorithm 1: Stochastic Gradient Descent

Set a starting point β_0
for $i = 1, 2, \ldots$ **do**
  take a fresh sample d_i
  $\beta_i = \beta_{i-1} - \mu\hat{\nabla J_i}(\beta_{i-1})$
end

Note: β_i is a stochastic process and the instantaneous approximation of the gradient is a *noisy* estimate of the true gradient based on the current sample d_i.

As time progresses, we are putting more samples. At the end we are computing the average of the gradients of the loss function for each sample but we are doing this over time.

Constant step-size We can show that with constant step-size, the stochastic gradient descent *iterates* (meaning the β_i) never reach the optimal solution. This is due to the *gradient noise* that is the *inherent* randomness of the gradient.

If we see a plot we can see that the iterates are *oscillating* around the optimal solution.

If we use a smaller step size, we are focusing in a large quantity of data in the sliding window.

For a μ smaller than a certain value (which depends on δ and ν), the iterates β_i oscillates in a smaller neighbourhood of the true minimizer β^*. The error $\mathbb{E}\left[\|\beta_i - \beta^*\|^2\right]$ converges to a steady-state error $O(\mu)$ at a geometric rate $O(\rho^i)$ (exponential convergence).

[TODO: add plot]

The solution to this problem is making μ smaller, but then we will surely have a slower convergence, this is because if μ is smaller then ρ is higher and this means that it will take more time to converge.

Decaying step-size We can show that with decaying step-size, the stochastic gradient descent *iterates* (meaning the β_i) converge to the exact minimizer and the error $\mathbb{E}\left[\|\beta_i - \beta^*\|^2\right]$ converges to zero.

If $\mu(i) = \frac{\tau}{i+1}$ for $\tau > \frac{1}{\nu}$ the convergence rate it $O(\frac{1}{i})$.

In conclusion, we can say that decaying step-sizes converge to the exact minimizer but they are not suited to online tasks, because the algorithm cannot react to data drifts since the updates are given less importance due to the increasingly smaller values of $\mu(i)$. We can still use it in *batch* applications rather than online tasks.

By contrast, constant step-sizes are suited to online tasks, but they do not converge to the exact minimizer. However, they converge to a neighbourhood of the minimizer and they are able to react to data drifts. The smaller the μ, the better the accuracy of the algorithm, but the slower the convergence.

Note: we can say in practice that the stochastic gradient descent with constant step-size allows us to react to data drifts within a fixed number of samples that it is about $\frac{1}{\mu}$. This means that if we set $\mu = 0.01$ we can react to data drifts within 100 samples.

Chapter 7

Cluster Analysis

7.1 PCA

Principal Component Analysis is a linear transformation that projects the data onto the space. The feature space is described in terms of the principal components. PCA has the following characteristics:

- the transformed features are uncorrelated
- the first transformed feature has the highest variance among the others, the second transformed feature has the second highest variance among the others, and so on...
- If our features indicated something like the temperature, the pressure and so on, then after projecting them onto the new space, they will lose that meaning.

An example of akin transformation is the Fourier transform, that maps all the samples of a sinusoidal function into one single features that represents the frequency. In this case, the *base* of the space in which we want to project the new features is known, but in the principal component analysis case we need to learn first the *bases* of the new space from the data.

Principal Component Analysis Construction Assume we have a dataset $X' \in \mathbb{R}^{n \times p}$. A preliminary thing to do is to center the dataset by subtracting the mean μ of each feature from the corresponding feature.

$$\mu = \frac{1}{n} \sum_{i=1}^{N} x'_i$$

$$X = \begin{bmatrix} x'_1 - \mu_1 \\ x'_2 - \mu_2 \\ \vdots \end{bmatrix}$$

Principal Component Analysis applying the following linear transformation:

$$T = XW$$

where $W \in \mathbb{R}^{p \times p}$ is the projection matrix which contains the **principal directions** as columns and $T \in \mathbb{R}^{n \times p}$ are the **principal components** of X, i.e. the new representation of X in the space described by the principal directions. By construction, we assume that the principal directions are orthonormal, i.e. $W^T W = I$.

Each row of X is a sample of the dataset, each column of W is a principal direction, so the dot product between a row of X and a column of W is the projection of the sample onto the principal direction. In other words, By mutiplying X by W we are applying the dot product on each feature, thus projecting each feature onto the principal directions.

Now we want to understand *how do we obtain the matrix W*. We know that the **first** column of T should have the highest variance among the others. So we will need to solve the following optimization problem:

$$w^{(1)} = \arg\max_{\omega \in \mathbb{R}^p} \left\{ \frac{1}{n-1} \sum_{i=1}^{n} \left(\sum_{i=1}^{P} x_{ij} \omega_j \right)^2 \right\} \quad \text{s.t.} \|w\|^2 = 1$$

This problem is constrained by the fact that the norm of w must be equal to 1 because we impose that the principal directions are orthonormal.

Note that the inner sum is the dot product between the i-th sample and the respective principal direction w so it gives us the squared sum of the principal components, which averaged for the number of samples gives us the variance of the j-th principal component (recall x is zero-mean because we scaled the features).

Rewriting the problem in matrix notation:

$$w^{(1)} = \arg\max_{\omega \in \mathbb{R}^p} \|X\omega\|^2 = \arg\max \left(\omega^T X^T X \omega \right) \quad \text{s.t.} \|w\|^2 = 1$$

Under the assumption of orthonormality, we can rewrite the problem as:

$$w^{(1)} = \arg\max \left(\frac{\omega^T X^T X \omega}{\omega^T \omega} \right) \quad \text{s.t.} \|w\|^2 = 1$$

The function we want to maximize is known as **Rayleigh quotient** and we know that if $X^T X$ is a positive semi-defined matrix then the Rayleigh quotient is maximized by **the eigenvector corresponding to the largest eigenvalue** of $X^T X$.

Now in order to find the other principal directions we first need to find the orthogonal part of the projection, by projecting the data onto the space spanned by the first principal direction and then subtracting the projection from the data. We can visually understand this by looking at the following figure:

[TODO: add figure]

We can perform this projection by applying the following transformation:

$$X_k = X - X \sum_{i=1}^{k-1} w^{(i)} w^{(i)T}$$

Where k is the number of the principal direction we want to compute. We can now apply the same procedure as before to find the k-th principal direction. Then it can be shown that the k-th principal direction is the eigenvector corresponding to the k-th largest eigenvalue of X^TX.

To sum up we've shown that the principal directions are the eigenvectors of the covariance matrix X^TX and the principal components are the eigenvectors associated to the largest eigenvalues of X^TX.

Data Covariance Matrix The covariance matrix of $X \in \mathbb{R}^{n \times p}$ is defined as:

$$C = \frac{1}{n-1} \sum_{i=1}^{n} x_i^T x_i = \frac{1}{n-1} X^T X$$

We will now show that the eigenvalues are the variances of each principal component. Let us consider the first eigenvalue, by definition of eigenvalue and eigenvector we have:

$$Au^{(k)} = \lambda_k u^{(k)} \rightarrow X^T X w^{(1)} = \lambda_1 w^{(1)}$$

Multiplying both sides by $(w^{(1)})^T$ we get:

$$(w^{(1)})^T X^T X w^{(1)} = \lambda_1 w^{(1)} (w^{(1)})^T \rightarrow \|Xw^{(1)}\|^2 = \lambda_1 \|w^{(1)}\|^2 = \lambda_1$$

Then by definition of variance we have:

$$\text{Var}\left[Xw^{(1)}\right] = \frac{1}{n-1} \|Xw^{(1)}\|^2 = \frac{\lambda_1}{n-1}$$

So we have shown that the eigenvalues of X^TX are (proportional to) the variances of the principal components.

Another characteristics of the principal component analysis is that it *gives us uncorrelated data* in the new space. To show this, let us consider the covariance matrix of the principal components:

$$\frac{T^T T}{n-1} = \frac{W^T X^T X W}{n-1} = \frac{W^T W \Lambda W W^T}{n-1} = \frac{\Lambda}{n-1}$$

First we used the fact that X^TX is similar to $W\lambda W^T$ and then the fact that $W^TW = I$ because we imposed orthogonality. So we have shown that the covariance matrix of the principal components is a diagonal matrix, which means that all the covariances are zero, i.e. the principal components are uncorrelated.

Dimensionality Reduction We can use PCA to select only a subset of size m of the principal components, thus reducing the dimensionality of the data. We can do this by selecting the first m columns of T.

In this way we are selecting the m principal component that explain the most variance of the data. We can compute the **percentage of variance explained** or **PVE** by the m principal components as:

$$PVE = \frac{\sum_{k=1}^{m} \lambda_k}{\sum_{j=1}^{p} \lambda_j}$$

The main drawback of applying PCA is that we are losing the meaning of the features (explainability), so we are not able to interpret the data anymore.

PCA and Singular Value Decomposition **Singular Value Decomposition** is a generalization of the eigenvalue decomposition for rectangular matrices. Let us consider a matrix $M \in \mathbb{R}^{n \times p}$, then we can decompose it as:

$$M = U\Sigma V^T$$

where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{p \times p}$ are orthogonal matrices respectively called **left singular vectors** and **right singular vectors**, while $\Sigma \in \mathbb{R}^{n \times p}$ is a *rectangular diagonal* matrix with the singular values of M. Since the elements under the diagonal of Σ are zero, we can cut the matrix to simplify the calculations.

If we decompose X as $X = U\Sigma V^T$ then we can rewrite the covariance matrix as:

$$X^T X = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T$$

because U is defined as an orthogonal matrix, so $U^T U = I$. So we found out that the covariance matrix of X is similar to $\Sigma^T\Sigma$ and we know that similar matrices have the same eigenvalues.

If we call σ_i the i-th singular value of X then we have that σ_i^2 is equal to the i-th eigenvalue of $X^T X$.

We can also show that:

$$T = XV = U\Sigma V^T V = U\Sigma$$

7.2 Clustering

Clustering

Clustering are a familiy of unsupervised learning problems, where we we want to group objects in non-overlapping subsets, called clusters, according to a criterion of *similarity*.

In other kind of problems, such as classiifcation and regression, we want to minimize the error terms, while in clustering we want to maximize the similarity between the objects in the same cluster.

There are four main families of clustering:

- **Combinatorial**, where we want to minimize over the set of all possible assigments, a suitable function based on the chosen similarity measure. This is a combinatorial problem since we have a large number of possible assignments.
- **Distribution-based**, where we assume that the data observations have been drawn from a mixture of **generative models**, and we want to find the parameters of the models. The parameters of the models are estimated from the observation using an *expectation-maximization* algorithm. After the parameters have been estimated, we can assign each observation to a model according to the probability of being generated by one of the models.

- **Hierarchical**, assign each data observation to a cluster according to the similairt among pair of groups of observations. This is done by building a *tree structure* to represent the data. Such structure can be obtained by using a *bottom-up* or *top-down* approach and the algorithms are called respectively *agglomerative* or *divisive* clustering.
- **Density-based**, where clusters follow more closely the spatial arrangement of the data. The clusters are defined as regions with densely packed observation. Density is usually intendend as the number of observations within a given volume.

K-Means

The K-Means algorithm is a combinatorial clustering algorithm.

The number of clusters K is a parameter of the algorithm, and it is usually chosen by the user. The algorithm is iterative and its goal is to find the optimal assigment of the observation that minimizes the following sum of squares:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

Where r_{nk} is a variable that is equal to 1 if the observation x_n is assigned to the cluster C_k, and 0 otherwise; while μ_k is the **centroid** of the cluster C_k, defined as:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$

The centroid can be seen as the *baricenter* of the cluster C_k. **Note** that we are minimizing a quantity that represents how dispersed the observations are around the centroid of each cluster.

There is a problem with this formulation, that is the fact that the centroids are a function of the assigments, but we don't know both of them. So a *naive* solution would be to enumerate both of them and choose the best one, and this is why it is called a combinatorial problem.

However, we know that given the centroids, the assigments r_{nk} should be:

$$r_{nk} = 1 \text{if } k = \arg\min_j \|x_n - \mu_j\|^2$$

and 0 otherwise. This is called the **nearest neighbor condition**. If the assigments follow this rule, this will reduce the objective function. Given the assigments, the centroids can be computed by definition **centroid condition**.

Convergence At each step of the algorithm, the objective function J is becoming smaller and smaller. This is not sufficient to prove that the algorithm converges, because the algorithm may diverge to -∞. However, sinche the objective function is bounded below by 0 (because it is a squared sum) the algorithm must converge, at least to a local minimum.

Algorithm 2: Lloyd's algorithm for K-Means

Input: data set X, number of clusters K, initial centroids μ_k
repeat
  form K clusters assigning each observation to the nearest centroid
  recompute the centroids μ_k of the clusters
until *termination criterion is met*;

Even if both the neighborhood condition and the centroid condition are satisfied, the algorithm may not converge to the global optimum. These two conditions are necessary but not sufficient for convergence to the global optimum.

In the last lecture, we've showed that the K-Means algorithm is guaranteed to converge to a local minimum of the objective function under the nearest neighbor condition and the centroid condition.

If we initialize the centroids in the right way, the algorithm will converge to the global minimum. However, if we initialize the centroids in the wrong way, the algorithm may converge to a local minimum. This is because there are different configurations of the centroids that satisfy the two conditions, but for us they are not equivalent.

Another thing is that the algorithm will converge in a finite number of steps, but the number of steps may be very large, due to the combinatorial nature of the problem. In practice, the algorithm is stopped when the change in the objective function is below a certain threshold.

Gaussian Mixture Model

The Gaussian Mixture Model is a distribution-based clustering algorithm. The idea is that we assume that the observations have been generated by a mixture of K Gaussian distributions, each of which is representative of a cluster. The parameters of the model are estimated from the observations using an *expectation-maximization* algorithm.

With K clusters, the mixture model for the distribution of the entry x_n is defined as:

$$p(x_n) = \sum_{k=1}^{K} \pi_k p_k(x_n \mid \theta_k)$$

where the coefficients π_k are called **mixing probabilities**, and they are constrained to be non-negative and to sum to 1. The mixing probabilities are to be learned from the data.

In the Gaussian Mixture Model, the individual likelihoods $p_k(\cdot \mid \theta_k)$ are h-dimensional Gaussian distributions, parametrized by the mean μ_k and the covariance matrix Σ_k:

$$p_k(x \mid \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{h}{2}} \sqrt{\det \Sigma_k}} \exp\left(\frac{-}{1} 2(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\right)$$

The unknown parameters of the model are the mixing probabilities π_k, the means μ_k and the covariance matrices Σ_k for $k = 1, \ldots, K$.

We could use the maximum likelihood estimator to estimate the parameters of the model, but there is no closed form solution for the maximization, because all of the quantities we find depend on a term called **responsibility** γ_{nk} that is not known and depends on the parameters on the model.

$$\gamma_{nk} = \frac{\pi_k p_k(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j p_j(x \mid \mu_j, \Sigma_j)}$$

while the parameters can be computed as:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}$$

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma_{nk}}$$

This is why we use the expectation-maximization algorithm. The algorithm is iterative and the basic idea is to estimate in an alternating manner the estimation of unknown coefficients and responsibilities.

Algorithm 3: EM algorithm for Gaussian Mixture Model

Input: data set X, number of clusters K, initial parameters μ, Σ, π

repeat

  Expectation step: with the parameters fixed we compute the responsibility

  Maximization step: with the

until *termination criterion is met*;

The Expectation-Maximization algorithm is a general algorithm used not only in GMM clustering to find maximum likelihood solutions when there are latent variables.

A latent variable is a variabile that cannot be observed directly, but it is useful to explain the observed data. In the case of the Gaussian Mixture Model, the latent variable is the Gaussian from which the observation has been generated.

The Expectation-Maximization algorithm actually maximizes a surrogate objective function which allows to maximize the likelihood function of interest, because in the expectation step it computes the expectation of the log-likelihood with respect to the conditional distribution of the latent variables given the observations, while the maximization step find the parameters that maximize the expectation and will then be used for the next expectation step.

The convergence conditions are similar to the K-Means algorithm. EM is not guaranteed to converge to the global maximum of the likelihood, but it is guaranteed to increase the value of the likelihood function at each step.

Hierarchical Clustering

Hierarchical clustering can be performed in two ways: agglomerative or divisive. We will focus on the **agglomerative clustering**, which starts from **singleton clusters**, that are the single data observations. At each step of the algorithm, the two most similar clusters are merged. There are different *similarity measures* that we can use to find similar clusters.

Algorithm 4: EM algorithm for Gaussian Mixture Model

Input: data set X
repeat
| Examine all pairs of subgroups, and merge the most similar
until *One single cluster remains*;

The dissimilarity $d(G, H)$ between two groups of observations G and H is computed from the pairwise dissimilarities $d_{ii'}$ between the observations in the two groups. There are different ways to compute the dissimilarity between two groups:

- **Single-linkage**
- **Complete-linkage**
- **Group average**
- **Ward's criterion**

Single Linkage The single linkage measure is defined as:

$$d_{SL}(G, H) = \min_{i \in G, i' \in H} d_{ii'}$$

The similarity among subgroups is based on the similarity between the **nearest observations**. This method tends to produce **elongated clusters**, with non-elliptical shapes and it is sensitive to noise and outliers.

Complete Linkage The complete linkage measure is defined as:

$$d_{CL}(G, H) = \max_{i \in G, i' \in H} d_{ii'}$$

The similarity among subgroups is based on the similarity between the **farthest observations**. This method tends to produce compact clusters, because it merges the smallest diameter clusters first. It is still sensitive to noise and outliers.

Average Group The average group measure is defined as:

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}$$

The similarity among subgroups is based on the average similarity between the "centroid" of the groups. This methods is in between the single and complete linkage.

Ward's Criterion Ward's criterion is defined as:

$$d_W(G,H) = \frac{N_G N_H}{N_G + N_H} \|\mu_G - \mu_H\|^2$$

where μ_G and μ_H are the centroids of the groups G and H respectively.

It can be shown that this method is equivalent to measuring the increase of variance when merging two groups.

Dendogram The final output of the agglomerative clustering is a binary tree structure called **dendogram**, in which the root of this structure represents the entire dataset, while the leaves represent singleton clusters. Only by slicing the dendogram at a given height we obtain the cluster, this means that, differently from the other algorithms, K is not an input of the algorithm but a parameter chosen by the user. K depends on the specific application domain.

To choose the number of clusters, there are different *heuristics* scores that can be used:

- **Silhouette score**
- **Caliniski-Harabasz score**

Silhouette Score

Caliniski-Harabasz score

DBSCAN

DBSCAN is a density-based clustering algorithm. This algorithm segments the data observations by looking at dense regions with a given *reachability*.

Differently from other clustering algorithms, can classify points as *noisy points*. This is because some points may not belong to any cluster, because they are not dense enough...